Corrigés

Introduction

Naissance d'un programme

Exercice I-1: Apprendre à décomposer une tâche en sous-tâches distinctes

Corrigé

a. Objets nécessaires : `1 tableau, 1 clou, 2 pointes, 1 ficelle, 1 marteau, 1 crayon, 1 mètre`.

b. Liste des opérations :

```
Mesurer le mur en hauteur, le mur en largeur, le tableau en hauteur ;
Calculer le centre du mur, le tiers de la hauteur du tableau ;
Tracer une marque au centre du mur, sur le cadre (face arrière) du tableau ;
Prendre le marteau, le tableau, le mètre, le crayon, la ficelle, le clou, la pointe ;
Poser le marteau, le tableau, le mètre, le crayon ;
Enfoncer la pointe, le clou ;
Accrocher la ficelle à la pointe, la ficelle au clou ;
Ajuster le tableau ;
```

c. Liste ordonnée des opérations :

```
Prendre le mètre
Mesurer le mur en hauteur ;
Mesurer le mur en largeur ;
Poser le mètre ;
Calculer le centre du mur ;
Prende le crayon ;
Tracer une marque au centre du mur ;
Poser le crayon ;
Prendre le marteau ;
Prendre le clou ;
Enfoncer le clou dans le mur ;
Poser le marteau ;
Prendre le mètre
Mesurer le tableau en hauteur ;
Poser le mètre
Calculer le tiers de la hauteur du tableau ;
Prende le crayon ;
Tracer une marque sur le bord gauche du cadre (face arrière) au tiers de la hauteur ;
Tracer une marque sur le bord droit du cadre (face arrière) au tiers de la hauteur ;
Poser le crayon ;
```

```
Prendre le marteau ;
Prendre une pointe ;
Enfoncer une pointe sur la marque de droite ;
Prendre une pointe ;
Enfoncer une pointe sur la marque de gauche ;
Poser le marteau ;
Accrocher la ficelle à la pointe de droite ;
Accrocher la ficelle à la pointe de gauche ;
Accrocher la ficelle au clou ;
Ajuster le tableau ;
```

Exercice I-2 : Observer et comprendre la structure d'un programme Java

Corrigé

```
import java.util.*;
public class Premier {
 public static void main(String [] argument) {
      double a;
      Scanner lectureClavier = new Scanner(System.in);
      System.out.print("Entrer une valeur : ") ;
      a = lectureClavier.nextDouble();
      System.out.print("Vous avez entre : " + a) ;
 }
}
```

a. Repérez les instructions définissant la fonction `main()` : voir tracé **orange** sur le programme ci-dessus.
   Celles délimitant la classe `Premier` : voir tracé **vert** sur le programme ci-dessus.

b. Recherchez les instructions d'affichage : voir tracé **jaune** sur le programme ci-dessus.

c. L'instruction `double a;` a pour rôle de réserver une case mémoire afin d'y stocker une valeur réelle de double précision. Cette case à pour nom d'appel `a`.

d. Exécution du programme :
   Le message `Entrer une valeur` s'affiche à l'écran ;
   L'utilisateur tape `10` au clavier et, puis sur la touche `Entrée` ;
   Le message `Vous avez entre : 10` s'affiche à l'écran

Exercice I-3 : Observer et comprendre la structure d'un programme Java

Corrigé

```
import java.util.*;
public class Carre{ // Donner un nom à la classe
 public static void main(String [] argument) {
      // Déclaration des variables représantant le périmètre et le coté
      double périmètre, côté ;
      Scanner lectureClavier = new Scanner(System.in);
      // Afficher le message "Valeur du cote : " à l'écran
      System.out.print("Valeur du cote : ");
```

```
        // Lire au clavier une valeur
        // placer cette valeur dans la variable correspondante
        côté = lectureClavier.nextDouble();
        // Calculer le périmètre du carré
        périmètre = 4 * côté ;
        // Afficher le résultat
        System.out.print("Perimetre: " + périmètre);
 }
}
```

Exercice I-4 : Ecrire un premier programme Java

Corrigé

a. Nombre de variables à déclarer : 3, une pour la surface, une pour la largeur, une pour la longueur.

b. Nombre de valeurs à saisir au clavier : 2, la largeur et la longueur.

```
import java.util.*;
public class Rectangle{ // Nom à la classe
 public static void main(String [] argument){
        // Déclaration des variables
        double surface, largeur, longueur ;
        Scanner lectureClavier = new Scanner(System.in);
        // Afficher un message à l'écran
        System.out.print("Valeur de la longueur : ");
        // Lire au clavier une valeur
        longueur = lectureClavier.nextDouble();
        // Afficher un message à l'écran
        System.out.print("Valeur de la largeur : ");
        // Lire au clavier une valeur
        largeur = lectureClavier.nextDouble();
        // Calculer le surface du rectangle
        surface = largeur * longueur;
        // Afficher le résultat
        System.out.print("Surface: " + surface);
 }
}
```

Partie 1 : Outils et techniques de base

Chapitre 1 : Stocker une information

Exercice 1-1 : Repérer les instructions de déclaration, observer la syntaxe d'une instruction

Corrigé

a. Déclaration de trois entiers nommés `i`, `j`, `valeur` ;

b. Opération non valide, pas d'opérateur à gauche de l'affectation ;

c. Instruction d'affectation, pas de déclaration

d. Déclaration non valide, une variable ne peut s'appeler `char`

e. Opération non valide, ce n'est pas une instruction ;

f. Déclaration d'un entier nommé `X` ;

g. Déclaration d'un réel simple précision, nommé `A` ;

h. Affectation, pas une déclaration ;

i. Affectation non valide, un `float` ne peut être affecté à un entier ;

j. Affectation, pas une déclaration ;

Exercice 1-2 : Comprendre le mécanisme de l'affectation

Corrigé

a.			
Instructions	A	B	C
`float A = 3.5f ;`	3.5f	-	-
`float B = 1.5f ;`	3.5f	1.5f	-
`float C ;`	3.5f	1.5f	-
`C = A + B ;`	3.5f	1.5f	5.0f
`B = A + C ;`	3.5f	8.5f	5.0f
`A = B ;`	8.5f	8.5f	5.0f

b.				
Instructions	A	B	C	D
`double A = 0.1 ;`	0.1	-	-	-
`double B = 1.1 ;`	0.1	1.1	-	-
`double C, D ;`	0.1	1.1	-	-
`B = A ;`	0.1	0.1	-	-
`C = B ;`	0.1	0.1	0.1	-
`D = C ;`	0.1	0.1	0.1	0.1
`A = D`	0.1	0.1	0.1	0.1

Exercice 1-3 : Comprendre le mécanisme de l'affectation

Corrigé

a.		
Instructions	a	b
`int a = 5, b ;`	5	-
`b = a + 4 ;`	5	9
`a = a + 1`	6	9
`b = a - 4 ;`	6	2

b.	
Instructions	valeur
`int valeur = 2 ;`	2
`valeur = valeur + 1 ;`	3
`valeur = valeur * 2`	6
`valeur = valeur % 5;`	1

c.			
Instructions	x	y	z
`int x = 2, y = 10, z ;`	2	10	-
`z = x + y ;`	2	10	12
`x = 5;`	5	10	12
`z = z - x ;`	5	10	7

Exercice 1-4 : Comprendre le mécanisme d'échange de valeurs

Corrigé

1.			2.		
	a	b		a	b
`int a = 5`	5	-	`int a = 5`	5	-
`int b = 7`	5	7	`int b = 7`	5	7
`a = b`	7	7	`b = a`	5	5
`b = a`	7	7	`a = b`	5	5

Exercice 1-5 : Comprendre le mécanisme d'échange de valeurs

Corrigé

Les instructions `a = b ; b = a ;` ne permettent pas l'échange de valeurs puisque la valeur contenue dans la variable `a` est perdue dès la première instruction (voir exercice 1-4, ci-dessus).

Les instructions `t = a ; a = b ; b = t ;` permettent l'échange des valeurs entre `a` et `b`, puisque la valeur de `a` est mémorisée dans la variable `t`, avant d'être effacée par le contenu de `b`.

Les instructions `t = a ; b = a ; t = b ;` ne permettent pas l'échange des valeurs car, la première instruction mémorise le contenu de la variable `a`, alors la seconde instruction efface le contenu de `b`.

Exercice 1-6 : Comprendre le mécanisme d'échange de valeurs

Corrigé

```
tmp = c ; c = b ; b = a ; a = tmp;
```

Exercice 1-7 : Comprendre le mécanisme d'échange de valeurs

Corrigé

Instruction	a	b
`initialisation`	2	5
`a = a + b ;`	7	5
`b = a - b;`	7	2
`a = a - b ;`	5	2

Partant de `a = 2` et `b = 5`, nous obtenons `a = 5` et `b = 2`. Ainsi, grâce à ce calcul, les valeurs de `a` et `b` sont échangées sans utiliser de variable intermédiaire.

Exercice 1-8 : Calculer des expressions mixtes

Corrigé

a. `i = 16` `i` est un entier, le résultat de la division est donc un entier ;

b. `j = 4` `4` est le reste de la division entière de `100` par `6` ;

c. `i = 5` `5` est le reste de la division entière de `5` par `8` ;

d. `(3 * 5 - 2 * 4) / (2 * 2.0 - 3.0)` ⇒ `(15 - 8) / (4.0 - 3.0)`
⇒ `(7) / (1.0)` ⇒ `7.0`

e. `2 * ((5 / 5) + (4 * (4 - 3)) % (5 + 4 - 2)` ⇒ `2 * 5 % 7`
⇒ `10 % 7` ⇒ `3`

f. `(5 - 3 * 4) / (2.0 + 2 * 3.0) / (5 - 4)` ⇒ `(5 - 12) / (2.0 + 6.0) / 1`
⇒ `-7 / 8.0` ⇒ `-0.875`

Exercice 1-9 : Calculer des expressions mixtes

Corrigé

a.

x + n % p	⇒	2.0f + 10 % 7	⇒	2.0f + 3	⇒	5.0f	
x + n / p	⇒	2.0f + 10 / 7	⇒	2.0f + 1	⇒	3.0f	
(x + n) / p	⇒	(2.0f + 10) / 7	⇒	12.0f / 7	⇒	1.7142857f	
5. * n	⇒	5. * 10	⇒	50.0f			
(n + 1) / n	⇒	(10 + 1) / 10	⇒	11 / 10	⇒	1	
(n + 1.0) / n	⇒	(10 + 1.0) / 10	⇒	11.0 / 10	⇒	1.1	
r + s / t	⇒	8 + 7 / 21	⇒	8 + 0	⇒	8	

b.

r + t / s	⇒	8 + 21 / 7	⇒	8 + 3	⇒	11	
(r + t) / s	⇒	(8 + 21) / 7	⇒	29 / 7	⇒	4	
r + t % s	⇒	8 + 21 % 7	⇒	8 + 0	⇒	8	
(r + t) % s	⇒	(8 + 21) % 7	⇒	29 % 7	⇒	1	
r + s / r + s	⇒	8 + 7 / 8 + 7	⇒	8 + 0 + 7	⇒	15	
(r + s) / (r + s)	⇒	(8 + 7) / (8 + 7)	⇒	15 / 15	⇒	1	
r + s % t	⇒	8 + 7 % 2	⇒	8 + 1	⇒	9	

Exercice 1-10 : Comprendre le mécanisme du cast

Corrigé

```
i1 = valeur / chiffre  ⇒ 7 / 2      ⇒ 3
i2 = chiffre / valeur  ⇒ 2 / 7      ⇒ 0
f1 = (float) (valeur / chiffre)  ⇒ (float) (7 / 2) ⇒ (float) 3 ⇒ 3.0f
f2 = (float) (valeur / chiffre) + 0.5f  ⇒ 3.0f + 0.5f ⇒ 3.5f
i1 = (int) f1 ⇒ (int) 3.0f   ⇒ 3
i2 = (int) f2 ⇒ (int) 3.5f   ⇒ 3
f1 = (float) valeur / (float)chiffre ⇒ 7.0f / 2.0f ⇒ 3.5f
f2 = (float) valeur / (float)chiffre + 0.5f   ⇒ 3.5f + 0.5f ⇒ 4.0f
i1 = (int) f1 ⇒ (int) 3.5f   ⇒ 3
```

```
i2 = (int) f2⇒  (int) 4.0f    ⇒  4
```

Chapitre 2 : Communiquer une information

Exercice 2- 1 : Comprendre les opérations de sortie

Corrigé

```
Vous avez entre : 223
Pour un montant de 335.5 le total vaut : 223135
Apres reduction de 20.5 %, vous gagnez : 68.8 Euros
La variable R = R et T = T
```

Exercice 2-2 : Comprendre les opérations de sortie

Corrigé

```
System.out.println("x = " + x + " et y = " + y) ;
System.out.println("Racine carree de " + x + " = " + Math.sqrt(x)) ;
System.out.print(x + " a la puissance " + y + " = "+ Math.pow(x,y)) ;
```

Exercice 2-3 : Comprendre les opérations d'entrée

Corrigé

Dans le premier cas, lorsque l'utilisateur fournit au clavier 2, puis 3, puis 4, le programme affiche :

```
X = 7Y = 3
```

Dans le second cas, lorsque l'utilisateur fournit au clavier 2, le programme affiche :

```
X = 2Y = 0
```

Exercice 2-4 : Observer et comprendre la structure d'un programme Java

Corrigé

```
import java.util.*;
public class Euro {
      public static void main (String [] argument) {
         double F, E = 0.0 ;
         double T = 6.55957 ;
         Scanner lectureClavier = new Scanner(System.in);
         System.out.print("Nombre de Francs : ") ;
         F = lectureClavier.nextDouble( ) ;
         E = F / T ;
         System.out.println("Conversion F/E : " + T) ;
         System.out.println("Nombre d'Euro : " + E) ;
      }
}
```

Chapitre 3 : Faire des choix

Exercice 3-1 : Comprendre les niveaux d'imbrication

Corrigé

Si la valeur saisie au clavier est **4**, le résultat affiché à l'écran est :

```
Entrer un chiffre : 4
Pour 4 le resultat est 2
```

Explication : `x` a pour valeur `4`. Le contenu de la variable `x` est donc supérieur ou égal à `0`. Le programme exécute donc l'instruction `r = Math.sqrt (4)`

Si la valeur saisie au clavier est **-9**, le résultat affiché à l'écran est :

```
Entrer un chiffre : -9
Pour -9 le resultat est 3
```

Explication : `x` a pour valeur `-9`. Le contenu de la variable `x` est donc strictement inférieur à `0`. Le programme exécute le bloc `else`, c'est à dire l'instruction `r = Math.sqrt (-(-9))`

Exercice 3-2 : Construire une arborescence de choix

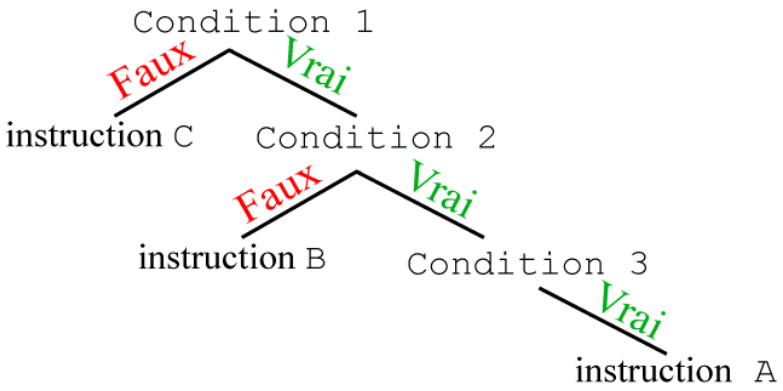
Corrigé

```
import java.util.*;
public class Maximum{
 public static void main (String [] parametre){
      int première, deuxième, laPlusGrande ;
      Scanner lectureClavier = new Scanner(System.in);
      System.out.print("Entrer une valeur :") ;
      première =lectureClavier.nextInt() ;
      System.out.print("Entrer une deuxieme valeur :") ;
      deuxième =lectureClavier.nextInt() ;
      if (première > deuxième)
      {
         System.out.println(deuxième + "" + première) ;
         laPlusGrande = première ;
         System.out.println("La plus grande valeur est : " + laPlusGrande) ;
      }
       else
      {
         if (première < deuxième) {
         System.out.println(première + " " + deuxième) ;
         laPlusGrande = deuxième ;
         System.out.println("La plus grande valeur est : " + laPlusGrande) ;
      }
         else System.out.println("Les deux valeurs saisies sont identiques") ;
      }
 } // Fin du main ()
```

```
} // Fin de la Class Maximum
```

Exercice 3-3 : Construire une arborescence de choix

Corrigé

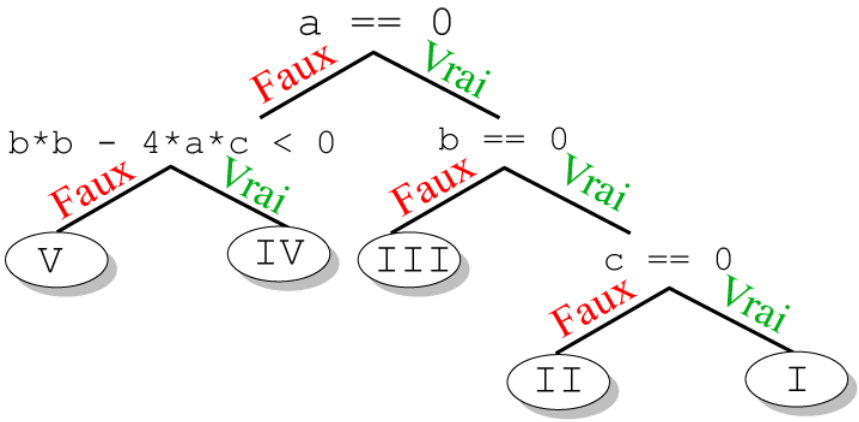


Les deux accolades fermantes situées après l'`instruction B` font que le bloc `else instruction C` est rattaché au premier `if (condition1)`.

Exercice 3-4 : Construire une arborescence de choix

Corrigé

a. Arbre des choix :



b. Les variables à déclarer sont :

```
double a, b, c, x, x1, x2 ;
```

c. Les instructions `if-else` s'écrivent de la façon suivante :

```
if (a == 0)
{
  if (b == 0)
  {
     if (c == 0)
     {
        // bloc d'instructions I
     }
     else
     {
```

```
            // bloc d'instructions II
        }
    }
    else
    {
        // bloc d'instructions III
    }
}
else
{
    if (b*b - 4*a*c < 0)
    {
        // bloc d'instructions IV
    }
    else
    {
        // bloc d'instructions V
    }
}
```

d. Dans chaque bloc `if` ou `else`, les instructions de calcul et d'affichage appropriées sont les suivantes :

- Le bloc d'instructions `I` :

```
System.out.println("tout reel est solution") ;
```

- Le bloc d'instructions `II` :

```
System.out.println("il n'y a pas de solution") ;
```

- Le bloc d'instructions `III` :

```
x = -c/b ;
System.out.println("la solution est " + x) ;
```

- Le bloc d'instructions `IV` :

```
System.out.println("il n'y a pas de solution dans les reels") ;
```

- Le bloc d'instructions `V` : Attention de bien placer les parenthèses pour obtenir un résultat cohérent.

```
x1 = (-b + Math.sqrt(b*b - 4*a*c)) / (2*a) ;
x2 = (-b - Math.sqrt(b*b - 4*a*c)) / (2*a) ;
System.out.println("il y deux solutions egales a " + x1 + " et " + x2) ;
```

e. En insérant l'ensemble des instructions dans la classe `SecondDegre` et à l'intérieur d'une fonction `main()`, le programme complet s'écrit de la façon suivante :

```
import java.util.*;
public class SecondDegre {
   public static void main (String [] arg)   {
      double a, b, c, delta ;
      double x, x1, x2 ;
```

```
        Scanner lectureClavier = new Scanner(System.in);
        System.out.print("Entrer une valeur pour a : ") ;
        a =lectureClavier.nextDouble() ;
        System.out.print("Entrer une valeur pour b : ") ;
        b =lectureClavier.nextDouble() ;
        System.out.print("Entrer une valeur pour c : ") ;
        c =lectureClavier.nextDouble() ;
        if (a == 0)    {
           if (b == 0)    {
              if (c == 0)    {
                 System.out.println("tout reel est solution") ;
              }
              else  {
                 System.out.println("il n'y a pas de solution") ;
              }
           }
           else  {
              x = -c / b ;
              System.out.println("la solution est " + x) ;
           }
        }
        else  {
           delta = b*b - 4*a*c ;
           if (delta < 0)    {
              System.out.println("il n'y a pas de solution dans les reels") ;
           }
           else  {
              x1 = (-b + Math.sqrt(delta))/ (2*a) ;
              x2 = (-b - Math.sqrt(delta))/ (2*a) ;
              System.out.println("il y deux solutions egales a " + x1 + " et " + x2) ;
           }
        }
     }
  }
```

Remarquez les instructions de saisie des trois coefficients `a`, `b`, `c` nécessaires à la bonne marche du programme ainsi que l'utilisation d'une variable intermédiaire `delta` utilisée pour éviter la répétition du même calcul `b*b - 4*a*c`.

Exercice 3-5 : Manipuler les choix multiples, gérer les caractères

Corrigé

a. Le code source complet :

```
import java.util.*;
public class Calculette {
```

```
public static void main( String [] argument) {
 int a, b;
 char opérateur;
 double calcul = 0;
 Scanner lectureClavier = new Scanner(System.in);
 // Lire et stocker la première valeur dans a
 System.out.print("Entrer la premiere valeur : ");
 a = lectureClavier.nextInt();
 // Lire et stocker la première valeur dans b
 System.out.print("Entrer la seconde valeur : ");
 b = lectureClavier.nextInt();
 // Lire et stocker le signe de l'opération dans opérateur
 System.out.print("Type de l'operation : (+, -, *, /) : ");
 opérateur = lectureClavier.next().charAt(0);
 // suivant le signe de l'opération
 switch (opérateur ) {
     // Si c'est un +, faire une addition
    case '+' : calcul = a + b;
             break;
     // Si c'est un -, faire une soustraction
    case '-' : calcul = a - b;
             break;
    // Si c'est un /, faire une division
    case '/' : calcul = a / b;
             break;
    // Si c'est une *, faire une multiplication
    case '*' : calcul = a * b ;
             break;
 }
 // Afficher le résultat
 System.out.print("Cette operation a pour resultat : ");
 System.out.println(a + " " + opérateur + " "+ b + " =" + calcul);
}
}
```

b. Exécution du programme avec le jeu de valeurs 2, 0 et /

```
Entrer la première valeur : 2
Entrer la seconde valeur : 0
Type de l'opération : (+, -, *, /) : /
java.lang.ArithmeticException: / by zero
at Calculette.main(Calculette.java:22)
```

L'interpréteur détecte une exception de type arithmétique. Il s'agit de la division par zéro.

c. L'erreur provient de la division. Il suffit de vérifier que la valeur de `b` soit non nulle pour l'étiquette `'/'` de la structure `switch`. Examinons la correction :

```
import java.util.*;
public class Calculette      {
 public static void main( String [] argument)  {
  int a, b;
  char opérateur;
  double calcul = 0;
  // Définir et initialiser un booléen à true
  boolean OK = true;
  Scanner lectureClavier = new Scanner(System.in);
  System.out.print("Entrer la premiere valeur : ");
  a = lectureClavier.nextInt();
  System.out.print("Entrer la seconde valeur : ");
  b = lectureClavier.nextInt();
  System.out.print("Type de l'operation : (+, -, *, /) : ");
  opérateur = lectureClavier.next().charAt(0);
  switch (opérateur ){
     case '+' : calcul = a + b;
                break;
     case '-' : calcul = a - b;
                break;
     // Si c'est un /, tester la valeur de b
     case '/' :  if ( b != 0) calcul = a / b;
                 // Si b vaut 0, mettre le booléen OK à false
                 else OK = false;
                 break;
     case '*' : calcul = a * b ;
                break;
     default : OK = false ;
  }
  // Si OK vaut true, afficher le résultat
  if (OK) {
     System.out.print("Cette operation a pour resultat : ");
     System.out.println(a + " " +opérateur+ " "+ b + " =" + calcul);
  }
  // Sinon OK vaut false, afficher un message d'erreur
  else System.out.println("Operation non conforme !");
 }
}
```

À l'étiquette `'/'`, le programme vérifie si le contenu de la variable `b` est non nul. Si tel est le cas, il exécute normalement l'instruction réalisant la division. Sinon, si le contenu de la variable `b` est nul,

la division n'est pas effectuée mais la valeur `false` est affectée à la variable `OK` de type booléen (initialisée par défaut à `true` lors de la déclaration de la variable).

De cette façon, pour afficher le résultat du calcul, le programme vérifie la valeur de la variable `OK`. Si elle vaut `true`, cela signifie que l'opération a été effectuée sans rencontrer de difficulté particulière sinon, cela veut dire qu'aucune opération n'a pu être réalisée. Le programme signale alors par un message que l'opération est non conforme

Remarquez que la valeur `false` est aussi affectée à la variable `OK` pour l'étiquette `default`. Ainsi, si l'utilisateur entre un autre caractère que `+`, `-`, `/` ou `*`, le programme n'exécute aucun calcul et signale par un message que l'opération est non conforme.

Dans le jargon informatique, on dit que la variable `OK` est un drapeau (en anglais *flag*). En effet, il change d'état (de valeur) en fonction des instructions exécutées.

Le terme « drapeau » fait allusion au système de fonctionnement des boîtes aux lettres américaines munies d'un drapeau rouge. Lorsque le facteur dépose du courrier, le drapeau est relevé. Le facteur abaisse le drapeau pour indiquer la présence de courrier. Lorsque le destinataire prend son courrier, il relève le drapeau, indiquant que la boîte est désormais vide. Ainsi, la position (état) du drapeau indique la présence (drapeau abaissé) ou non (drapeau levé) de courrier, dans la boîte aux lettres.

Chapitre 4 : Faire des répétitions

Exercice 4-1 : Comprendre la boucle *do…while*

Corrigé

```
import java.util.*;
public class Exercice1 {
 public static void main (String [] argument) {
      int a,b,r;
      Scanner lectureClavier = new Scanner(System.in);
      System.out.print("Entrer un entier : ");
    a = lectureClavier.nextInt();
      System.out.print("Entrer un entier : ");
      b = lectureClavier.nextInt();
      do {
        r = a%b;
        a = b;
        b = r;
      } while (r != 0 );
      System.out.println("Le resultat est " + a);
 }
}
```

a. Repérez les instructions concernées par la boucle : voir tracé **orange** sur le programme ci-dessus.
Déterminez les instructions de début et fin de boucle.: voir tracé **vert** sur le programme ci-dessus.

b. Recherchez l'instruction permettant de modifier le résultat du test de sortie de boucle : voir tracé **jaune** sur le programme ci-dessus.

c.

c.			
Instructions	a	b	r
initialisation	30	42	-

c.			
`do {`	On entre dans la boucle		
`r = a % b ;`	30	42	30
`a = b ;`	42	42	30
`b = r ;`	42	30	30
`while (r != 0)`	`r` est différent de `0`, on retourne à `do`		
`r = a % b ;`	42	30	12
`a = b ;`	30	30	12
`b = r ;`	30	12	12
`while (r != 0)`	`r` est différent de `0`, on retourne à `do`		
`r = a % b ;`	30	12	6
`a = b ;`	12	12	6
`b = r ;`	12	6	6
`while (r != 0) ;`	`r` est différent de `0`, on retourne à `do`		
`r = a % b ;`	12	6	0
`a = b ;`	6	6	0
`b = r ;`	6	0	0
`while (r != 0) ;`	`r` est égal à `0`, on sort de la boucle		

Le programme affiche : `le resultat est 6`.

d.

d.			
Instructions	`a`	`b`	`r`
`initialisation`	35	6	-
`do {`	On entre dans la boucle		
`r = a % b ;`	35	6	5
`a = b ;`	6	6	5
`b = r ;`	6	5	5
`while (r != 0)`	`r` est différent de `0`, on retourne à `do`		
`R = a % b ;`	6	5	1
`A = b ;`	5	5	1
`B = r ;`	5	1	1
`While (r != 0)`	`r` est différent de `0`, on retourne à `do`		
`R = a % b ;`	5	1	0
`A = b ;`	1	1	0
`B = r ;`	1	0	0
`While (r != 0) ;`	`r` est égal à `0`, on sort de la boucle		

Le programme affiche : `le resultat est 1`.

e. Pour comprendre ce que réalise ce programme, examinons les résultats des deux exécutions précédentes. Pour les valeurs `30` et `42`, le résultat vaut `6`. Pour les valeurs `35` et `6`, le résultat vaut `1`. Remarquons que `30` et `42` sont divisibles par `6`, alors que `35` et `6` n'ont aucun diviseur commun mis à part `1`. Nous pouvons donc dire que le résultat trouvé est le plus grand diviseur commun aux deux valeurs saisies, autrement dit le PGCD.

Exercice 4-2 Comprendre la boucle *do…while*

Corrigé

```
import java.util.*;

public class Exercice2 {
```

```
 public static void main (String [] argument) {
   int valeur;
   Scanner lectureClavier = new Scanner(System.in);
   do{
      System.out.print("Entrer un entier : ");
      valeur = lectureClavier.nextInt();
   } while ( valeur < 0 || valeur > 100) ;
   System.out.println("Vous avez saisi : " + valeur);
 }
}
```

Le programme entre dans la boucle `do … while` sans test préalable. Il demande la saisie d'une entière au clavier. Lorsque la saisie a été effectuée, le programme teste si la valeur saisie est plus petite que `0` ou plus grande que `100`. Si tel est le cas - le test est vrai, le programme retourne en début de la boucle et demande à nouveau la saisie d'une valeur. Si la valeur saisie est plus grande que `0` et plus petite que `100` - le test est faux, le programme sort de la boucle et affiche la valeur saisie.

Exercice 4-3 : Apprendre à compter, accumuler et rechercher une valeur

Corrigé

a. **Faire**

Lire une valeur entière

Mémoriser la plus grande valeur

Mémoriser la plus petite valeur

Calculer la somme des valeurs saisies

Compter le nombre de valeurs saisies

Tant que la valeur saisie est différente de 0

Afficher la plus grande et la plus petite valeur

Calculer et afficher la moyenne des valeurs

b. Le code source complet :

```
import java.util.*;
public class Exercice3{
public static void main (String [] parametre){
   int valeur, laPlusGrande, laPlusPetite, laSomme = 0, leNombre = 0 ;
   double laMoyenne;
   Scanner lectureClavier = new Scanner(System.in);
   System.out.print("Entrer une valeur :") ;
   valeur =lectureClavier.nextInt() ;
   laPlusGrande = valeur ;
   laPlusPetite = valeur ;
   do {
      if (laPlusGrande < valeur) laPlusGrande = valeur ;
      if (laPlusPetite > valeur) laPlusPetite = valeur ;
      laSomme = laSomme + valeur ;
```

```
        leNombre = leNombre + 1 ;
        System.out.print("Entrer une valeur (0 pour sortir) :") ;
        valeur =lectureClavier.nextInt() ;
    } while (valeur != 0);
    System.out.println("La plus grande valeur est : " + laPlusGrande) ;
    System.out.println("La plus petite valeur est : " + laPlusPetite) ;
    laMoyenne = (float) laSomme / leNombre ;
    System.out.println("La moyenne de ces valeurs : " + laMoyenne) ;
  } // Fin du main ()
} // Fin de la Class Maximum
```

Observez qu'une première valeur est saisie en dehors de la boucle, afin d'initialiser les deux variables `laPlusPetite` et `laPlusGrande`. Ainsi, en initialisant par exemple ces valeurs à `-1`, le programme peut donner un mauvais résultat. Imaginez par exemple que vous n'entriez que des valeurs positives. Le programme affichera en résultat comme plus petite valeur `-1`, alors que cette dernière ne fait pas partie des valeurs saisies par l'utilisateur. Grâce à l'initialisation des variables à la première valeur saisie, nous sommes surs d'obtenir un résultat cohérent.

Pour finir, remarquez le cast (`float`) devant le calcul de la moyenne. En effet, les deux variables `laSomme` et `leNombre` sont de type entier. Sans ce cast, la division fournit un résultat entier.

Exercice 4-4 : Comprendre la boucle `while`, traduire une marche à suivre en programme Java

Corrigé

Traduction de la marche à suivre en Java

a. Tirer au hasard un nombre entre 0 et 10.
   `i = (int) (10*Math.random()) ;`

b. Lire un nombre.
   `nombreLu = lectureClavier.nextInt();`

c. Tant que le nombre lu est différent du nombre tiré au hasard :
   `while ( nombreLu != i)`
   - Lire un nombre
     `nombreLu = lectureClavier.nextInt();`
   - Compter le nombre de boucle.
     `nbBoucle = nbBoucle + 1` (ou encore `nbBoucle++`)

d. Afficher un message de réussite ainsi que le nombre de boucles.
   ```
   System.out.print("Bravo ! ");
   System.out.println("vous avez reussi en " + nbBoucle + " fois");
   ```

Exercice 4-5 : Comprendre la boucle `while`, traduire une marche à suivre en programme Java

Corrigé

Le code source complet :

```
import java.util.*;
public class Devinette {
 public static void main (String [] parametre) {
      int i, nombreLu = -1, nbBoucle = 0;
      i = (int) (10*Math.random());
      Scanner lectureClavier = new Scanner(System.in);
```

```
      System.out.print("Ceci est un jeu, j'ai tire un nombre au ");
      System.out.println("hasard entre 0 et 10, devinez lequel ? ");
      while (nombreLu != i){
         System.out.print("Votre choix : ");
         nombreLu =lectureClavier.nextInt();
         nbBoucle = nbBoucle + 1;
      }
      System.out.print("Bravo ! ");
      System.out.println("vous avez reussi en " + nbBoucle + " fois");
 } // Fin du main ()
} // Fin de la Class Devinette
```

Remarquez l'initialisation de la variable `nombreLu` à `-1`. En effet, pour être sûr d'entrer dans la boucle `while`, la variable `nombreLu` doit contenir une valeur différente de `i`. Or celle-ci par définition, varie entre `0` et `10`. L'initialisation à `-1` permet donc de certifier que le programme entrera au moins une fois dans la boucle.

Exercice4-6 : Comprendre la boucle `while`, traduire une marche à suivre en programme Java

Corrigé

Quelques améliorations :

```
import java.util.*;
public class Jeu{
public static void main (String [] parametre)      {
      int i, nombreLu = -1, nbBoucle = 0;
      Scanner lectureClavier = new Scanner(System.in);
      // a. Les valeurs tirées au hasard soit comprises entre 0 et 50.
      i = (int) (50*Math.random());
      System.out.print("Ceci est un jeu, j'ai tire un nombre au ");
      System.out.println("hasard entre 0 et 50, devinez lequel ? ");
      while (nombreLu!= i){
         System.out.print("Votre choix : ");
         nombreLu =lectureClavier.nextInt();
         nbBoucle++;
         // b. Un message d'erreur doit afficher si la réponse est mauvaise.
         if (nombreLu != i) System.out.println("Mauvaise reponse");
         // c. Indique si la valeur saisie est plus grande
         //ou plus petite que la valeur tirée au hasard.
         if (nombreLu < i) System.out.println(" Trop petit !");
         if (nombreLu > i) System.out.println(" Trop grand !");
      }
      System.out.print("Bravo ! ");
      System.out.println("vous avez reussi en " + nbBoucle + " fois");
} // Fin du main ()
} // Fin de la Class Jeu
```

d. **À titre de réflexion :** comment faut-il s'y prendre pour trouver la valeur en donnant le moins de réponses possibles ?

Les valeurs tirées au hasard sont comprises entre 0 et 50. le programme indique si la valeur lue au clavier est plus grande ou plus petite que celle tirée au hasard. La meilleure méthode pour trouver le plus rapidement la réponse est de choisir toujours une valeur de milieu, par rapport à un ensemble de valeurs (essai par dichotomie).

Exemple : nous supposons que l'ordinateur a choisi 8

Notre ensemble de valeurs est initialement `[0, 50]`, choisissons une valeur moyenne dans cet intervalle, soit `25`. L'ordinateur répond : `trop grand !` `(nbBoucle = 1)`

Si 25 est une valeur trop grande, notre ensemble de valeurs se restreint à `[0, 25[`, choisissons une valeur moyenne dans cet intervalle, soit `12`. L'ordinateur répond : `trop grand !` `(nbBoucle = 2)`

Si 12 est une valeur trop grande, notre ensemble de valeurs se restreint à `[0, 12[`, choisissons une valeur moyenne dans cet intervalle, soit `6`. L'ordinateur répond : `trop petit !` `(nbBoucle = 3)`

Si 6 est une valeur trop petite, notre ensemble de valeurs se restreint à `]6, 12[`, choisissons une valeur moyenne dans cet intervalle, soit `9`. L'ordinateur répond : `trop grand !` `(nbBoucle = 4)`

Si 9 est une valeur trop grande, notre ensemble de valeurs se restreint à `]6, 9[`, choisissons une valeur moyenne dans cet intervalle, soit `8`. L'ordinateur répond : `Bravo ! vous avez reussi en 5 fois.` `(nbBoucle = 5)`

Exercice 4-7 : Comprendre la boucle `for`

Corrigé :

```
import java.util.*;
public class Exercice7 {
 public static void main (String [] paramètre) {
      long i, b = 1;
      int a;
      Scanner lectureClavier = new Scanner(System.in);
      System.out.print("Entrer un entier :");
      a = lectureClavier.nextInt();
      for (i = 2; i <= a; i++)
        b = b * i;
      System.out.println("La resultat vaut " + b);
 }
}
```

Corrigé

a. Repérez les instructions concernées par la boucle : voir tracé **orange** sur le programme ci-dessus.
Déterminez les instructions de début et fin de boucle : voir tracé **vert** sur le programme ci-dessus.

b. La variable `i` est initialisée à `2`. Elle vaut `a + 1` en sortant de la boucle. Le nombre de boucles est calculé par la formule : (valeur en sortie de boucle – valeur initiale) soit, `a + 1 - 2 = a - 1` tours de boucle.

c. Recherchez l'instruction permettant de modifier le résultat du test de sortie de boucle : voir tracé **jaune** sur le programme ci-dessus.

d.

d.			
Instructions	i	b	a
Initialisation	-	1	6
for(…) {	2	On entre dans la boucle car, i <= a	
b = b * i ;	2	2	6
for(…) {	3	On reste dans la boucle car, i <= a	
b = b * i ;	3	6	6
for(…) {	4	On reste dans la boucle car, i <= a	
b = b * i ;	4	24	6
for(…) {	5	On reste dans la boucle car, i <= a	
b = b * i ;	5	120	6
for(…) {	6	On reste dans la boucle car, i <= a	
b = b * i ;	6	720	6
for(…) {	7	On sort de la boucle car, i > a	

e. Ce programme calcul la factorielle d'un nombre soit, `n! = 1 * 2 * 3 * … * n`.

Exercice 4-8 : Comprendre la boucle `for`

Corrigé

```
public class Exercice9{
public static void main (String [] parametre){
      char c;
      for (c = 'a'; c <= 'z'; c++) System.out.print(c + " ");
      System.out.println();
      for (c = 'z'; c >= 'a'; c--) System.out.print(c + " ");
      System.out.println();
 }
}
```

Les caractères correspondent en réalité à des valeurs numériques (Unicode). Il est donc possible de les utiliser comme variable de contrôle d'une boucle `for`.

Partie 2 : Initiation à la programmation objet

Chapitre 5 : De l'algorithme paramétré à l'écriture de fonctions

Exercice 5-1 :Apprendre à déterminer les paramètres d'un algorithme

Corrigé

a. Afficher "Combien de sucre ?" et, saisir le nombre souhaité de morceaux de sucre au clavier, mettre la valeur dans `nombreSucre`.

b. Tant que le nombre de morceaux de sucre mis dans la boisson chaude ne correspond pas à `nombreSucre`, ajouter un sucre.

c. Le paramètre qui permet de sucrer plus ou moins la boisson, est la variable correspondant au nombre maximum de sucres à mettre dans la boisson. Soit, `nombreSucre`.

d. L'algorithme a pour nom `sucrer`, son paramètre a pour nom : **`nombre`**. L'entête de l'algorithme s'écrit : `sucrer(`**`nombre`**`)`

L'algorithme s'écrit : Tant que le nombre de sucres mis dans la boisson chaude ne correspond pas à **nombre**, ajouter un sucre. Le nom du paramètre a remplacé la variable `nombreSucre`.

e. Pour appeler l'algorithme afin qu'il sucre avec le bon nombre de morceaux de sucre : `sucrer(nombreSucre)`. Ainsi, l'algorithme est exécuté en remplaçant **nombre** par **nombreSucre**.

Exercice 5-2 :Comprendre l'utilisation des fonctions

Corrigé

```
public class Fonction{
public static void main(String [] parametre){
   // Déclaration des variables
   int a,compteur;
   for (compteur = 0; compteur <= 5; compteur++){
      a = calculer(compteur);
      System.out.print(a + " a ");
   }
} // fin de main()
public static int calculer(int x){
   int y;
   y = x * x;
   return y ;
} // fin de fonc1()
} //fin de class
```

a. Le bloc définissant la fonction `main()` : voir tracé **orange** sur le programme ci-dessus.
Le bloc définissant la fonction `calculer()` voir tracé **vert** sur le programme ci-dessus.
Le bloc définissant la classe `Fonction`, voir tracé **jaune** sur le programme ci-dessus.

b. `x` est le paramètre formel de la fonction `calculer()`

c. Les valeurs transmises au paramètre `x` de la fonction `calculer()`, lors de son appel depuis la fonction `main()` sont celles placées dans la variable `compteur` soit `0`, `1`, `2`, `3`, `4` et `5`.

d. Le produit de `x` par `x`, soit `0`, `1`, `4`, `9`, `16` et `25`.

e. Les valeurs transmises à la variable `a` sont les résultats de la fonction `calculer()`.

f. `0 a 1 a 4 a 9 a 16 a 25 a`

Exercice 5-3 :Comprendre l'utilisation des fonctions

Corrigé

a. Ecrire la fonction `main()` qui affiche le résultat de la fonction `f(0)`.

```
public static void main(String [] parametre) {
   int R ;
   R = f(0) ;
   System.out.print(R);
}
```

b. Calculer `f(x)` pour `x` variant entre `−5` et `5`

```
public static void main(String [] parametre) {
```

```
    int R, max = f(0);
    for (int x = -5; x <= 5; x++) {
       R = f(x) ;
       System.out.print(R) ;
    }
}
```

c. Déterminer le maximum de la fonction `f(x)` entre `-5` et `5`,

```
public static void main(String [] parametre) {
    int R, max = f(0);
    for (int x = -5; x <= 5; x++) {
       R = f(x) ;
       if (R > max) max = R ;
    }
    System.out.print("Le max est : " + max);
 }
```

Exercice 5-4 : Détecter des erreurs de compilation concernant les paramètres ou le résultat d'une fonction

a.

```
public static void menu (int c) {
    switch (c) {...
    }
    return c;
}
```

Corrigé

a. La fonction `max()` est définie dans ce chapitre, avec deux paramètres entiers alors qu'ici les deux paramètres utilisés sont de type `double`.

b. L'entête de la fonction précise que le résultat retourné par la fonction `max()` est de type `int` alors que, la variable `m` effectivement retournée par l'instruction `return` est déclarée de type `float`.

c. La fonction `menu()` décrite au cours de ce chapitre, est de type `void`. L'instruction **`v1 =`** `menu(v2)` n'est pas valide, à cause de la présence de l'affectation `v1 = ....`

d. L'entête de la fonction `menu()` précise qu'elle est de type `void`. Le corps de la fonction ne peut donc pas posséder d'instruction `return`.

Exercice 5-5 : Ecrire une fonction simple

Corrigé

a. Les instructions composant la fonctions sont :

```
    double prct ;
    prct = (double) nb / t * 100;
```

b. En supposant que le nom de la fonction soit `pourcentage()`, l'entête de la fonction s'écrit :

```
public static ..... pourcentage(.....)  {
    double prct = (double) nb / t * 100;
}
```

c. Les deux valeurs pouvant modifier le résultat sont `t` et `nb`. Les paramètres de la fonction s'écrivent :

```
public static … pourcentage(int t, int nb)
```

d. Le résultat étant stocké dans la variable `prct`, de type `double`, la méthode est doit être de type `double`. L'entête de la fonction s'écrit donc :

```
public static double pourcentage(int t, int nb)
```

La fonction `pourcentage()` s'écrit :

```
public static double pourcentage(int t, int nb){
   double prct = (double) nb / t * 100;
   return prct ;
}
```

e. Ecrire la fonction `main()` qui fait appel à la fonction `pourcentage()`

```
import java.util.*;
public class Exercice5 {
 public static void main (String [] arg){
   int nbCB, nbCheque, nbVirement, nbDebit;
   double résultat;
   Scanner lectureClavier = new Scanner(System.in);
   System.out.print(" Nombre d'achat Cartes Bleues : ");
   nbCB = lectureClavier.nextInt();
   System.out.print(" Nombre de cheques emis : ");
   nbCheque = lectureClavier.nextInt();
   System.out.print(" Nombre de virements automatiques : ");
   nbVirement = lectureClavier.nextInt();
   nbDebit = nbCB + nbCheque + nbVirement;
   System.out.println("Vous avez emis " + nbDebit + " debits ");
   résultat = pourcentage(nbDebit, nbCB) ;
   System.out.println(" dont " + résultat + " % par Carte bleue ");
   résultat = pourcentage(nbDebit, nbCheque) ;
   System.out.println("" + résultat + " % par Cheques ");
   résultat = pourcentage(nbDebit, nbVirement) ;
   System.out.println("" + résultat + " % par Virement automatique ");
 }
 public static double pourcentage(int t, int nb)  {
   double prct = (double) nb / t * 100;
   return prct ;
 }
}
```

Exercice 5-6 : Ecrire une fonction simple

Corrigé

```
import java.util.*;
public class Exercice6 {
 // 5.6.a La fonction verifier() retourne un entier
```

```
  public static int verifier() {
    int resultat;
    Scanner lectureClavier = new Scanner(System.in);
    do{
       System.out.print("Entrer un entier : ");
       resultat= lectureClavier.nextInt();
    } while ( resultat < 0 || resultat > 100) ;
    // Lorsque la valeur saisie est > 0 ou < à 100
    // sortir de la fonction en retournant la valeur saisie
    return resultat ;
  }
// 5.6.d La fonction verifierAvecBornes() retourne un entier
// Elle possède deux paramètres de type entier qui correspondent
// aux bornes de saisie de la valeur.
    public static int verifierAvecBornes(int a, int b) {
    int resultat;
    Scanner lectureClavier = new Scanner(System.in);
    do{
       System.out.print("Entrer un entier : ");
       resultat= lectureClavier.nextInt();
    } while ( resultat < a || resultat > b) ;
    // Lorsque la valeur saisie est > a ou < à b
    // sortir de la fonction en retournant la valeur saisie
    return resultat ;
  }
// La fonction main()
  public static void main(String [] parametre) {
    int valeur;
    valeur = verifier();
    System.out.println("valeur : " + valeur);
    // 5.6.d Appel de fonction verifierAvecBornes()
    // La valeur à saisir doit être comprise entre 10 et 20
    valeur = verifierAvecBornes(10, 20);
    System.out.println("valeur : " + valeur);
  }
}
```

b. Que réalise l'application si l'utilisateur saisi les valeurs `-10`, `123` et `22` ?

Si l'utilisateur saisit `-10`, le programme retourne en début de boucle et ne peut sortir de la fonction. De même pour `123`. Lorsqu'il saisit `22`, le programme sort de la boucle et retourne la valeur à la fonction appelante, c'est-à-dire la fonction `main()` qui affiche à son tour la valeur saisie.

c. Pour que la valeur saisie soit comprise non plus entre `0` et `100` mais entre deux valeurs `a` et `b` choisies par l'utilisateur, il suffit de remplacer à l'intérieur de la fonction `vérifier()` les valeurs

numériques `0` et `100` par des variables. Ces variables sont alors définies comme paramètres de la fonction. De cette façon, le choix des bornes s'effectue au moment de l'appel de la fonction.

d. Voir la zonedans le code source ci-avant.

Chapitre 6 : Fonctions, notions avancées

Exercice 6-1 : Repérer les variables locales et variables de classe

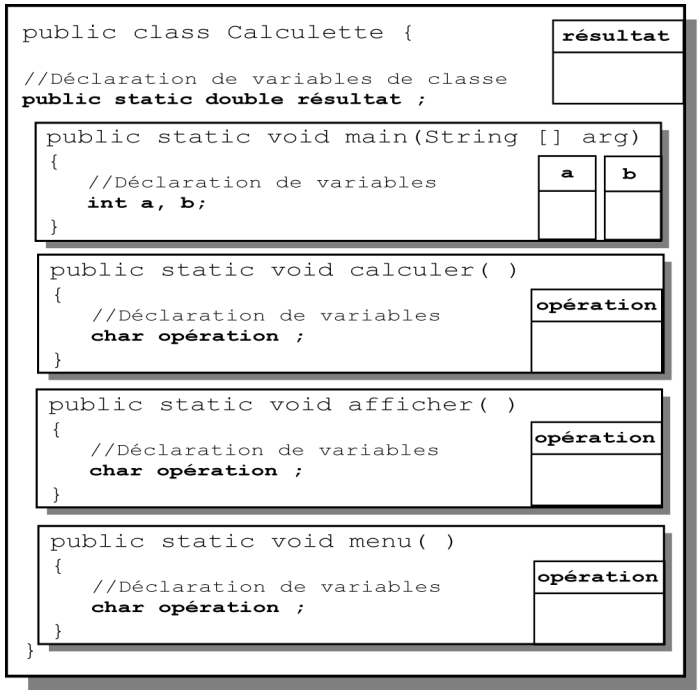
Corrigé :

```
import java.util.*;
public class Calculette {
 public static double résultat ;
  public static void main( String [] argument){
   int a, b;
   Scanner lectureClavier = new Scanner(System.in);
   menu();
   System.out.println("Entrer la premiere valeur ");
   a = lectureClavier.nextInt();
   System.out.println("Entrer la seconde valeur ");
   b = lectureClavier.nextInt();
   calculer();
   afficher();
  }

  public static void calculer() {
   char opération ;
   switch (opération) {
      case '+' : résultat = a + b ;
      break ;
      case '-' : résultat = a - b ;
      break ;
      case '/' : résultat = a / b ;
      break ;
      case '*' : résultat = a * b ;
   break ;
   }
 }

  public static void afficher() {
   char opération ;
   System.out.print(a + "" +opération + ""+ b + " =" + résultat);
  }

  public static void menu() {
```

```
    char opération ;
    Scanner lectureClavier = new Scanner(System.in);
    System.out.println("Je sais compter, entrez l'operation choisie") ;
    System.out.println(" + pour additionner ") ;
    System.out.println(" - pour soustraire ") ;
    System.out.println(" * pour pultiplier ") ;
    System.out.println(" / pour diviser ") ;
    System.out.println(" (+, -, *, /)? : ") ;
    opération = lectureClavier.next().charAt(0) ;
  }
}
```

a. Les fonctions de la classe `Calculette` sont au nombre de quatre et ont pour nom : `main()`, `afficher()`, `calculer()` et `menu()` (voir tracé **orange** sur le programme ci-dessus).

b. Lorsque les variables `a`, `b` et `opération` sont déclarées à l'intérieur des fonctions, elles ne sont pas visibles en dehors de la fonction où elles sont déclarées, comme le montre le schéma suivant :



c. Les variables locales à la fonction `main()` sont : `a` et `b` (voir tracé **vert** sur le programme ci-dessus).

Les variables locales à la fonction `afficher()` sont : `opération` (voir tracé **vert** sur le programme ci-dessus).

Les variables locales à la fonction `calculer()` sont : `opération` (voir tracé **vert** sur le programme ci-dessus).

Les variables locales à la fonction `menu()` sont : `opération` (voir tracé **vert** sur le programme ci-dessus).

d. La fonction `calculer()` ne peut réaliser l'opération demandée puisque la variable `opération` est déclarée dans la fonction `menu()`. Le caractère correspondant à l'opération est stocké dans la case mémoire `opération` localement à la fonction `menu()`. Dans la case mémoire `opération`

de la fonction `calculer()`, il n'y a par conséquent, aucun caractère. En effet, ce n'est pas parce que deux variables portent le même nom qu'elles représentent la même case mémoire.

e. De la même façon, la fonction `afficher()` ne peut réaliser l'opération demandée puisque `a` et `b` sont des variables déclarées localement à la fonction `main()`.

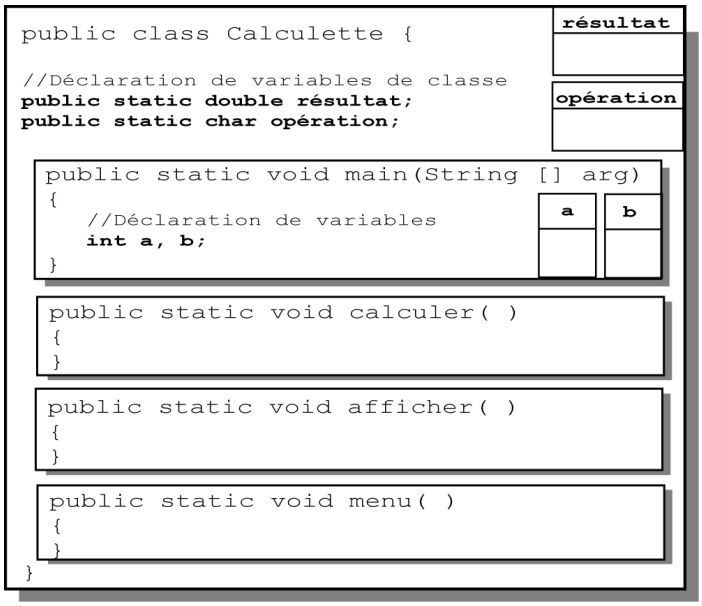
En réalité, ce programme ne peut être exécuté puisque la phase de compilation détecte les erreurs suivantes :

```
Calculette.java:16: Variable opération may not have been initialized.
Calculette.java:17: Undefined variable: a
Calculette.java:17: Undefined variable: b
Calculette.java:19: Undefined variable: a
Calculette.java:19: Undefined variable: b
Calculette.java:21: Undefined variable: a
Calculette.java:21: Undefined variable: b
Calculette.java:23: Undefined variable: a
Calculette.java:23: Undefined variable: b
Calculette.java:30: Undefined variable: a
Calculette.java:30: Variable opération may not have been initialized.
Calculette.java:30: Undefined variable: b
```

Exercice 6-2 : Communiquer des valeurs à l'appel d'une fonction

Corrigé :

a. Lorsque les variables `résultat` et `opération` sont déclarées comme variable de classe, elles sont visibles et accessibles depuis toutes les fonctions de la classe `Calculette`, comme le montre le schéma suivant :



b. Pour que les fonctions `calculer()` et `afficher()` connaissent le contenu des variables `a` et `b`, il suffit de passer les valeurs contenues dans ces variables en paramètre des fonctions.

c. Les fonctions s'écrivent :

```
public static void calculer(int x, int y) {
```

```
    switch (opération){
       case '+' : résultat = x + y;
       break;
       case '-' : résultat = x - y;
       break;
       case '/' : résultat = x /y;
       break;
       case '*' : résultat = x * y ;
       break;
    }
  }
  public static void afficher(int x, int y){
    System.out.println(x + "" +opération+ " "+ y + " =" + résultat);
  }
```

Pour éviter toute confusion, les paramètres formels(`x` et `y`) des fonctions ne portent pas les mêmes noms que les paramètres réels (`a` et `b`). Les instructions composant les fonctions ont donc été modifiées de façon à ne plus utiliser les variables `a` et `b` mais, `x` et `y`.

Exercice 6-3 : Transmettre un résultat à la fonction appelante

Corrigé :

a. Les variables `résultat` et `opération` étant déclarées localement aux fonctions, il ne peut y avoir transmission des valeurs (*voir correction Exercice 6-1.d*) entre les fonctions.

b. La fonction `menu()` doit transmettre l'opérateur choisi par l'utilisateur à la fonction `main()` en utilisant l'instruction `return`. Puis l'opérateur est transmis à la fonction `calculer()`, à l'aide d'un paramètre.

c. et d. voir tracé **orange** sur le programme ci-dessous.

e. La fonction `calculer()` doit transmettre le résultat de l'opération à la fonction `main()` en utilisant l'instruction `return`. Puis le résultat est transmis à la fonction `afficher()`, à l'aide d'un paramètre.

f. et g. voir tracé **vert** sur le programme ci-dessous.

```
import java.util.*;
public class Exercice3 {
 public static void main( String [] argument){
   int a, b;
   char opérateur;
   double calcul;
   Scanner lectureClavier = new Scanner(System.in);
   opérateur = menu();
   System.out.println("Entrer la premiere valeur ");
   a = lectureClavier.nextInt();
   System.out.println("Entrer la seconde valeur ");
   b = lectureClavier.nextInt();
   calcul = calculer(a, b, opérateur );
   afficher(a, b, opérateur, calcul);
```

```
}
public static double calculer (int x, int y, char o) {
  double résultat =0;
  switch (o){
     case '+' : résultat = x + y;
     break;
     case '-' : résultat = x - y;
     break;
      case '/' : résultat = x / y;
     break;
     case '*' : résultat = x * y ;
     break;
  }
  return résultat;
}

public static void afficher(int x, int y, char o, double r) {
  System.out.println(x + "" +o+ " "+ y + " =" + r);
}

public static char menu(){
  char opération ;
  Scanner lectureClavier = new Scanner(System.in);
  System.out.println("Je sais compter, entrer en premier l'operation choisie ");
  System.out.println("+ pour additionner ");
  System.out.println(" - pour soustraire ");
  System.out.println(" * pour pultiplier ");
  System.out.println(" / pour diviser ");
  System.out.println(" (+, -, *, /)? : ");
  opération = lectureClavier.next().charAt(0);
  return opération ;
}
}
```

Chapitre 7 : Les classes et les objets

Exercice 7-1 : Utiliser les objets de la classe String

Corrigé :

```
import java.util.*;
public class Exercice1 {
   public static void main(String [] argument) {
      String s1 = "", s2 = "", s3 = "", s4 = "";
      int nbA = 0;
```

```
        Scanner lectureClavier = new Scanner(System.in);
        // a. demande la saisie d'une phrase
        System.out.print("Entrez une phrase : ");
        s1 = lectureClavier.next();
        // b. affiche la phrase en majuscule
        s2 = s1.toUpperCase();
        // c. compte le nombre de 'a'
        for (int i = 0; i < s2.length(); i++)
           if(s2.charAt(i) == 'A') nbA++;
        System.out.println("Vous avez entre : " + s1);
        System.out.println("Soit en majuscule : " + s2);
        System.out.println("Ce mot contient : " + nbA + " A ");
        // c. transforme tous les 'a' par des '*'
        s3 = s2.replace('A','*');
        System.out.println("Il s'ecrit donc : " + s3);
        System.out.print("Entrez un second mot : ");
        s4 = lectureClavier.next();
        // d. teste si s4 se trouve entre les 5ième et 12ième caractères de s1
        if (s1.regionMatches(5,s4,0,7))
        System.out.println("La sous chaine " + s4 + " est bien placee ");
    }
}
```

Exercice 7-2 : Utiliser les objets de la classe String

Corrigé :

Le programme reprend la marche à suivre de l'exercice 2 du chapitre 4, qui recherche la plus grande et la plus petite valeur d'une liste de nombres saisis au clavier, la saisie s'arrêtant lorsque l'utilisateur entre la valeur `0`. Pour cet exercice, la démarche est identique. Seules, les techniques de comparaison diffèrent puisque les variables utilisées ne sont plus numériques mais, alphabétiques.

```
import java.util.*;
public class Exercice2 {
   public static void main(String [] argument) {
      String s1 = "", sPlusGrand = "", sPlusPetit = "";
      Scanner lectureClavier = new Scanner(System.in);
      System.out.print("Entrez un mot : ");
      s1 = lectureClavier.next();
      sPlusGrand = s1 ;
      sPlusPetit = s1 ;
      do {
         if (s1.compareTo(sPlusGrand) < 0) sPlusGrand = s1 ;
         if (s1.compareTo(sPlusPetit) > 0) sPlusPetit = s1 ;
         System.out.print("Entrer une mot (FIN pour sortir) : ") ;
         s1 = lectureClavier.next();
```

```
        } while ( ! s1.equalsIgnoreCase("FIN") );
        System.out.println("Le plus grand mot : " + sPlusGrand) ;
        System.out.println("Le plus petit mot : " + sPlusPetit) ;
    }
}
```

Remarquez le `'!'` devant l'expression `s1.equalsIgnoreCase("FIN")`. Le `'!'` est utilisé pour nier une expression située juste après. Littéralement, l'expression se traduit par « tant que `s1` n'est pas égal à `"FIN"` sans tenir compte des majuscules ».

Exercice 7-3 et 7.4: Créer une classe d'objets et consulter les variables d'instance

Corrigé :

```
public class Exercice3  {
  public static void main (String [] parametre)  {
   byte nbjours = 0 ;
   String mois="";
   short annee ;
   Scanner lectureClavier = new Scanner(System.in);
   System.out.println("De quel mois s'agit-il ? :") ;
   mois =  lectureClavier.nextLine();
   // remplacer les û éventuels par u
   mois  = mois.replace("û", "u");
   // Remplacer tous les é éventuels par e
   mois  = mois.replace("é", "e");
   // Supprimer les éventuelles majuscules
   mois = mois.toLowerCase();
   System.out.println("De quelle annee ? :") ;
   annee =  lectureClavier.nextShort();
   switch(mois)   {
     case "janvier" : case "mars" :    // Pour les mois à 31 jours
     case "mai" : case "juillet" :
     case "aout"  : case "octobre" :
     case "decembre"  :
            nbjours = 31 ;
            break ;
     case "avril" : case "juin" :      // Pour les mois à 30 jours
     case "septembre" : case "novembre" :
            nbjours  = 30 ;
            break ;
     case "fevrier" :
          // Pour le cas particulier du mois de fevrier
          if (annee % 4 == 0 && annee % 100 != 0 || annee % 400 == 0)
               nbjours  = 29 ;
          else  nbjours  = 28 ;
```

```
            break ;
        default :                          // En cas d'erreur de frappe
                System.out.println("Impossible, ce mois n'existe pas ") ;
                System.exit(0) ;
    }
    System.out.print(" En " + annee + ", le mois de " + mois) ;
    System.out.println(" a " + nbjours + " jours ") ;
   }
}
```

Exercice 7-4 et 7.5: Créer une classe d'objets et consulter les variables d'instance

Corrigé :

La classe `Livre` :

```
import java.util.*;
public class Livre {
   // 7.3.a Définition des propriétés
   public String titre;
   public String categorie ;
   public String isbn ;
   public String nomAuteur ;
   public String prenomAuteur ;
   public String code ;

   // 7.4.a Définition des comportements : La méthode afficherUnLivre()
   public void afficherUnLivre(){
      System.out.println("Titre : " + titre);
      System.out.println("Auteur : " + nomAuteur + " " + prenomAuteur);
      System.out.println("Categorie : " + categorie);
      System.out.println("ISBN : " + isbn);
   }
   // 7.4.c Définition des comportements : La méthode calculerLeCode()
   public String calculerLeCode () {
      String debutNom;
      String debutPrenom;
      String debutCategorie;
      int longueurIsbn;
      String finIsbn;
      // 7.4.c Récupérer les deux premières lettres du nom
      debutNom=nomAuteur.substring(0,2);
      // 7.4.c Récupérer les deux premières lettres du prénom
      debutPrenom=prenomAuteur.substring(0,2);
      // 7.4.c Récupérer les deux premières lettres de la catégorie
```

```
        debutCategorie=categorie.substring(0,2);
        // 7.4.c Calculer la longueur du mot ISBN
        longueurIsbn=isbn.length();
        // 7.4.c Récupérer les deux dernières lettres du numéro ISBN
        finIsbn=isbn.substring((longueurIsbn-2),longueurIsbn);
        // 7.4.c Retourner la suite des caractères extraits ci-avant
        return debutNom+debutPrenom+debutCategorie+finIsbn;
    }
}
```

La classe `Bibliotheque` :

```
import java.util.*;
public class Bibliotheque {
   public static void main(String [] arg){
      Scanner lectureClavier = new Scanner(System.in);
      // 7.3.b Définition de l'objet livrePoche
      Livre livrePoche = new Livre();
      // 7.3.b Saisie du titre, pour l'objet livrePoche
      System.out.print("Entrez le titre : ");
      livrePoche.titre= lectureClavier.next();
      // 7.3.b Saisie de la catégorie pour l'objet livrePoche
      System.out.print("Entrez la categorie : ");
      livrePoche.categorie = lectureClavier.next();
      // 7.3.b Saisie du nom de l'auteur pour l'objet livrePoche
      System.out.print("Entrez le nom de l'auteur : ");
      livrePoche.nomAuteur= lectureClavier.next();
      // 7.3.b Saisie du prénom de l'auteur pour l'objet livrePoche
      System.out.print("Entrez le prenom de l'auteur : ");
      livrePoche.prenomAuteur= lectureClavier.next();
      // 7.3.b Saisie du numéro ISBN pour l'objet livrePoche
      System.out.print("Entrez le numero ISBN : ");
      livrePoche.isbn= lectureClavier.next();
      // 7.4.b Affichage des caractéritiques de l'objet livrePoche
      livrePoche.afficherUnLivre();
      // 7.4.d Affichage du code de l'objet livrePoche
      System.out.println("code du livre : " + livrePoche.calculerLeCode());
   }
}
```

Exercice 7-6 : Analyser les résultats d'une application "objet"

Corrigé

a. Le programme correspondant à l'application est celui qui contient la fonction `main()`. Dans cet exercice, le programme s'appelle `FaireDesTriangles`.

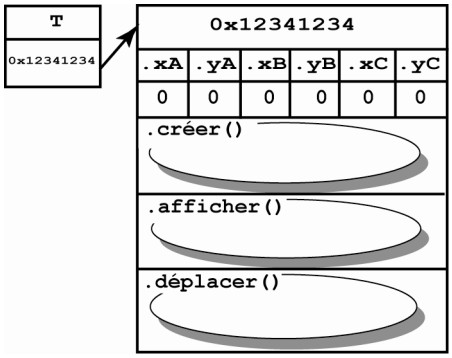
b. Le type `Triangle` est défini par la classe `Triangle`, décrite dans le fichier `Triangle.java`.

c. Les attributs de la classe `Triangle` sont les coordonnées définissant les trois sommets d'un triangle. Ils ont pour nom respectivement `xA`, `yA`, `xB`, `yB`, `xC` et `yC` déclarés comme variables d'instance de type `int`.

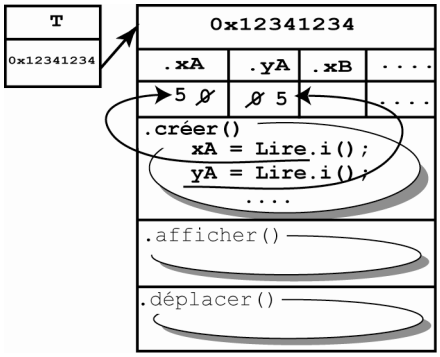
d. Trois méthodes sont définies dans la classe `Triangle`. Elles ont pour nom : `créer()`, `afficher()` et `déplacer()`.

e. L'application utilise un objet portant le nom `P`. Les données `x` et `y` de l'objet `P` valent toutes les deux `0`, juste après l'instruction de déclaration de l'objet `P`.

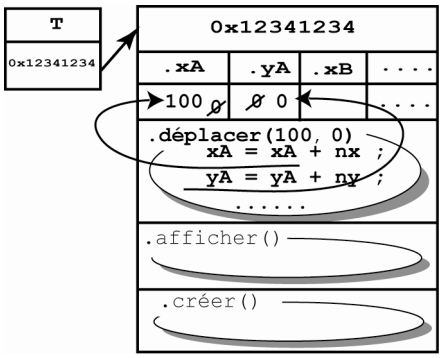
f. L'objet `T` est représenté de la façon suivante :



g. La méthode `créer()` est appelée par l'intermédiaire de l'objet `T`. Ce sont donc les données `xA`, `yA`, `xB`, `yB`, `xC` et `yC` de l'objet `T` qui mémorisent les valeurs saisies au clavier.



h. De la même façon, la méthode `déplacer()` est appelée par l'intermédiaire de l'objet `T`. Ce sont donc les données `xA`, `yA`, `xB`, `yB`, `xC` et `yC` de l'objet `T` qui prennent les valeurs transmises en paramètre de la méthode.



i. Le résultat final est le suivant :

```
Point A : 0 0
Point B : 0 0
```

```
Point C : 0 0
Point A :
Entrez l'abscisse : 10
Entrez l'ordonnée : 10
Point B :
Entrez l'abscisse : 20
Entrez l'ordonnée : 40
Point C :
Entrez l'abscisse : 40
Entrez l'ordonnée : 10
Point A : 10 10
Point B : 20 40
Point C : 40 10
Point A : 110 10
Point B : 120 40
Point C : 140 10
```

Exercice 7-7 et 7.8 : La classe Rectangle et l'application FaireDesRectangles.java

Corrigé

La classe `Rectangle` :

```
import java.util.*;
public class Rectangle{
   // 7.6.a Définition des propriétés d'un rectangle
   public int x, y, couleur, hauteur, largeur ;
   // 7.6.b Définition de la méthode créer()
   public void créer() {
      System.out.println("Position en X ");
      // 7.7.b Chaque propriété du rectangle est vérifiée
      x = verifier(0, 800);
      System.out.println("Position en Y ");
      y = verifier(0, 600);
      System.out.println("Couleur ");
      couleur = verifier(0, 10);
      System.out.println("Hauteur ");
      hauteur = verifier(0, 600);
      System.out.println("Largeur ");
      largeur = verifier(0, 800);
   }
    // 7.7.a Définition de la méthode créer()
   publicint verifier(int a, int b) {
      int resultat;
```

```
        Scanner lectureClavier = new Scanner(System.in);
        do{
           System.out.print("Entrez une valeur : ");
           resultat= lectureClavier.nextInt();
        } while ( resultat < a || resultat > b) ;
        return resultat ;
     }
     // 7.6.b Définition de la méthode afficher()
     public void afficher(){
        System.out.println("Position en X : " + x);
        System.out.println("Position en Y : " + y);
        System.out.println("Couleur : " + couleur);
        System.out.println("Hauteur: " + hauteur);
        System.out.println("Largeur: " + largeur);
      }
     // 7.6.b Définition de la méthode déplacer()
     public void deplacer(int nx, int ny){
        x += nx;
        y += ny;
     }
  } // Fin de la classe Rectangle
```

La classe `FaireDesRectangles` :

```
  public class FaireDesRectangles {
     public static void main(String[] args) {
        Rectangle R= new Rectangle();
        R.afficher();
  // 7.6.c Les valeurs 200, 200, 150 et 100 sont saisies au clavier
  // 7.7.c Les valeurs -10, -5, 20 et 900 ne correspondent pas aux intervalles de
  // valeurs demandées, il n'est pas possible de créer un tel rectangle
        R.créer();
        R.afficher();
  // 7.6.c Le rectangle se déplace de 200 sur l'axe des y
        R.deplacer(0, 200);
        R.afficher();
     }
  }
```

Chapitre 8 : Les principes du concept objet

Exercice 8-1 à 8.5 : La protection des données

Corrigé:

La classe `Livre` :

```
  import java.util.*;
```

```
public class Livre {
   //8.1.a Les propriétés sont définies en mode privé
   private String titre;
   private String categorie ;
   private String isbn ;
   private String nomAuteur ;
   private String prenomAuteur ;
   private String code ;
// 8.2.a Définition des méthodes d'accès en écriture
   // Le Titre
   public void setTitre (String valeur) {
      titre = valeur ;
   }
   // 8.2.a La catégorie
   public void setCategorie (String valeur) {
      categorie = valeur;
   }
   // 8.2.a Le numéro ISBN
   public void setIsbn (String valeur) {
      isbn =valeur;
   }
   // 8.2.a Le nom de l'auteur
   public void setNomAuteur (String valeur) {
      nomAuteur =valeur;
   }
   // 8.2.a Le prénom de l'auteur
   public void setPrenomAuteur (String valeur) {
      prenomAuteur =valeur;
   }
   // 8.3.a Définition des méthodes d'accès en lecture
   // Le titre
   public String getTitre () {
      return titre;
   }
   // 8.3.a La catégorie
   public String getCategorie () {
      return categorie;
   }
   // 8.3.a Le numéro ISBN
   public String getIsbn () {
      return isbn;
   }
```

```
// 8.3.a Le nom de l'auteur
public String getNomAuteur () {
   return nomAuteur;
}
// 8.3.a Le prénom de l'auteur
public String getPrenomAuteur () {
   return prenomAuteur;
}
// 8.3.a Le code d'enregistrement
public String getCode() {
   code= setCode ();
   return code;
}
public void afficherUnLivre(){
   System.out.println("Titre : " + titre);
   System.out.println("Auteur : " + nomAuteur + " " + prenomAuteur);
   System.out.println("Categorie : " + categorie);
   System.out.println("ISBN : " + isbn);
   // 8.4 Appel de la méthode invisible
   System.out.println("Code : " + getCode());
}
// 8.4 La méthode calculerLeCode() est renommée getCode()
// Elle est définie en mode privée pour ne pas etre accessible
// depuis l'extérieur. C'est une méthode invisible (métier).
private String setCode () {
   String debutNom;
   String debutPrenom;
   String debutCategorie;
   int longueurIsbn;
   String finIsbn;
   debutNom=nomAuteur.substring(0,2);
   debutPrenom=prenomAuteur.substring(0,2);
   debutCategorie=categorie.substring(0,2);
   longueurIsbn=isbn.length();
   finIsbn=isbn.substring((longueurIsbn-2),longueurIsbn);
   return debutNom+debutPrenom+debutCategorie+finIsbn;
}
// 8.5.a Définition du constructeur par défaut
public Livre() {
   Scanner lectureClavier = new Scanner(System.in);
   System.out.print("Entrez le titre : ");
   setTitre(lectureClavier.next());
```

```
        System.out.print("Entrez la categorie : ");
        setCategorie(lectureClavier.next());
        System.out.print("Entrez le nom de l'auteur : ");
        setNomAuteur(lectureClavier.next());
        System.out.print("Entrez le prenom de l'auteur : ");
        setPrenomAuteur(lectureClavier.next());
        System.out.print("Entrez le numero ISBN : ");
        setIsbn(lectureClavier.next());
    }
    // 8.5.c Surcharge du constructeur
    public Livre(String t, String c, String na, String pa, String i) {
        setTitre(t);
        setCategorie(c);
        setNomAuteur(na);
        setPrenomAuteur(pa);
        setIsbn(i);
    }
}
```

La classe `Bibliotheque` :

```
import java.util.*;
public class Bibliotheque {
    public static void main(String [] arg){
        Scanner lectureClavier = new Scanner(System.in);
        // 8.5.b Appel du constructeur par défaut
        Livre livrePoche = new Livre();
        // 8.5.b Les instructions de saisie des propriétés sont à supprimer
// à partir de l'exercice 8.5
        // 8.2.c Modifie la propriété d'un livre par une méthode d'accès en écriture
        System.out.print("Entrez le titre : ");
        livrePoche.setTitre(lectureClavier.next());
        // 8.2.c Modifie la propriété d'un livre par une méthode d'accès en écriture
        System.out.print("Entrez la categorie : ");
        livrePoche.setCategorie(lectureClavier.next());
        // 8.2.c Modifie la propriété d'un livre par une méthode d'accès en écriture
        System.out.print("Entrez le nom de l'auteur : ");
        // 8.2.c Modifie la propriété d'un livre par une méthode d'accès en écriture
        livrePoche.setNomAuteur(lectureClavier.next());
        System.out.print("Entrez le prenom de l'auteur : ");
        // 8.2.c Modifie la propriété d'un livre par une méthode d'accès en écriture
        livrePoche.setPrenomAuteur(lectureClavier.next());
        System.out.print("Entrez le numero ISBN : ");
        // 8.2.c Modifie la propriété d'un livre par une méthode d'accès en écriture
```

```
        livrePoche.setIsbn(lectureClavier.next());
        // 8.3.b Affiche une propriété d'un livre
        System.out.print("Code du livre " + livrePoche.getTitre()+ " : ");
        System.out.println(livrePoche.getCode());
        // Affiche toutes les propriétés
        livrePoche.afficherUnLivre();
        // 8.5.d Appel du constructeur paramétré
        Livre unPolar = new Livre( "Le mystère de la chambre jaune","Leroux", "Gaston",
                                   "Policier", "2253005495");
        unPolar.afficherUnLivre();
    }
}
```

8.1 Les propriétés `titre`, `categorie`, `isbn`, `nomAuteur`, `prenomAuteur` et `code` sont déclarées en mode privé. Elles ne sont plus accessibles depuis la fonction `main()`, cette dernière étant définie à l'extérieur de la classe `Livre`. Le compilateur détecte les erreurs :

- `Variable titre in class Livre not accessible from class Bibliotheque.`
- `Variable categorie in class Livre not accessible from class Bibliotheque.`
- `Variable isbn in class Livre not accessible from class Bibliotheque.`
- `Variable nomAuteur in class Livre not accessible from class Bibliotheque.`
- `Variable prenomAuteur in class Livre not accessible from class Bibliotheque.`
- `Variable code in class Livre not accessible from class Bibliotheque.`

8.2 Voir les commentaires du code source des classes `Livre` et `Bibliotheque` ci-avant.

8.2.d Le calcul du code est réalisé automatiquement par la méthode `calculerLeCode()`. Pour être sûr que le code soit toujours valide, il ne peut être ni saisi, ni modifié par l'utilisateur. La méthode `calculerLeCode()` ne doit pas être exécutable par une autre classe que la classe `Livre`.

8.3 Voir les commentaires du code source des classes `Livre` et `Bibliotheque` ci-avant.

8.4 Voir les commentaires du code source des classes `Livre` et `Bibliotheque` ci-avant.

8.5 Voir les commentaires du code source des classes `Livre` et `Bibliotheque` ci-avant.

8.5.b Le constructeur par défaut `Livre()` demande la saisie des données associées à l'objet `livrePoche`. Les instructions qui suivent demandent à nouveau la saisie de ces valeurs. Ces instructions sont redondantes. Elles sont devenues inutiles depuis la mise en place du constructeur. Il convient donc de les supprimer.

Exercice 8-6 à 8.9 : L'Héritage

Corrigé:

La classe `Forme` (exercice 8-6) :

```
import java.util.*;
public class Forme implements CalculGeometrique {
   // a. Définition des propriétés en mode protected
```

```
protected int x, y, couleur ;
// b. Définition des constantes
public final static int largeurEcran = 800 ;
public final static int hauteurEcran = 600 ;
public final static int couleurMax = 10 ;
// c. Définition de la méthode métier verifier()
//La méthode doit être accessible par les classes filles
//Elle est déclarée en mode protected
protected int verifier(String s, int a, int b) {
   int resultat;
   Scanner lectureClavier = new Scanner(System.in);
do{
   System.out.print(s+":");
   resultat= lectureClavier.nextInt()
} while ( resultat < a || resultat > b) ;
return resultat ;
}
// c. surcharge de la méthode métier verifier()
protected int verifier(int tmp, int a, int b) {
   if (tmp < a) return a;
   else if ( tmp > b) return b ;
   else return tmp;
}
// d. Définition du constructeur par défaut
public Forme() {
   x = verifier("en X", 0, largeurEcran);
   y = verifier("en Y", 0, hauteurEcran);
   couleur = verifier("couleur", 0, couleurMax);
}
// d. Surcharge du constructeur
public Forme(int nx, int ny, int nc) {
   x = verifier(nx, 0, largeurEcran) ;
   y = verifier(ny, 0, hauteurEcran) ;
   couleur = verifier(nc, 0, couleurMax);
}
// e. Définition de la méthode deplacer()
public void deplacer(int nx, int ny) {
   x = verifier(x+nx, 0, largeurEcran) ;
   y = verifier(y+ny, 0, hauteurEcran) ;
}
// f. Définition de la méthode colorier()
public void colorier(int nc) {
```

```
        couleur = verifier(nc, 0, couleurMax);
    }
    // g. Définition de la méthode afficher()
    public void afficher() {
        System.out.println("Couleur : " + couleur);
        System.out.println("Position en " + x + ", " + y);
    }
    public double surface() {
        return -1 ;
    }
    public double perimetre() {
        return -1 ;
    }
} // Fin de la classe Forme
```

La classe `Rectangle` (exercice 8-7) :

```
// a. La classe Rectangle hérite de la classe Forme
public class Rectangle extends Forme {
    // b. Définition des propriétés spécifiques à un rectangle
    private int largeur, hauteur;
    // c. Définition du constructeur par défaut
    public Rectangle() {
        largeur = verifier("Largeur", 0, largeurEcran);
        hauteur = verifier("Hauteur", 0, hauteurEcran);
    }
    // c. Surcharge du constructeur
    public Rectangle(int nx, int ny, int nl, int nh, int nc){
        super(nx, ny, nc);
        largeur = verifier(nl, 0, largeurEcran);
        hauteur = verifier(nh, 0, hauteurEcran);
    }
    // d. Définition de la méthode afficher()
    public void afficher() {
        super.afficher();
        System.out.println("Largeur du rectangle : " + largeur);
        System.out.println("Hauteur du rectangle : " + hauteur);
    }
    // e. Définition de la méthode perimetre()
    public double perimetre() {
        return 2*(largeur+hauteur);
    }
    public double surface() {
        return largeur*hauteur ;
```

```
    }
} // Fin de la classe Rectangle
```

La classe `Triangle` (exercice 8-8) :

```
// a. La classe Triangle hérite de la classe Forme
public class Triangle extends Forme {
    // b. Définition des propriétés spécifiques à un triangle
    private int xB, yB, xC, yC;
    // c. Définition du constructeur par défaut
    public Triangle() {
        xB = verifier("second sommet en X : ", 0, largeurEcran);
        yB = verifier("second sommet en Y : ", 0, hauteurEcran);
        xC = verifier("troisieme sommet en X : ", 0, largeurEcran);
        yC = verifier("troisieme sommet en Y : ", 0, hauteurEcran);
    }
    // c. Surcharge du constructeur
    public Triangle(int nxa, int nya, int nxb, int nyb, int nxc, int nyc, int nc )    {
        super(nxa, nya, nc);
        xB = verifier(nxb, 0, largeurEcran);
        yB = verifier(nyb, 0, hauteurEcran);
        xC = verifier(nxc, 0, largeurEcran);
        yC = verifier(nyc, 0, hauteurEcran);
    }
    // d. Définition de la méthode afficher()
    public void afficher(){
        super.afficher();
        System.out.println("Position en " +xB + ", " + yB);
        System.out.println("Position en " +xC + ", " + yC);
    }
    // 8.e. Définition de la méthode deplacer()
    public void deplacer(int nx, int ny){
        super.deplacer(nx, ny);
        xB = verifier(xB+nx, 0, largeurEcran);
        yB = verifier(yB+ny, 0, hauteurEcran);
        xC = verifier(xC+nx, 0, largeurEcran);
        yC = verifier(yC+ny, 0, hauteurEcran);
    }
} // Fin de la classe Triangle
```

L'application `FaireDesFormesGeometriques` (exercice 8-9) :

```
public class FaireDesFormesGeometriques {
    // a. Créer un cercle, un rectangle et un triangle
    public static void main(String [] args) {
        // b. La vérification des valeurs est réalisée par chaque constructeur
```

```
        Cercle C= new Cercle(105, 105, 20, 10);
        // c. affiche un cercle
        C.afficher();
        C.deplacer(100, 100);
        C.afficher();
        // f. affiche le périmètre et la surface du cercle C
        if ( C.perimetre() >=0 ||  C.surface() >=0  ) {
           System.out.println("Le perimetre de C vaut " + C.perimetre());
           System.out.println("La surface de C vaut " + C.surface());
        } else {
           System.out.println("Calcul impossible");
        }
        Rectangle R = new Rectangle(100, 100, 50, 20, 5);
        // c. affiche un rectangle
        R.afficher();
           R.deplacer(200, 200);
        R.afficher();
        // f. affiche le périmètre et la surface du rectangle R
        if ( R.perimetre() >=0 ||  R.surface() >=0  ) {
           System.out.println("Le perimetre de R vaut " + R.perimetre());
           System.out.println("La surface de R vaut " + R.surface());
        } else {
            System.out.println("Calcul impossible");
        }
        Triangle T = new Triangle (200, 200, 100, 300, 300, 300, 2);
        // c. affiche un triangle
        T.afficher();
        // d. et e. le déplacememnt du triangle consiste à déplacer tous les sommets du
        //triangle et pas seulement le point de référence (voir méthode deplacer()
        //de la classe Triangle ci-avant)
        T.deplacer(300, 300);
        T.afficher();
    }
}
```

Exercice 8-10 à 8.11 : Les Interfaces

Corrigé:

L'interface `Deplacement` (exercice 8-10) :

```
// a. Définition de l'interface Deplacement
interface Deplacement {
  // a. Définition du comportement seDeplacer()
 public void seDeplacer();
```

```
 // a. Définition du comportement deplacementEnX()
 // Par défaut il n'y a pas de déplacement en X
 default public void deplacementEnX() {
 }
// a. Définition du comportement deplacementEnY()
// Par défaut il n'y a pas de déplacement en Y
 default public void deplacementEnY() {
 }
} // Fin de la l'interface Deplacementpublic
```

La classe `MoyenDeTransport` (exercice 8-10) :

```
// b. Création de la classe MoyenDeTransport
public class MoyenDeTransport implements Deplacement{
 // b. Création des propriétés x,y,vitesse et constante vitesseInitiale
 public final static int vitesseInitiale = 10 ;
 protected int x, y ;
 protected int vitesse ;
 public MoyenDeTransport(int nx, int ny) {
    x = nx ;
    y = ny ;
 }
// d. Un moyen de transport possède une vitesse bornée
 protected int limitation (int tmp, int max) {
   tmp = vitesseInitiale * tmp;
   if (tmp < 0) return 0;
   else if ( tmp > max) return max ;
   else return tmp;
 }
// c. Definition de la méthode afficher()
 public void afficher() {
  System.out.println("Position en " + x + ", " + y);
 }
// c. Un moyen de transport se déplace en X et en Y
 // selon les méthode définies par l'interface Deplacement
 public void seDeplacer(){
   deplacementEnX();
   deplacementEnY();
 }
// c. les méthodes deplacementEnX() et deplacementEnY n'ont pas besoin d'être
//    redéfinies ces dernières sont définies en méthode par défaut dans l'interface
//    Deplacement
}
```

La classe `Voiture` (exercice 8-11) :

```
// a. Définition de la classe Voiture héritant de la classe MoyenDeTransport
public class Voiture extends MoyenDeTransport {
public Voiture(int xx, int yy)   {
   super(xx, yy);
   // a. Verifier que la vitesse de la voiture ne dépasse pas 100
   vitesse = limitation(10, 100);
   System.out.println("Je suis une voiture ! ");
   System.out.println("Vitesse : " + vitesse);
 }
// a. Une voiture se déplace sur l'axe des X
//    la méthode deplacementEnX() est redéfinie
 public void deplacementEnX()   {
       x+=vitesse;
       System.out.println("x : " + x);
 }
// a. Une voiture ne se déplace pas sur l'axe des Y
//    La méthode deplacementY() est celle définie par l'interface Deplacement
} // Fin de la classe Voiture
```

La classe `Ascenceur` (exercice 8-11) :

```
// b. Définition de la classe Fusee héritant de la classe MoyenDeTransport
public class Ascenseur extends MoyenDeTransport {
public Ascenseur(int xx, int yy) {
   super(xx, yy);
   // c. Verifier que la vitesse de la fuséée ne dépasse pas 20
   vitesse = limitation(5, 20);
   System.out.println("Je suis un ascenseur ! ");
   System.out.println("Vitesse : " + vitesse);
 }
 // b. Un ascenceur se déplace sur l'axe des Y
 //    la méthode deplacementEnY() est redéfinie
 public void deplacementEnY()   {
    y+=vitesse;
    System.out.println("y : " + y);
 }
 // b. Un ascenceur ne se déplace pas sur l'axe des X
 //    La méthode deplacementX() est celle définie par l'interface Deplacement
} // Fin de la classe Ascenseur
```

La classe `Fusee` (exercice 8-11) :

```
public class Fusee extends MoyenDeTransport {
public Fusee(int xx, int yy)  {
   super(xx, yy);
    // c. Verifier que la vitesse de la fuséée ne dépasse pas 2000
```

```
    vitesse = limitation(100, 2000);
    System.out.println("Je suis une fusee ! ");
    System.out.println("Vitesse : " + vitesse);
  }
 // c. Une fusée se déplace sur l'axe des X
  public void deplacementEnX()   {
       x+=vitesse;
       System.out.println("x : " + x);
  }
  // c. Une fusée se déplace sur l'axe des Y
  public void deplacementEnY()   {
       y+=vitesse;
       System.out.println("y : " + y);
  }
 } // Fin de la classe Fusee
```

Partie 3 : Les outils et techniques orientés objet

Chapitre 9 : Collectionner un nombre fixe d'objets

Exercice 9-1 : Les tableaux à une dimension

Corrigé:

I	valeur[i]	valeur.length	Affichage
0	1	6	
1	1+2	6	`i < valeur.length`
2	3+2	6	`i < valeur.length`
3	5+2	6	`i < valeur.length`
4	7+2	6	`i < valeur.length`
5	9+2	6	`i < valeur.length`
6	**-**	**6**	`i = valeur.length` sortie de boucle
0	1	6	`i < 6` affiche `valeur[0] = 1`
1	3	6	`i < 6` affiche `valeur[1] = 3`
2	5	6	`i < 6` affiche `valeur[2] = 5`
3	7	6	`i < 6` affiche `valeur[3] = 7`
4	9	6	`i < 6` affiche `valeur[4] = 9`
5	11	6	`i < 6` affiche `valeur[5] = 11`
6	**-**	**6**	`i =6` sortie de boucle

A l'écran le résultat s'affiche de la façon suivante :

- pour la première boucle :

  ```
  valeur[0] = 1
  valeur[1] = 3
  valeur[2] = 5
  valeur[3] = 7
  ```

- pour la seconde boucle :

  ```
  valeur = 1
  valeur = 3
  valeur = 5
  valeur = 7
  ```

```
valeur[4] = 9                    valeur = 9
valeur[5] = 11                   valeur = 11
```

La différence entre le premier et le second affichage se situe dans l'affichage du compteur de boucle. Dans la seconde boucle, il n'est pas possible d'afficher l'indice du tableau, puisque le compteur de boucle est pris en charge par Java. S'il on souhaite afficher l'indice du tableau, il convient de placer soit même un compteur de boucle à l'intérieur de la boucle, comme suit :

```
int i = 0 ;
for (int v : valeur) {
   System.out.println("valeur [" + i +" ] = " + v);
   i++ ;
}
```

Exercice 9-2 : Les tableaux à une dimension

Corrigé :

```
public class Exercice2{
 public static void main (String [] argument){
  int laPlusGrande, laPlusPetite, laSomme = 0, iMax = 0, nbMoy = 0 ;
  double laMoyenne;
  if (argument.length > 0) {
     int [] valeur = new int [argument.length];
     for (int i= 0; i < argument.length; i++)
        // a. stocke dans un tableau, des valeurs entières passées
        //en paramètre de la ligne de commande ;
        valeur[i] = Integer.parseInt(argument[i]);
     laPlusGrande = valeur[0] ;
     for (int i= 0; i < valeur.length; i++) {
        if (laPlusGrande < valeur[i]) {
        // d. recherche la plus grande des valeurs
           laPlusGrande = valeur[i] ;
        // e. détermine la position de la plus grande des valeurs
           iMax = i;
        }
        // b. calcule la somme de ces valeurs
        laSomme = laSomme + valeur[i] ;
     }
     // c. calcule la moyenne de ces valeurs
     laMoyenne = (float) laSomme / valeur.length ;
     for (int i= 0; i < valeur.length; i++)
        // f. calcule le nombre de valeurs supérieure à la moyenne
        if ( valeur[i] >= laMoyenne) nbMoy++;
     System.out.println("La plus grande valeur est : " + laPlusGrande) ;
     System.out.println("A l'indice : " + iMax + " du tableau ") ;
     System.out.println("La moyenne de ces valeurs : " + laMoyenne) ;
     System.out.println("Nombre de valeurs > a la moyenne : " + nbMoy) ;
  }
```

```
    else System.out.println("Commande : java Exercice2 listeDesValeursEntieres ");
   } // Fin du main ()
  }
```

Exercice 9-3 : Les tableaux d'objets

Corrigé :

```
public class Triangle extends Forme {
 public final static int TailleEcran = 600 ;
   // 9.3.a Définition des sommets du triangle sous forme de tableaux
   private int [] xPoints = new int[3];
   private int [] yPoints = new int[3];
   // 9.3.b Définition des constructeurs
   public Triangle(int nxa, int nya, int nxb, int nyb, int nxc, int nyc, int nc)    {
      // Le premier sommet correspond au point de référence de la forme
      // il est traité par le constructeur de la classe Forme
      super(nxa, nya, nc);
      // Le premier élément du tableau correspond au point de référence de la
      // Forme, il prend les valeurs des propriétés de la classe Forme. Elles ont
      // été vérifiée par le constructeur super(...)
      xPoints[0] = x;
      yPoints[0] = y;
      xPoints[1] = verifier(nxb, 0, largeurEcran);
      yPoints[1] = verifier(nyb, 0, hauteurEcran);
      xPoints[2] = verifier(nxc, 0, largeurEcran);
      yPoints[2] = verifier(nyc, 0, hauteurEcran);
   }
   // 9.3.b Définition des constructeurs
   public Triangle() {
      // Le premier élément du tableau correspond au point de référence de la
      // Forme, il prend les valeurs saisies par le constructeur de la classe Forme
      xPoints[0] = x;
      xPoints[0] = y;
      for(int i=1; i < xPoints.length; i++){
          xPoints[i] = verifier("["+i+"] en X", 0, largeurEcran);
          yPoints[i] = verifier("["+i+"] en Y", 0, hauteurEcran);
      }
   }
   // 9.3.c Les sommets s'affichent à l'aide d'une boucle for
   public void afficher(){
      // 9.3.c Le premier sommet est affiché par la méthode afficher()
      // de la classe Forme
      super.afficher();
  // 9.3.c Les sommets suivants sont affichés à partir de l'indice 1
```

```
        for (int i=1; i < 3; i++){
            System.out.println("Position en " +xPoints[i] + ", " + yPoints[i]);
        }
    }
    // 9.3.c Les sommets se déplacent à l'aide d'une boucle for
    public void deplacer(int nx, int ny){
        // 9.3.c Le premier sommet est déplacé par la méthode deplacer()
        // de la classe Forme
        super.deplacer(nx, ny);
        xPoints[0] = x;
        yPoints[0] = y;
        // 9.3.c Les sommets suivants sont déplacés à partir de l'indice 1
        for(int i=1; i < xPoints.length; i++){
            xPoints[i] = verifier(xPoints[i]+nx, 0, largeurEcran);
            yPoints[i] = verifier(yPoints[i]+ny, 0, hauteurEcran);
        }
    }
}
```

Exercice 9-4 : Les tableaux d'objets

Corrigé :

```
public class FaireDesCercles {
 public static void main(String [] arg) {
   Scanner lectureClavier = new Scanner(System.in);
   System.out.print("Combien de Cercles : ");
   int nbCercle = lectureClavier.nextInt();
   if (nbCercle < 3) nbCercle = 4;
   // a. Créer un tableau de type Cercle, dont la taille est choisie
   //par l'utilisateur….
   Cercle [] C= new Cercle [nbCercle];
   for (int i = 0; i < C.length; i++)
      // b. Initialisation par le constructeur par défaut
      C[i] = new Cercle();
   System.out.println("----------Recapitulatif---------- ");
   for (int i = 0; i < C.length; i++)  C[i].afficher();
   // c. Déplacement du cercle 1 en 20, 20
   System.out.println("-----Cercle 1 : Deplacement en 20,20----- ");
   C[1].déplacer(20, 20);
   C[1].afficher();
   // d. Agrandissement du cercle 1 de 50
   System.out.println("-----Cercle 2 : Agrandissement de 50 ---- ");
   C[2].agrandir(50);
   C[2].afficher();
```

```
    // e. Echange du cercle 0 avec le cercle 3
    System.out.println("---- Echange du Cercle 0 avec le cercle 3 ---- ");
    C[0].échangerAvec(C[3]);
    C[0].afficher();
    C[3].afficher();
    // f. Permute les cercles, le cercle 0 va en 1, le cercle 1 en 2 etc.
    System.out.println("----------Permutations ----------- ");
    for ( int i = 0;i < C.length; i++)
       C[i].échangerAvec(C[0]);
    for ( int i = 0;i < C.length; i++)
       C[i].afficher();
  }
 }
```

Exercice 9-5 : Les tableaux à deux dimensions

Corrigé :

```
public class Exercice5 {
 public static void main(String[] arg){
   int [][] Etoile = new int[7][7];
   // a. Initialisation du tableau
   for (int i = 0; i < Etoile.length; i++) {
      for (int j = 0; j < Etoile[0].length; j++) {
         // a. Initialise la première diagonale
         Etoile[i][i] = 1;
         // a. Initialise la troisième ligne
         Etoile[3][i] = 1;
         // a. Initialise la troisième colonne
         Etoile[i][3] = 1;
         // a. Initialise la deuxième diagonale
         Etoile[i][6-i] = 1;
      }
   }
   // b. Affichage du tableau
   for (int i = 0; i < Etoile.length; i++) {
      for (int j = 0; j < Etoile[0].length; j++) {
         if(Etoile[i][j] == 0)   System.out.print(" ");
         else  System.out.print("*");
      }
      System.out.println();
   }
 } // Fin de la fonction main()
} // Fin de la classe Exercice4
```

Exercice 9-6 : Pour mieux comprendre le mécanisme des boucles imbriquées (for – for)

Corrigé :

```
import java.util.*;
public class Exercice6 {
 public static void main (String [] parametre) {
   int i,j, N = 5;
   char C;
   Scanner lectureClavier = new Scanner(System.in);
   System.out.print("Entrer uncaractere : ");
   C = lectureClavier.next().charAt(0);
   for (i = 1; i < N; i++){
      for (j = 1; j < N; j++){
         if (i < j) System.out.print(C);
         else System.out.print(" ");
      }
   }
 }
}
```

a. Repérer les instructions concernées par les deux boucles répétitives : voir tracé **orange** sur le programme ci-dessus.

b. Déterminer les instructions de début et fin de boucle : voir tracé **vert** sur le programme ci-dessus.

Recherchez les instructions qui permettent de modifier le résultat du test de sortie de boucle : voir tracé **jaune** sur le programme ci-dessus.

c. Tableau d'évolution des variables :

i	j	N	C	Affichage
1	1	2	!	i = j affiche un espace
1	2	2	!	i < j affiche un !
1	3	2	!	i < j affiche un !
1	4	2	!	i < j affiche un !
1	**5**	2	!	j > N sortie de boucle
2	1	2	!	i > j affiche un espace
2	2	2	!	i = j affiche un espace
2	3	2	!	i < j affiche un !
2	4	2	!	i < j affiche un !
2	**5**	2	!	j > N sortie de boucle
3	1	2	!	i > j affiche un espace
3	2	2	!	i > j affiche un espace
3	3	2	!	i = j affiche un espace
3	4	2	!	i < j affiche un !
3	**5**	2	!	j > N sortie de boucle
4	1	2	!	i > j affiche un espace
4	2	2	!	i > j affiche un espace
4	3	2	!	i > j affiche un espace
4	4	2	!	i = j affiche un espace
4	**5**	2	!	j > N sortie de boucle

i	j	N	C	Affichage
5	5	2	!	i > N sortie de boucle

d. Ce programme a pour résultat :

```
Entrer un caractère : !
 !!!!! !
```

Exercice 9-7 : Pour mieux comprendre le mécanisme des boucles imbriquées (for – for)

Corrigé :

i	Affichage
1	affiche un i = 1
2	affiche un i = 2
3	sortie de la boucle interne
4	affiche un i = 4
1	affiche un i = 1
2	affiche un i = 2
3	sortie de la boucle interne
4	affiche un i = 4
1	affiche un i = 1
2	affiche un i = 2
3	etc. Le programme boucle !

Chapitre 10 : Collectionner un nombre indéterminé d'objets

Exercice 10-1 : Comprendre les listes

Corrigé :

```
import java.util.*;
import java.lang.Number.*;
public class Etudiant      {
   private String nom, prénom;
   // a. Définiton d'une liste de notes ne pouvant contenir que des Double
   private ArrayList<Double> notes;
   private double moyenne;
   public Etudiant()    {
      Scanner lectureClavier = new Scanner(System.in);
      System.out.print("Entrer le nom de l'etudiant : ");
      nom = lectureClavier.next();
      System.out.print("Entrer le prenom de l'etudiant : ");
      prénom = lectureClavier.next();
      System.out.print("Combien de notes pour l'etudiant");
      System.out.print( prénom + " " + nom + " :");
      int nombre = lectureClavier.nextInt();
      notes = new ArrayList()<Double>;
      for (int i = 0; i < nombre; i ++)   {
         System.out.print("Entrer la noten° "+ (i+1)+ " :");
```

```
            // b. Mise en place des notes dans la liste
            //Les notes saisies sont préalablement transformées en objet Double
            notes.add(new Double(lectureClavier.nextDouble()));
        }
        moyenne = calculMoyenne();
    }
    // c. Calcul de la moyenne d'une note, notez la nouvelle syntaxe de la boucle for
    private double calculMoyenne()   {
        double somme = 0.0;
        int nbnotes = notes.size();
        for(Double valeur : notes) {
            somme = somme + valeur;
        }
        return somme/nbnotes;
    }
    public double quelleMoyenne() {
        return moyenne;
    }
    public void afficheUnEtudiant(){
    // d. Modification de l'affichage des notes
        System.out.print(" Les notesde "+prénom+" "+nom+ " sont : ");
        int nbnotes = notes.size();
        for(Double valeur : notes)
            System.out.print(" "+ valeur);
        System.out.println();
        System.out.println("Sa moyenne vaut"+ moyenne);
    }
}
```

Exercice 10-3 et 10-4 : Comprendre les listes et stream

Corrigé

10.3.a et 10.4.a La classe `Etudiant`

```
public class Etudiant      {
    private String nom, prenom;
    private double [] notes;
    private double moyenne;
    public Etudiant()    {
          Scanner lectureClavier = new Scanner(System.in);
        System.out.print("Entrer le nom de l'etudiant : ");
        nom = lectureClavier.next();
        System.out.print("Entrer le prenom de l'etudiant : ");
        prenom = lectureClavier.next();
```

```
        System.out.print("Combien de notes pour l'etudiant  ");
        System.out.print( prenom + " " + nom + " :  ");
        int nombre = lectureClavier.nextInt();
        notes = new double [nombre];
        for (int i = 0; i < notes.length; i ++)  {
         System.out.print("Entrer la note  numero "+ (i+1)+ " :  ");
         notes[i] = lectureClavier.nextDouble();
        }
        moyenne = calculMoyenne();
    }
    public void afficheUnEtudiant()  {
        System.out.print("Les notes  de " + prenom + " " + nom + " sont : ");
        for(double valeurNote : notes)   System.out.print(" " + valeurNote);
        System.out.println();
        System.out.println("Sa moyenne vaut " + moyenne);
    }

    private double calculMoyenne()  {
        double somme = 0.0;
        for(double valeurNote : notes) somme = somme + valeurNote;
        return somme/notes.length;
    }
     // 10.4.a. Définition de la méthode quelNom()
    public int quelleMoyenne() {
        return (int) moyenne;
    }
     // 10.3.a. Définition de la méthode quelNom()
    public String quelNom() {
        return nom;
    }

}
```

10.3.b et 10.4.b La classe `Cursus`

```
public class Cursus  {
 private ArrayList<Etudiant>  liste;
 public Cursus()   {
   liste = new ArrayList<Etudiant>();
 }
 public void ajouteUnEtudiant() {
   liste.add(new Etudiant());
 }
 public void afficheLesEtudiants()  {
```

```
    int nbEtudiants = liste.size();
    if (nbEtudiants > 0) {
                    for (Etudiant e : liste) e.afficheUnEtudiant();
    }
    else System.out.println("Il n'y a pas d'etudiant dans cette liste");
  }
   // 10.4.b Pour chaque element calculer la plus grande moyenne et afficher
   // l'étudiant correspondant à ce max
   public void rechercheLeMajor()    {
      Etudiant   max =liste.stream()
     .max((e1, e2) -> (e1.quelleMoyenne() - e2.quelleMoyenne()))
     .get();
      max.afficheUnEtudiant();
 }
   // 10.3.b Rechercher un étudiant par le prrénom passé en paramètre
   public void rechercheUnEtudiant(String tmp)   {
   // creation d'un flux de traitement
    liste.stream() // Pour chaque element du flux, rechercher ceux commençant par les
                   // lettres contenues ds tmp
     .filter(e -> e.quelNom().startsWith(tmp))
     .forEach(e -> e.afficheUnEtudiant()); // Pour chaque element commençant par les
                                           // lettres contenues dans tmp, afficher les
                                           // informations
   }
}
```

10.3.c et 10.4.c La classe `GestionCursus`

```
public class GestionCursus  {
 public static void main(String [] argument)  {
   byte choix = 0 ;
   Cursus C = new Cursus();
        Scanner lectureClavier = new Scanner(System.in);
   do   {
      System.out.println("1. Ajoute un etudiant");
      System.out.println("2. Affiche la liste des etudiants");
      System.out.println("3. Recherche le major de la promotion");
      System.out.println("4. Pour sortir");
      System.out.print("Votre choix : ");
      choix  = lectureClavier.nextByte();
      switch (choix) {
         case 1 :    C.ajouteUnEtudiant();
         break;
         case 2 :    C.afficheLesEtudiants();
```

```
            break;
            case 3 :        // 10.3.c Rechercher un étudiant pas son prénom
                            System.out.print("Nom de l'étudiant recherche : ");
                            nom  = lectureClavier.next();
                            C.memePrenom(nom);
            case 4 :        // 10.3.c Rechercher l'étudiant qui a la plus grande moyenne
                            C.rechercheLeMajor();
            break;
            case 5 :    System.exit(0);
            default :
            System.out.println("Cette option n'existe pas ");
         }
      } while (choix != 5);
   }
}
```

Exercice 10.2, 10-5 et 10-9 : Comprendre les listes et créer des fichiers texte

Corrigé

La classe `FaireDesListesDeFormes`

```
import java.util.*;
import java.io.IOException;
public class ListeDeFormes{
   // 10.5.a Définition de la propriété listeFormes (liste de <Forme>)
   private ArrayList<Forme> listeFormes;
   // 10.5.b Définition du constructeur de la liste
   public ListeDeFormes()  {
      listeFormes = new ArrayList<Forme>();
   }
   // 10.5.c La méthode ajoute une forme à la liste
   public void ajouterUneForme(char type){
      // 10.5.c Si le paramètre vaut 'C', la forme ajoutée à la liste est un cercle
      if (type == 'C') {
         listeFormes.add(new Cercle());
      }
      // 10.5.c Si le paramètre vaut 'T', la forme ajoutée à la liste est un triangle
      else if(type == 'T'){
         listeFormes.add(new Triangle());
      }
      // 10.5.c Dans tous les autres cas, la forme ajoutée à la liste est un
rectangle
      else {
         listeFormes.add(new Rectangle());
      }
```

```
    }
    // 10.5.d Affiche la liste des formes, tmp sait s'il est un cercle, un triangle ou
    un rectangle, il appelle la méthode afficher() correspondant à son type
    public void afficherLesFormes() {
       int nbFormes = listeFormes.size();
       if (nbFormes > 0) {
          for (Forme tmp : listeFormes) tmp.afficher();
       }
       else {
          System.out.print("Il n'y a pas de forme dans cette liste ");
       }
    }
    // 10.9.b Parcourt la liste et écrit dans un fichier, une ligne par forme
    //en utilisant la chaîne construite par la méthode getInfos()
    public void enregistrerLesFormes(Fichier f) throws IOException {
       int nbFormes = listeFormes.size();
       if (nbFormes > 0) {
           String chaine = "";
          for (Forme tmp : listeFormes){
             tmp.afficher();
             chaine = tmp.getInfos();
             if (chaine != null)f.ecrire(chaine);
          }
       }
    }
    // 10.9.e Lit le fichier ligne par ligne et l'enregistre dans la liste
    public void lireLesFormes(Fichier f) throws IOException {
       String chaine;
       String [] mot = {"", "","", "", "","","",""};
       char tt;
       int cc, xx0, yy0, xx1, yy1, xx2, yy2, ll, hh, rr;
       do {
          // 10.9.e Lit le fichier ligne par ligne et récupère la ligne lue sous la
          // forme d'un tableau. Examiner la méthode lire() de la classe Fichier.
          mot = f.lire();
          if (mot != null) {
          tt = mot[0].charAt(0);
          // 10.9.de Si le premier élément du tableau récupéré est un 'C'
          if ( tt == 'C') {
             // 10.9.d Transformer les autres éléments du tableau en entier
             cc = Integer.parseInt(mot[1]);
             xx0 = Integer.parseInt(mot[2]);
```

```
            yy0 = Integer.parseInt(mot[3]);
            rr = Integer.parseInt(mot[4]);
            // 10.9.e Ajouter à la liste un cercle en appelant le constructeur avec
            //comme paramètre, les valeurs lues dans le fichier
            listeFormes.add(new Cercle( xx0, yy0, rr, cc));
         }
         // 10.9.d Si le premier élément du tableau récupéré est un 'T'
         else if (tt == 'T'){
            // 10.9.d Transformer les autres éléments du tableau en entier
            cc = Integer.parseInt(mot[1]);
            xx0 = Integer.parseInt(mot[2]);
            yy0 = Integer.parseInt(mot[3]);
            xx1 = Integer.parseInt(mot[4]);
            yy1 = Integer.parseInt(mot[5]);
            xx2 = Integer.parseInt(mot[6]);
            yy2 = Integer.parseInt(mot[7]);
            // 10.9.d Ajouter à la liste un triangle en appelant le constructeur avec
            //comme paramètre, les valeurs lues dans le fichier
            listeFormes.add(new Triangle( xx0, yy0, xx1, yy1, xx2, yy2, cc));
         }
          // 10.9.e Si le premier élément du tableau récupéré est un 'R'
         else if( tt == 'R') {
            // 10.9.e Transformer les autres éléments du tableau en entier
            cc = Integer.parseInt(mot[1]);
            xx0 = Integer.parseInt(mot[2]);
            yy0 = Integer.parseInt(mot[3]);
            ll = Integer.parseInt(mot[4]);
            hh = Integer.parseInt(mot[5]);
            // 10.9.e Ajouter à la liste un rectangle en appelant le constructeur
            //comme paramètre, les valeurs lues dans le fichier
            listeFormes.add(new Rectangle( xx0, yy0, ll, hh, cc));
         }
      }
   } while (mot != null);
 }
}
```

La classe `FaireDesListesDeFormes`

```
import java.util.*;
public class FaireDesListesDeFormes {
   public static void main(String[] args) throws IOException{
      Scanner lectureClavier = new Scanner(System.in);
      ListeDeFormes LdF= new ListeDeFormes();
```

```
        Fichier f = new Fichier();
        // 10.9.f Ouvrir le fichier en lecture
        if(f.ouvrir("Formes.txt", "Lecture")){
           // 10.9.f Si le fichier existe, lire les lignes et stocker les formes dans
           //la liste LdF
           LdF.lireLesFormes(f);
           f.fermer();
        }
        byte choix = 0;
        do{
           System.out.println("1. Ajouter un cercle");
           System.out.println("2. Ajouter un triangle");
           System.out.println("3. Ajouter un rectangle");
           System.out.println("4. Afficher la liste");
           System.out.println("5. Pour sortir");
           System.out.print("Votre choix : ");
           choix = lectureClavier.nextByte();
           switch(choix) {
              // 10.2.e Pour ajouter un cercle
              case 1 : LdF.ajouterUneForme('C');
              break;
              // 10.2.e Pour ajouter un triangle
              case 2 : LdF.ajouterUneForme('T');
              break;
              // 10.2.e Pour ajouter un rectangle
              case 3 : LdF.ajouterUneForme('R');
              break;
              // 10.2.e Afficher la liste des formes
              case 4 : LdF.afficherLesFormes();
              break;
              case 5 :
                  // 10.9.c Enregistre la liste des formes dans un fichier
                 f.ouvrir("Formes.txt", "Ecriture");
                 dF.enregistrerLesFormes(f);
                 f.fermer();
                 System.exit(0);
              default : System.out.println("Cette option n'existe pas");
           }
        }while (choix != 5);
     }
  }
```

10.7.a : La méthode `getInfos()`

- Dans la classe `Forme`

```
public String getInfos() {
   return couleur+";"+x+";"+y;
 }
```

La valeur retournée par la méthode `getInfos()` de la classe `Forme`, doit être insérée dans la chaîne construite par les classes fille, comme suit :

- Dans la classe `Cercle`

```
public String getInfos() {
   String tmp = super.getInfos();
   return "C;"+tmp+";"+rayon+";";
}
```

- Dans la classe `Triangle`

```
public String getInfos() {
   String tmp = super.getInfos();
   return "T;"+tmp+";"+xPoints[1]+";"+yPoints[1]+";"+xPoints[2]+";"+yPoints[2] ;
}
```

- Dans la classe `Rectangle`

```
public String getInfos() {
   String tmp = super.getInfos();
   return "R;"+tmp+";"+largeur+";"+hauteur;
}
```

10.7.d : La méthode `lire()` de la classe `fichier` :

```
// 10.7.d Lit et décompose une ligne du fichier
public String [] lire() throws IOException {
String ligne;
// 10.7.d Lit une ligne du fichier
ligne = fR.readLine();
if (ligne != null) {
   // 10.7.d Le séparateur de mot est le ";"
   StringTokenizer st = new StringTokenizer(ligne,";");
   int i=0;
   // 10.7.d Créer un tableau dont la longueur correspond au nombre de champs
   //séparé par des ";"
   String mot[] = new String[st.countTokens()];
   // 10.7.d Tant qu'il y a des champs séparés par des ";"
   while (st.hasMoreTokens()) {
       // 10.7.d enregistrer le champs courant dans le tableau à l'indice courant
      mot[i] = st.nextToken();
      i++;
   }
   // 10.7.d Retourner le tableau composé de la liste des champs
```

```
    return mot;
    }
    else return null;
}
```

Exercice 10-5 : Comprendre les dictionnaires

Corrigé

Cette méthode est à insérer dans la classe `Cursus`

```
// a. Définition des paramètres de la méthode
public void modifieUnEtudiant(String p, String n) {
    // b. Calcul de la clé
    String clé = créerUneClé(p, n);
    // b. Vérification de l'existence ou non de l'étudiant
    if (listeClassée.get(clé) != null) {
    // c. S'il existe, l'étudiant est modifié grâce au constructeur
          Etudiant eModifié = new Etudiant(p, n) ;
          listeClassée.put(clé, eModifié);
    }
    else System.out.println(p + " " + n + " est inconnu ! ");
}
```

c. Dans la classe `Etudiant` le constructeur est surchargé de la façon suivante :

```
public Etudiant(String p, String n)    {
    Scanner lectureClavier = new Scanner(System.in);
    nom = n;
    prénom = p;
    System.out.print("Combien de notes pour l'etudiant");
    System.out.print( prénom + " " + nom + " :");
    int nombre = lectureClavier.nextInt();
    notes = new double [nombre];
    for (int i = 0; i < notes.length; i ++)   {
        System.out.print("Entrer la noten° "+ (i+1)+ " :");
        notes[i] = lectureClavier.nextDouble();
    }
    moyenne = calculMoyenne();
}
```

d. La nouvelle classe `GestionCursus`

```
import java.util.*;
public class GestionCursus{
 public static void main (String [] argument) {
    Scanner lectureClavier = new Scanner(System.in);
    byte choix = 0 ;
    Cursus C = new Cursus();
```

```
String prénom, nom;
do{
    System.out.println("1. Ajoute un etudiant");
    System.out.println("2. Supprime un etudiant");
    System.out.println("3. Affiche la classe");
    System.out.println("4. Affiche un etudiant");
    // d. Indique le choix d'une la nouvelle option
    System.out.println("5. Modifie un etudiant");
    System.out.println("6. Sortir");
    System.out.println();
    System.out.print("Votre choix : ");
    choix= lectureClavier.nextByte();
    switch (choix) {
      case 1 : C.ajouteUnEtudiant();
      break;
      case 2 :
          System.out.print("Entrer le prenom de l'etudiant a supprimer : ");
          prénom = lectureClavier.next();
          System.out.print("Entrer le nom de l'etudiant a supprimer : ");
          nom = lectureClavier.next();
          C.supprimeUnEtudiant(prénom, nom);
      break;
      case 3 : C.afficheLesEtudiants();
      break;
      case 4 :
          System.out.print("Entrer le prenom de l'etudiant recherche : ");
          prénom = lectureClavier.next();
          System.out.print("Entrer le nom de l'etudiant recherche : ");
          nom = lectureClavier.next();
          C.rechercheUnEtudiant(prénom, nom);
      break;
      case 5 :
         // d. Ajout de la nouvelle option
          System.out.print("Entrer le prenom de l'etudiant a modifier : ");
          prénom = lectureClavier.next();
          System.out.print("Entrer le nom de l'etudiant a modifier : ");
          nom = lectureClavier.next();
          C.modifieUnEtudiant(prénom, nom);
      break;
      case 6 : System.exit(0) ;
      default : System.out.println("Cette option n'existe pas ");
    }
```

```
    } while (choix != 6);
  }
}
```

Exercice 10-6 et 10-8 : Gérer les erreurs

Corrigé :

La classe `ListeDeLivres`

```
import java.util.*;
import java.io.*;
 public class ListeDeLivres implements Serializable{
   // 10.6.a Définition de la propriété liste
   // (liste de <Livre> avec une clé <String>)
   private HashMap<String, Livre> liste;
   // 10.6.b Définition du constructeur de la liste
   public ListeDeLivres() {
      liste = new HashMap<String, Livre> ();
   }
   // 10.6.c Ajouter un livre à la liste
   public void ajouterUnLivre() {
      // 10.6.c Création d'un livre
      Livre nouveau = new Livre();
      // 10.6.c Calcule le code du livre
      String clé = nouveau.getCode();
      // 10.6.c Si le code n'existe pas dans la liste, ajoute le livre
      if (liste.get(clé) == null) liste.put(clé, nouveau);
      // 10.6.c Sinon affiche un message d'erreur
      else System.out.println("Ce livre a deja ete saisi !");
   }
   // 10.6.d Recherche un livre, connaissant le nom et le prénom de l'auteur,
   //sa catégorie et son numéro isbn
   public void rechercherUnLivre(String na, String pa,String c, String i) {
      // 10.6.d Crée un livre temporaire, le titre étant vide
      Livre tmp = new Livre(null, c, na, pa,i);
      // 10.6.d Calcule le code du livre
      String clé = tmp.getCode();
      System.out.println("---> " + clé);
      // 10.6.d Récupère le livre, connaissant la clé
      tmp = (Livre) liste.get(clé);
      // 10.6.d Affiche le livre, s'il existe
      if (tmp != null) tmp.afficherUnLivre();
      // 10.6.d Sinon afficher un message d'erreur
      else System.out.println(pa + " " + na + " est inconnu ! ");
```

```
    }
    // 10.6.f Supprime un livre, connaissant le nom et le prénom de l'auteur,
    //sa catégorie et son numéro isbn
    public void supprimerUnLivre(String na, String pa,String c, String i) {
       // 10.6.f Crée un livre temporaire, le titre étant vide
       Livre tmp = new Livre(null, c, na, pa,i);
       // 10.6.f Calcule le code du livre
       String clé = tmp.getCode();
       // 10.6.f Récupère le livre, connaissant la clé
       tmp = (Livre) liste.get(clé);
       if (tmp != null){
          // 10.6.f Supprime le livre, s'il existe
          liste.remove(clé);
          System.out.println(pa + " " + na + " a ete supprime ");
       }
       // 10.6.f Sinon affiche un message d'erreur
       else System.out.println(pa + " " + na + " est inconnu ! ");
    }
    // 10.6.f Affiche la liste des livres par l'intermédiaire d'une collection
    // de <Livre>
    public void afficherLesLivres()  {
       if(liste.size() != 0) {
          Collection<Livre> c = liste.values();
          for (Livre tmp : c)tmp.afficherUnLivre();
       }
       else System.out.println("Il n'y a pas de livre dans cette liste");
    }
}
```

La classe `Bibliothèque`

```
import java.util.*;
public class Bibliotheque {
   public static void main(String [] arg){
      Scanner lectureClavier = new Scanner(System.in);
      byte choix = 0 ;
      ListeDeLivres LdL = new ListeDeLivres();
   FichierDeLivres F = new FichierDeLivres();
      // 10.8.c. Ouvrir le fichier en lecture
      if (F.ouvrir("L")) {
         // 10.8.c Si le fichier existe, lire les lignes et stocker les formes dans
         //la liste LdF
          LdL = F.lire();
          F.fermer();
```

```
        }
        String prénom, nom, categorie, isbn;
        do {
           System.out.println("1. Ajouter un livre");
           System.out.println("2. Supprimer un livre");
           System.out.println("3. Afficher la liste des livres");
           System.out.println("4. Afficher un livre");
           System.out.println("5. Sortir");
           System.out.println();
           System.out.print("Votre choix : ");
           choix= lectureClavier.nextByte();
           switch (choix) {
              case 1 : LdL.ajouterUnLivre();
              break;
              case 2 :
              // 10.6.g Pour supprimer un livre
              System.out.print("Entrer le prenom de l'auteur du livre a supprimer : ");
              prénom = lectureClavier.next();
              System.out.print("Entrer le nom de l'auteur du livre a supprimer : ");
              nom = lectureClavier.next();
              System.out.print("Entrer la categorie du livre a supprimer : ");
              categorie = lectureClavier.next();
              System.out.print("Entrer les deux derniers chiffres du code ISBN : ");
              isbn = lectureClavier.next();
              LdL.supprimerUnLivre(nom, prénom, categorie, isbn);
              break;
              // 10.6.g Pour afficher la liste des livres
              case 3 : LdL.afficherLesLivres();
              break;
              case 4 :
              // 10.6.g Pour rechercher et afficher un seul livre
              System.out.print("Entrer le prenom de l'auteur du livre recherché : ");
              prénom = lectureClavier.next();
              System.out.print("Entrer le nom de l'auteur du livre recherché : ");
              nom = lectureClavier.next();
              System.out.print("Entrer la categorie du livre recherché : ");
              categorie = lectureClavier.next();
              System.out.print("Entrer les deux derniers chiffres du code ISBN : ");
              isbn = lectureClavier.next();
              LdL.rechercherUnLivre(nom, prénom, categorie, isbn);
              break;
              case 5 :
```

```
            // 10.8.d Pour enregistrer les données dans un fichier d'objets
            System.out.println("Sauvegarde des données dans Bibliotheque.dat");
            F.ouvrir("Ecriture");
            F.ecrire(LdL);
            F.fermer();
            System.exit(0) ;
            default : System.out.println("Cette option n'existe pas ");
         }
      } while ( choix != 5);
   }
}
```

Les méthode `lire()` et `ecrire()` de la classe `FichierDeLivres`

```
// 10.8.a Pour extraire du fichier Bibliotheque.dat des données de type ListeDeLivres
public ListeDeLivres lire() {
   try {
      ListeDeLivres tmp = (ListeDeLivres) fRo.readObject();
   return tmp;
   }
   catch (IOException e) {
      System.out.println(nomDuFichier+" : Erreurde lecture ");
   }
   catch (ClassNotFoundException e) {
      System.out.println(nomDuFichier+" n'est pas du bon format ");
   }
   return null;
}
// 10.8.a Pour enregistrer des données de type ListeDeLivres
public void ecrire(ListeDeLivres tmp) {
   try {
      if (tmp != null)fWo.writeObject(tmp);
   }
   catch (IOException e) {
   System.out.println(nomDuFichier+" : Erreur en cours d'écriture "+e);
   }
}
```

Exercice 10-9: Gérer les erreurs

Corrigé :

a. Les blocs `catch` et `try` s'écrivent de la façon suivante :

```
public void fermer() {
   try {
      if (mode == 'R' || mode == 'L') fRo.close();
```

```
        else  if (mode == 'W' || mode == 'E')fWo.close();
    }
    catch (IOException e){
        System.out.println(nomDuFichier+" : Erreur a la fermeture ");
    }
}
public void ecrire(Classe tmp) {
    try {
        if (tmp != null)fWo.writeObject(tmp);
    }
    catch (IOException e) {
        System.out.println(nomDuFichier+" : Erreur en cours d'ecriture ");
    }
}
```

b. L'entête de la fonction `main()` définie dans la classe `GestionClasse` s'écrit à nouveau comme suit :

```
public static void main (String [] argument)
```

Exercice 10-10: Gérer les erreurs

Corrigé :

```
import java.io.*;
public class Exercice10_8 {
    // La close throws est supprimée de l'entête de la fonction main()
    public static void main(String[] args) {
        String encodage = System.getProperty("file.encoding");
        System.out.println("Encodage par défaut : " + encodage);
        String proverbe = "Qui s\u00E8me le vent, r\u00E9colte la temp\u00EAte" ;
        try {
            // a. Les instructions permettant l'encodage par défaut sont
            // placées dans le bloc try
            String proverbeEncode = new String ( proverbe.getBytes(), encodage );
            System.out.println("Proverbe encode : " + proverbeEncode );
        } catch (UnsupportedEncodingException e) {
            // b. Le bloc catch capture l'exception UnsupportedEncodingException
            // et affiche un message indquant que l'encodage n'est pas supporté
            System.out.println("L'encodage " + encodage + " n'est pas supporté par
                                l'interpréteur Java");
        }
    }
}
```

Chapitre 11 : Dessiner des objets

Exercice 11-1 : Comprendre les techniques d'affichage graphique

Corrigé :

a. Il s'agit de la classe `Dessin`, qui affiche de nouveaux sapins à chaque clic sur le bouton nouveau.

b. La nouvelle classe `Dessin`

```
import java.awt.*;
public class Dessin extends Canvas {
 private Color couleur = Color.green;
 public final static Color couleurFond = Color.white;
 private Arbre A;
 public Dessin()  {
   setBackground(couleurFond);
   setForeground(couleur);
   setCursor(new Cursor(Cursor.CROSSHAIR_CURSOR));
   A = new Arbre(8);
 }
 public void paint (Graphics g){
   A.dessine(g);
 }
 public void nouveau (){
   A = new Arbre((int ) (10*Math.random()+3));
   repaint();
 }
}
```

Exercice 11-2 : Comprendre les techniques d'affichage graphique

Corrigé :

a. et c. Afficher un sapin sans aucune décoration :

```
import java.awt.*;
class Arbre {
   int [][] sapin ;
   public Arbre(int nl) {
      int nc = 2*nl-1;
      sapin = new int[nl][nc];
      int milieu = sapin[0].length/2;
      for ( int j = 0 ; j < nl ; j++) {
         for ( int i = -j; i <= j; i++) {
            sapin[j][milieu+i] = (int ) (5*Math.random()+1);
         }
      }
   }
```

```
   public void dessine(Graphics g) {
      Color Vert = Color.green;
      for (int i = 0; i < sapin.length; i++) {
         for (int j = 0; j < sapin[0].length; j++) {
            switch(sapin[i][j]) {
               // a. Affiche un sapin sans décoration
               case 1 : Vert = Color.green;
                        new Triangle(i, j, g, Vert);
               break;
               case 2 : Vert = Vert.brighter();
                        new Triangle(i, j, g, Vert);
               break;
               case 3 : Vert = Vert.darker();
                        new Triangle(i, j, g, Vert);
               break;
               case 4 : Vert = Vert.brighter();
                        new Triangle(i, j, g, Vert);
               break;
               case 5 : Vert = Vert.darker();
                        new Triangle(i, j, g, Vert);
               break;
               case 6 : Vert = Vert.brighter();
                        new Triangle(i, j, g, Vert);
               break;
            }
         }
      }
      // c. Affiche la nouvelle décoration
      for (int i = 0; i < sapin.length; i++)
      for (int j = 0; j < sapin[0].length; j++) {
         if (sapin[i][j] == 1)new Boule(i, j, g);
      }
   }
}
```

b. et d. La classe `Boule`

```
import java.awt.*;
public class Boule {
   private  int centreX = Fenetre.LG/2-50;
   private int centreY = Fenetre.HT/2-50;
private Color [] couleur = {Color.red, Color.blue, Color.yellow,
                            Color.cyan, Color.magenta};
public Boule(int col, int lig, Graphics g) {
```

```
        // d. Choisi une couleur au hasard dans le tableau couleur[]
        g.setColor(couleur[(int) (5*Math.random())]);
        // b. Affiche un cercle rempli
        g.fillOval(5 * lig + centreX,15 * col - 3 + centreY , 10, 10);
    }
}
```

Exercice 11-3 : Apprendre à gérer les événements (*placer une case à cocher*)

Corrigé :

```
import java.awt.*;
import java.awt.event.*;
public class DesBoutons extends Panel {
 public DesBoutons(Dessin d){
   setBackground(Color.lightGray);
   // a. Définit une case à cocher
   Checkbox CaseCoche = new Checkbox("Taille Fixe");
   // b. Ajoute l'écouteur d'événement
   CaseCoche.addItemListener(new GestionAction(0, d));
   // a. Ajoute la case à cocher au Panel
   this.add(CaseCoche);
   Button bPeindre = new Button ("Nouveau");
   bPeindre.addActionListener(new GestionAction(1, d));
   this.add(bPeindre);
   Button bQuitter = new Button ("Quitter");
   bQuitter.addActionListener(new GestionAction(2, d));
   this.add(bQuitter);
 }
}
```

Exercice 11-4 : Apprendre à gérer les événements (*Associer l'événement à l'action*)

Corrigé :

```
import java.awt.*;
import java.awt.event.*;
// a. La classe GestionAction implemente les 2 interfaces
public class GestionAction implements ActionListener, ItemListener {
   private int n;
   private Dessin d;
   // b. Pour être visible du bouton bNouveau, tout en étant modifié par la case à
   // cocher CaseCoche, la variable OK doit être commune aux deux objets.
   // Elle doit donc être déclarée en static.
   private static boolean OK = true;
   public GestionAction( int n, Dessin d) {
      this.n = n;
```

```
        this.d = d;
    }
    public void actionPerformed(ActionEvent e) {
        switch (n) {
            case 2 : System.exit(0);
            break;
            // c. La valeur de OK est transmise a l'objet d de type Dessin
            //voir la classe Dessin ci-dessous
            case 1 :    d.nouveau(OK);
            break;
        }
    }
    public void itemStateChanged(ItemEvent e) {
        if(e.getStateChange() == ItemEvent.SELECTED)OK = false;
        else OK = true;
    }
}
```

c. La classe `Dessin` prend en compte la valeur du booléen `OK` :

```
import java.awt.*;
public class Dessin extends Canvas {
    private Color couleur = Color.green;
    public final static Color couleurFond = Color.white;
    private Arbre A;
    public Dessin()   {
        setBackground(couleurFond);
        setForeground(couleur);
        setCursor(new Cursor(Cursor.CROSSHAIR_CURSOR));
        A = new Arbre(8);
    }
    public void paint (Graphics g){
        A.dessine(g);
    }
    public void nouveau (boolean OK){
        if (OK) A = new Arbre((int ) (10*Math.random()+3));
        else A = new Arbre(8);
        repaint();
    }
}
```

Exercice 11-5 : Apprendre à gérer les événements

Corrigé :

11-5-a :

Les classes à modifier sont les classes `DessinerDesFormes`, `DesBoutons`, `DessinFormes`, `GestionAction`.

- La classe `DessinerDesFormes` est l'application qui lance la fenêtre d'affichage où sont dessinées les formes géométriques. L'application lit les coordonnées des formes à partir du fichier `Formes.txt`.
- Dans la classe `DesBoutons` sont définis les deux boutons « A gauche / scene » et « A gauche / formes ».
- Dans la classe `DessinFormes` sont définies les méthodes d'affichage des formes, les différents modes de déplacement des formes.
- L'association entre le clic sur le bouton et l'action à réaliser est décrite dans la classe `GestionAction`.

11-5-b :

La méthode `dessinerLesFormes()` crée une fenêtre Swing dans laquelle sont placés deux composants. Un composant contenant les formes géométriques (`f.getContentPane(new DessinFormes(…), "Center")` placé au centre de la fenêtre et un composant contenant les boutons (`f.getContentPane(new DesBouton(…), "South")`, placé dans le bas de la fenêtre.

L'appel au constructeur `new DesBouton(…)` à l'intérieur de la méthode `getContentPane()` permet de créer les deux boutons.

11-5-c :

L'association entre l'action et le clic sur l'un des deux boutons est réalisée grâce à la mise en place des écouteurs d'événements (appel à la méthode `addActionListener()`). Les paramètres du constructeur `GestionAction()` sont différents pour chaque écouteur afin de différencier les comportement des boutons.

11-5-d :

Dans la classe `DesBoutons`, l'écouteur du bouton `aGaucheScene` est associé avec l'action de paramètre `1` (`new GestionAction(1, df, j)`) alors que l'écouteur du bouton `aGaucheForme` est associé avec l'action de paramètre `2` (`new GestionAction(2, df, j)`). Dans la classe `GestionAction`, la méthode `actionPerformed()` doit reprendre ces deux valeurs pour déplacer les objets :

- soit sur le bord de la scène,
- soit aligner les formes sur celle se situant le plus à gauche.

La méthode s'écrit :

```
public void actionPerformed(ActionEvent e) {
   switch (n){
      case 1 : d.deplacerGaucheScene();
      break;
      case 2 : d.deplacerGaucheFormes();
      break;
   }
 }
```

Où `n` est une variable de classe récupérée grâce au constructeur de la classe.

11-5-e : Dans la classe `DessinFormes`

La méthode `déplacer()` décrite ci-après déplace tous les objets sur le point (en `x`) placé en paramètre. Pour déplacer les objets sur le bord gauche de la scène, il suffit d'appeler la méthode avec `0` pour paramètre. Déplacer un cercle ou un triangle revient à déplacer le point de référence de l'objet défini dans la classe `Forme`. Pour déplacer un triangle, il n'en est pas de même. Il se peut que le point de référence ne soit pas celui qui se trouve le plus à gauche.

11-5-f :

Pour trouver le sommet le plus à gauche, il convient d'appeler la méthode `trouverLeXMin()` définie dans la classe `Triangle`. Comme cette méthode n'est définie que dans la classe `Triangle` et pas dans la classe `Cercle` ni dans la classe `Rectangle`, nous sommes obligés d'utiliser le mécanisme du cast.

```
public void deplacer(int Xref){
   int nbFormes = listeFormes.size();
   if (nbFormes > 0) {
      for (Forme tmp : listeFormes)
         if ( tmp instanceof Triangle) {
         // Pour être sûr de travailler sur un objet de type Triangle
         tmp = (Triangle) tmp;
         int decalageEnX = tmp.trouverLeXMin();
          tmp.deplacer(Xref - decalageEnX, 0);
      }
      else tmp.deplacer(Xref-tmp.x,0);
      }
   }
public void deplacerGaucheScene(){
      deplacer(0);
      repaint();
}
```

Dans la classe `Triangle`

```
public int trouverLeXMin(){
   int min = super.trouverLeXMin();
   for(int i=1; i < xPoints.length; i++){
      if (xPoints[i] < min) min = xPoints[i];
   }
   return min;
}
```

Dans la classe `Forme`

```
public int trouverLeXMin(){
   return x;
}
```

11-5-g :

Pour aligner les formes sur celle la plus à gauche, vous devez rechercher le point se trouvant le plus à gauche – c'est-à-dire la plus petite coordonnée en `x`. La méthode `trouverLeXMin()`, placée dans la classe `DessinFormes`, parcourt l'ensemble des formes et recherche la plus petite valeur en `x`. Un traitement spécifique est réalisé pour le triangle afin de trouver le sommet le plus à gauche.

```
public int trouverLeXMin(){
   int nbFormes = listeFormes.size();
   int min = Forme.largeurEcran ;
   if (nbFormes > 0) {
      for (Forme tmp : listeFormes) {
         if (tmp.x < min) min = tmp.x;
```

```
            if (tmp instanceof Triangle) {
               tmp = (Triangle) tmp;
               int minT = tmp.trouverLeXMin();
                if (minT < min) min = minT;
            }
         }
      }
      return min;
   }
```

Les formes sont ensuites déplacées en appelant la méthode `deplacer()` de la classe `DessinFormes` avec pour paramètre la valeur correspondant à la plus petite valeur trouvée.

```
public void deplacerGaucheFormes(){
   int Xmin = trouverLeXMin();
   deplacer(Xmin);
   repaint();
}
```

Chapitre 12 : Créer une interface graphique

Exercice 12-1 : Mise en place des éléments graphiques

Corrigé :

Voir fenêtre Design de la classe `Main` du projet `Exercice12_1`. Les fichiers associés se trouvent dans le répertoire `Sources/Exercices/Chapitre12/NetBeansProjects/Exercice12_1/src/Exercices` du CD-Rom livré avec l'ouvrage.

Exercice 12-2 : Définir le comportement des composants

Corrigé :

```
public class Main extends javax.swing.JFrame {
   // 12.2.d La propriété choixAction est initialisée à "Heure"
   private String choixAction="Metre";
   public Main() {
      initComponents();
      this.setBounds(100, 100,359, 290);
      // 12.2.a Ajouter les boutons au groupe groupeRadioBtn
      groupeRadioBtn.add(metreBtn);
      groupeRadioBtn.add(litreBtn);
      groupeRadioBtn.add(heureBtn);
      // 12.2.b Le bouton metreBtn est sélectionné par défaut
      metreBtn.setSelected(true);
   }
}
private void litreBtnActionPerformed(java.awt.event.ActionEvent evt) {
   // 12.2.c Lorsque l'utilisateur sélectionne le bouton litreBtn
```

```
    // Les textes Litres(s) et Décilitre(s) s'affichent au bon endroit
    uniteInit.setText("Litre(s)");
    uniteFinal.setText("Décilitre(s)");
    resultat.setText("0") ;
    combien.setText("0") ;
    // 12.2.d La propriété choixAction est initialisée à "Litre"
    choixAction = "Litre";
}

private void metreBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // 12.2.c Lorsque l'utilisateur sélectionne le bouton metreBtn
    // Les textes Mètre(s) et Centimètre(s) s'affichent au bon endroit
    uniteInit.setText("Mètre(s)");
    uniteFinal.setText("Centimètre(s)");
    resultat.setText("0") ;
    combien.setText("0") ;
    // 12.2.d La propriété choixAction est initialisée à "Metre"
    choixAction = "Metre";
}

private void heureBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // 12.2.c Lorsque l'utilisateur sélectionne le bouton heureBtn
    // Les textes Heure(s) et Seconde(s) s'affichent au bon endroit
    uniteInit.setText("Heure(s)");
    uniteFinal.setText("Seconde(s)");
    resultat.setText("0") ;
     combien.setText("0") ;
    // 12.2.d La propriété choixAction est initialisée à "Heure"
    choixAction = "Heure";
}
```

Exercice 12-3 : Définir le comportement des composants

Corrigé :

```
private void calculerBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // 12.3.a Récupérer le contenu du champ de saisie combien
    float valeur = Float.parseFloat(combien.getText());
    // 12.3.b selon la valeur de choixAction, réaliser la conversion demandée
    if (choixAction.equalsIgnoreCase("Metre")){
        // 12.3.c Afficher le résultat sous forme de centimètre
        resultat.setText(convertirMetre (valeur)) ;
    }
    else if (choixAction.equals("Litre")){
        // 12.3.c Afficher le résultat sous forme de décilitre
```

```
            resultat.setText(convertirLitre(valeur));
        }
        else if (choixAction.equals("Heure")){
            // 12.3.c Afficher le résultat sous forme de seconde
            resultat.setText( convertirHeure(valeur));
        }
    }
    // 12.3.b fonction de convertion de mètre en centimètre
    private String convertirMetre(float val){
        float centimetre = val * 100;
        return Float.toString(centimetre);
    }
    // 12.3.b fonction de convertion d'heure en seconde
    private String convertirHeure(float val){
        float seconde = val * 3600;
        return Float.toString(seconde);
    }
    // 12.3.b fonction de convertion de litre en décilitre
    private String convertirLitre(float val){
        float decilitre = val * 10;
        return Float.toString(decilitre);
    }
```

Exercice 12.4 à 12.7 : Le gestionnaire d'étudiants version 2

Corrigé :

- 12.4.a et 12.4.b : Voir fenêtre Design de la classe `ClasseSwing` du projet `GestionClasseExercice`. Les fichiers associés se trouvent dans le répertoire `Sources/Exercices/Chapitre12/NetBeansProjects/GestionClasseExercice/src/Introduction` du CD-Rom livré avec l'ouvrage.
- 12.7.a et 12.7.b : Voir fenêtre Design de la classe `Message` du projet `GestionClasseExercice`. Les fichiers associés se trouvent dans le répertoire `Sources/Exercices/Chapitre12/NetBeansProjects/GestionClasseExercice/src/Introduction` du CD-Rom livré avec l'ouvrage.

Dans la classe `CursusSwing`, écrire les propriétés et les méthodes suivantes :

```
    private String urlPhoto="/Users/VotreCompte/Photos/Inconnu.jpg";
    private String periode ;
    private String [] listeMatieres;
    // 12.6.a Gestion du survol de la JComboBox, déclaration et initialisation du drapeau
    private boolean etatComboBox = false;
    // 12.4.d La propriété choixAction est initialisée à "Creer"
    private String choixAction="Creer";
    public ClasseSwing() {
        initComponents();
        this.setBounds(100, 100, 450, 500);
```

```
    // 12.4.c Ajouter les boutons au groupe groupeRadioBtn
    groupeBtn.add(creerRadioBtn);
    groupeBtn.add(modifierRadioBtn);
    groupeBtn.add(supprimerRadioBtn);
    // 12.4.c Le bouton creerRadioBtn est sélectionné par défaut
    creerRadioBtn.setSelected(true);
    info.setText("Créer : Tous les champs doivent être renseignés !");
}
private void creerRadioBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // 12.4.e La propriété choixAction est initialisée à "Creer"
    choixAction = "Creer";
}
private void modifierRadioBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // 12.4.e La propriété choixAction est initialisée à "Modifier"
    choixAction = "Modifier";
}
private void supprimerRadioBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // 12.4.e La propriété choixAction est initialisée à "Supprimer"
    choixAction = "Supprimer";
}
private void validerBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // 12.4.f Tester l'action à réaliser
    String nom = nomAsaisir.getText();
    String prenom = prenomAsaisir.getText();
    if (choixAction.equalsIgnoreCase("Creer")){
        creerUnEtudiant (nom, prenom) ;
     }
    else if (choixAction.equals("Modifier")){
        modifierUnEtudiant(nom, prenom);
    }
    else if (choixAction.equals("Supprimer")){
        supprimerUnEtudiant(nom, prenom);
    }
}
// 12.4.f fonction creerUnEtudiant()
private void creerUnEtudiant(String nn, String np){
    double [] moyenne = new double[listeMatieres.length];
    for (int i=0; i < moyenne.length; i++) moyenne[i]=0;
    Etudiant eleve = new Etudiant(nn, np, urlPhoto, listeMatieres, periode, moyenne);
    BulletinNotes bn = new BulletinNotes(eleve);
}
// 12.4.f fonction supprimerUnEtudiant()
```

```
private void supprimerUnEtudiant(String nn, String np){
   // Lire la liste des étudiants dans le fichier objet
   Cursus C = new Cursus();
   FichierEtudiant F = new FichierEtudiant();
   F.ouvrir("R");
   C = F.lire();
   F.fermer();
   // C.afficheLesEtudiants();
   Message msg;
   String text = nn+ " " + np;
   // Supprimer l'étudiant selon son nom et son prénom
   if( C.supprimeUnEtudiant(nn, np)) {
      // 12.7.e Si l'étudiant est supprimé l'indiquer par une boite de dialogue
      msg = new Message("L'etudiant(e) " ,text," a été supprimé(e) de la classe ");
   }
   // 12.7.e Si l'étudiant n'est pas supprimé l'indiquer par une boîte de dialogue
   else msg = new Message("L'etudiant(e) " ,text, "est inconnu(e) !");
   // Enregistrer la suppression
   F.ouvrir("W");
   F.ecrire(C);
   F.fermer();
}
private void modifierUnEtudiant(String nn, String np){
   // Lire la liste des étudiants dans le fichier objet
   Cursus C = new Cursus();
   FichierEtudiant F = new FichierEtudiant();
   F.ouvrir("R");
   C = F.lire();
   F.fermer();
   // C.afficheLesEtudiants();
   // Rechercher l'étudiant par son nom et son prénom
   Etudiant eleve = C.rechercheUnEtudiant(nn, np) ;
   if( eleve != null) {
      // Si l'étudiant existe, afficher son bulletin de notes
      BulletinNotes bn = new BulletinNotes(eleve);
      // Supprimer l'étdudiant de la liste et enregistrer cette supression
      C.supprimeUnEtudiant(nn, np);
      F.ouvrir("W");
      F.ecrire(C);
      F.fermer();
      eleve=null;
      // L'étudiant sera à nouveau enregistré par l'intermédiaire
```

```
            // du bulletin de notes
        }
        else {
            // 12.7.e Si l'étudiant n'existe pas l'indiquer par une boîte de dialogue
            Message msg;
            String text =nn+ " " + np;
            msg = new Message("L'etudiant(e) " ,text, "est inconnu(e) !");
        }
    }
    private void fermerBtnActionPerformed(java.awt.event.ActionEvent evt) {
        // Cesser l'exécution de l'application
        System.exit(0);
    }
    private void modifierRadioBtnMouseEntered(java.awt.event.MouseEvent evt) {
        // 12.5 Aide contextuelle
        // 12.6.c Gestion du survol de la JComboBox
        if(etatComboBox){
            info.setText("Info : Choisir la période de validation des notes");
        }
        else
        info.setText("Modifier un étudiant, vous devez connaître son nom et son prénom");
    }
    private void nomAsaisirMouseEntered(java.awt.event.MouseEvent evt) {
        // 12.5 Aide contextuelle
        // 12.6.c Gestion du survol de la JComboBox
        if(etatComboBox){
            info.setText("Info : Choisir la période de validation des notes");
        }
        else info.setText("Info : Entrer le nom de l'étudiant ");
    }
    private void prenomAsaisirMouseEntered(java.awt.event.MouseEvent evt) {
        // 12.5 Aide contextuelle
        // 12.6.c Gestion du survol de la JComboBox
        if(etatComboBox){
            info.setText("Info : Choisir la période de validation des notes");
        }
        else info.setText("Info : Entrer le prénom de l'étudiant ");
    }
    private void rechercherBtnMouseEntered(java.awt.event.MouseEvent evt) {
        // 12.5 Aide contextuelle
        // 12.6.c Gestion du survol de la JComboBox
        if(etatComboBox){
```

```
        info.setText("Info : Choisir la période de validation des notes");
    }
    else info.setText("Info : Choisir la photo de l'étudiant ");
}
private void validerBtnMouseEntered(java.awt.event.MouseEvent evt) {
    // 12.5 Aide contextuelle
    // 12.6.c Gestion du survol de la JComboBox
    if(etatComboBox){
        info.setText("Info : Choisir la période de validation des notes");
    }
    else info.setText("Info : Afficher le bulletin de notes de l'étudiant ");
}
private void fermerBtnMouseEntered(java.awt.event.MouseEvent evt) {
    // 12.5 Aide contextuelle
    // 12.6.c Gestion du survol de la JComboBox
    if(etatComboBox){
        info.setText("Info : Choisir la période de validation des notes");
    }
    else info.setText("Info : Quitter l'application ");
}
private void creerRadioBtnMouseEntered(java.awt.event.MouseEvent evt) {
    // 12.5 Aide contextuelle
    // 12.6.c Gestion du survol de la JComboBox
    if(etatComboBox){
        info.setText("Info : Choisir la période de validation des notes");
    }
    else
        info.setText("Info : Tous les champs doivent être renseignés !");
}
private void supprimerRadioBtnMouseEntered(java.awt.event.MouseEvent evt) {
    // 12.5 Aide contextuelle
    // 12.6.c Gestion du survol de la JComboBox
    if(etatComboBox){
        info.setText("Info : Choisir la période de validation des notes");
    }
    else
   info.setText("Supprimer un étudiant, vous devez connaître son nom et son prénom ");
}
private void formMouseEntered(java.awt.event.MouseEvent evt) {
    // 12.5 Aide contextuelle
    // 12.6.c Gestion du survol de la JComboBox
    if(etatComboBox){
```

```
        info.setText("Info : Choisir la période de validation des notes");
    }
    else info.setText("Info : Tous les champs doivent être renseignés !");
}
private void choixPeriodePopupMenuWillBecomeInvisible(
                                        javax.swing.event.PopupMenuEvent evt) {
    // 12.6.c Gestion du survol de la JComboBox
    etatComboBox = false;
    info.setText("Info : Tous les champs doivent être renseignés !");
}
private void choixPeriodePopupMenuWillBecomeVisible(
                                        javax.swing.event.PopupMenuEvent evt) {
    // 12.6.b Gestion du survol de la JComboBox
    etatComboBox = true;
    info.setText("Info : Choisir la période de validation des notes");
}
```

Dans la classe `Message`, écrire les propriétés et les méthodes suivantes :

```
public Message(String hmsg, String cmsg, String bmsg) {
    setVisible(true);
    initComponents();
    setBounds(300, 300, 222, 322);
    // 12.7.c Affiche le contenu du premier paramètre dans le composant hautLabel
    hautLabel.setText(hmsg);
    // 12.7.c Affiche le contenu du second paramètre dans le composant centreLabel
    centreLabel.setText(cmsg);
    // 12.7.c Affiche le contenu du troisième paramètre dans le composant basLabel
    basLabel.setText(bmsg);
}
private void okBtnActionPerformed(java.awt.event.ActionEvent evt) {
    // 12.7.d Fermer la fenêtre sans quitter l'application
    this.dispose();
}
```

Exercice 12-8 : L'éditeur graphique version 2 - Dessiner un rectangle à la souris

Corrigé :

Dans la classe `FeuilleDeDessins`, insérer la fonction :

```
private void dessinerUnRectangle( ){
    // 12.8.a Dessiner un rectangle
    int decalageX = 0;
    int decalageY = 0;
    // Si le tracé de la boîte s'est effectué de la droite vers la gauche :
    if (deltaX < 0) {
```

```
        // placer l'origine de la boîte englobante au coin supérieur gauche du tracé
        decalageX = deltaX;
        // rendre positive la largeur du rectangle
        deltaX = -deltaX;
    }
    // Le tracé de la boîte s'est effectué de la gauche vers la droite,
    // il n'y a pas de décalage.
    else  decalageX = 0;
    // Si le tracé de la boîte s'est effectué du bas vers le haut :
    if (deltaY < 0) {
        // placer l'origine de la boîte englobante au coin supérieur gauche du tracé
        decalageX = deltaX;
        // rendre positive la hauteur du rectangle
        deltaY = -deltaY;
    }
    // Le tracé de la boîte s'est effectué du haut vers le bas, il n'y a pas de
    // décalage.
    else decalageY = 0;
    // Créer un rectangle avec les valeurs de saisie.
    Rectangle r = new Rectangle(debutX + decalageX, debutY+decalageY, deltaX, deltaY,
                                couleur );
    listeAdessiner.ajouterUneForme(r);
    repaint();
}
```

Dans la classe `FeuilleDeDessins`, modifier le gestionnaire `formMouseReleased()` comme suit :

```
private void formMouseReleased(java.awt.event.MouseEvent evt) {
    // Enregistrer la distance parcourue par la souris, en x et en y, entre le moment
    // du clic et celui du relâchement.
    deltaX = evt.getX()- debutX;
    deltaY = evt.getY()- debutY;
    // Si la forme est un cercle, le dessiner.
    if (forme.equals("cercle")){
        dessinerUnCercle();
    }
    // 12.8.b Si la forme est un rectangle, le dessiner.
    else if (forme.equals("rectangle")){
        dessinerUnRectangle();
    }
}
```

Exercice 12.9 : L'éditeur graphique version 2 - Effacer la dernière forme dessinée

Corrigé :

Dans la classe `ListeDeFormes`, insérez la méthode `supprimerLaDerniereForme()` suivante :

```
public void supprimerLaDerniereForme(){
   // Calculer la taille de la liste
   int index = listeFormes.size();
   if (index >0)
   // Supprimer le dernier élément de la liste
   listeFormes.remove(index-1);
}
```

Dans la classe `Main`, créez le gestionnaire d'événements associé au bouton `undo`, comme suit :

```
private void undoActionPerformed(java.awt.event.ActionEvent evt) {
   // 12.9.b Supprimer la dernière forme stockée dans la liste
   // Voir classe ListeDeFormes
   liste.supprimerLaDerniereForme();
   // Afficher la nouvelle liste
   page.dessinerLesFormes(liste);
}
```

Exercice 12.10 : L'éditeur graphique version 2 - Ouvrir sous

Corrigé :

12.10.a :Voir fenêtre Design de la classe `Main` du projet `EditeurExercice`. Les fichiers associés se trouvent dans le répertoire `Sources/Exercices/Chapitre12/NetBeansProjects/EditeurExercice/src/Introduction` du CD-Rom livré avec l'ouvrage.

Dans la classe `Main`, créez le gestionnaire d'événements `itemSauverSousActionPerformed()` comme suit :

```
private void itemSauverSousActionPerformed(java.awt.event.ActionEvent evt) {
   // 12.10 Ouvrir un fichier à l'aide d'une boîte de dialogue
   File file;
   JFileChooser fc = new JFileChooser();
   String nomDuFichier = "Formes.txt";
   // Ouvrir une boîte de dialogue pour indiquer le répertoire et le nom du fichier
   // de sauvegarde
   int returnVal = fc.showSaveDialog(Main.this);
   if (returnVal == JFileChooser.APPROVE_OPTION) {
      file = fc.getSelectedFile();
      // Récupérer le nom du fichier de sauvegarde
      nomDuFichier = file.toString() ;
   }
   Fichier f = new Fichier();
   // Ouvrir le fichier en écriture
   f.ouvrir(nomDuFichier, "W");
```

```
    // Enregistrer les données
    if (liste != null)  liste.enregistrerLesFormes(f);
    f.fermer();
  }
```

Exercice 12.11 : L'éditeur graphique version 2 - Enregistrer sous

Corrigé :

Dans la classe `Main`, modifier le gestionnaire d'événements `itemOuvrirActionPerformed()` comme suit :

```
  private void itemOuvrirActionPerformed(java.awt.event.ActionEvent evt) {
    // 12.11 Ouvrir un fichier à l'aide d'une boîte de dialogue
    File file;
    JFileChooser fc = new JFileChooser();
    String nomDuFichier = "Formes.txt" ;
    // Récupérer le nom du fichier à ouvrir
    int returnVal = fc.showOpenDialog(Main.this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
       file = fc.getSelectedFile();
       nomDuFichier = file.toString() ;
    }
    // Vider la liste de ses formes
    if (liste != null) liste.supprimerLesFormes();
    // Ouvrir le fichier recherché en lecture
    Fichier f = new Fichier();
    if (f.ouvrir(nomDuFichier, "R")) {
       // Récupérer les formes stockées dans le fichier
       // Les stocker dans la liste
       liste.lireLesFormes(f);
       f.fermer();
       // Afficher la page avec la nouvelle liste de formes
       page.dessinerLesFormes(liste);
    }
  }
```

Exercice 12.12 : L'éditeur graphique version 2 - La fenêtre À propos

Corrigé :

12.10.a : Copier la classe `Message` du projet `GestionClasseExercice`.

12.10.b : Voir fenêtre Design de la classe `Message` du projet `EditeurExercice`. Les fichiers associés se trouvent dans le répertoire `Sources/Exercices/Chapitre12/NetBeansProjects/EditeurExercice/src/Introduction` du CD-Rom livré avec l'ouvrage.

Dans la classe `Message`, modifier le gestionnaire d'événements `itemAproposActionPerformed()` comme suit :

```
  private void itemAproposActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    // 12.12 Ouvrir la fenêtre À Propos avec le bon message
    Message msg = new Message("Le livre de Java premier langage " ,"Anne Tasso",
                              "Chapitre 12, Un éditeur graphique");
}
```

Chapitre 13 : Développer une application Android

Exercice 13-1 à 13.9

Corrigé :

Retrouvez les corrigés commentés en important sous Android Studio, les projets `Exercice13_1` et `ListeDeCoursesAvecLegumes` à partir du répertoire `Sources/Exercices/Chapitre13/AndroidStudioProjects`.

Le projet : Gestion d'un compte bancaire

Chapitre 1 : Stocker une information

Déterminer les variables nécessaires au programme

Corrigé

Compte tenu des informations fournies en énoncé, les variables utilisées pour le projet peuvent être déclarées de la façon suivante :

```
public class ProjetCh1 {
 public static void main (String [] argument) {
      byte choix;
      char typeCpte ='\0';
      double val_courante = 0.0, taux = 0.0;
      long numéroCpte = 0, numéroLu = 0 ;
 }
}
```

Chapitre 2 : Communiquer une information

Afficher le menu principal ainsi que ses options

Corrigé

Les instructions d'affichage et de saisie des données s'écrivent :

```
import java.util.*;
public class ProjetCh2 {
 public static void main (String [] argument) {
      byte choix;
      char typeCpte ='\0';
      double val_courante = 0.0, taux = 0.0;
      long numéroCpte = 0, numéroLu = 0 ;
      Scanner lectureClavier = new Scanner(System.in);
      // Afficher le menu principal
```

```
    System.out.println("1. Creation d'un compte");
    System.out.println("2. Affichage d'un compte");
    System.out.println("3. Ecrire une ligne comptable");
    System.out.println("4. Sortir");
    System.out.println("5. De l'aide");
    System.out.println();
    System.out.print("Votre choix : ");
    choix = lectureClavier.nextByte();
    //Option 1 : Créer un compte
    System.out.print("Type du compte [Types possibles : ");
    System.out.print("C(ourant), J(oint), E(pargne)] :");
    typeCpte = lectureClavier.next().charAt(0);
    System.out.print("Numero du compte :");
    numéroCpte = lectureClavier.nextLong();
    System.out.print("Premiere valeur creditee:");
    val_courante = lectureClavier.nextDouble();
    // Si compte épargne
    System.out.print("Taux de placement : ");
    taux = lectureClavier.nextDouble();
    // Option 2 : Afficher un compte
    //Demande à l'utilisateur de saisir le numéro du compte à afficher
    System.out.print ( " Quel compte souhaitez vous afficher ? : ");
    numéroLu = lectureClavier.nextLong();
    // Si le numéro du compte existe,
    System.out.print("Le compte n° : " + numéroCpte + " est un compte ");
    // affiche son taux dans le cas d'un compte épargne.
    System.out.println("epargnedont le taux est" + taux);
    System.out.println("Premiere valeur creditee: " + val_courante);
    // Sinon, il affiche un message indiquant que le numéro du compte n'est pas
    // valide.
    System.out.println("Le systeme ne connait pas le compte " + numéroLu);
    // Option 3 : le programme affiche "option non programmée"
    System.out.println("Option non programmee");
    // Option 4,le programme termine son exécution
  System.out.println("Au revoir et a bientot");
    // System.exit(0) ;
    // Option 5
    // le programme affiche une ligne d'explication pour chaque option du menu
    System.out.println("Option 1. Pour creer un compte Courant entrer C ");
    System.out.println("Pour creer un compte Joint entrer J ");
    System.out.println("Pour creer un compte Epargne entrer E");
    System.out.print("Puis, entrer le numero du compte, et");
```

```
        System.out.println(" sa premiere valeur creditee ");
        System.out.println("Dans le cas d'un compte epargne, entrer le taux ");
        System.out.println("Option 2. Le systeme affiche les donnees du compte choisi ");
        System.out.println("Option 3. Ecrire une ligne comptable");
        System.out.println("Option 4. Pour quitter le programme");
        System.out.println("Option 5. Pour afficher de l'aide"); }
 }
}
```

Chapitre 3 : Faire des choix

Accéder à un menu suivant l'option choisie

Corrigé

a. Sachant que les options du menu ont toute une probabilité voisine d'être choisies, la structure de test la plus appropriée est le `switch`.

b. Le programme s'écrit comme suit :

```
import java.util.*;
public class ProjetCh3  {
 public static void main (String [] argument) {
      byte choix;
      char typeCpte ='\0';
      double val_courante = 0.0, taux = 0.0;
      long numéroCpte = 0, numéroLu = 0 ;
      Scanner lectureClavier = new Scanner(System.in);
      System.out.println("1. Creation d'un compte");
      System.out.println("2. Affichage d'un compte");
      System.out.println("3. Ecrire une ligne comptable");
      System.out.println("4. Sortir");
      System.out.println("5. De l'aide");
      System.out.println();
      System.out.print("Votre choix :");
      choix = lectureClavier.nextByte();
      // Suivant l'option choisie
      switch (choix){
         // Si l'option vaut 1
         case 1 :
            System.out.print("Type du compte [Types possibles :" );
            System.out.print("C(ourant), J(oint), E(pargne)] :");
            typeCpte = lectureClavier.next().charAt(0);
            System.out.print("Numero du compte :");
            numéroCpte = lectureClavier.nextLong();
            System.out.print("Premiere valeur creditee :");
            val_courante = lectureClavier.nextDouble();
```

```
            // c. Si le type est un compte epargne, le programme demande automatiquement
            // le taux
            if ( typeCpte == 'E') {
               System.out.print("Taux de placement : ");
               taux = lectureClavier.nextDouble();
             }
         break;
         // Si l'option vaut 2
         case 2 :
            // demande à l'utilisateur de saisir le numéro du compte à afficher
            System.out.print ( " Quel compte souhaitez vous afficher ? : ");
            numéroLu = lectureClavier.nextLong();
             // vérifie que le numéro du compte existe,
            if ( numéroLu == numéroCpte) {
               // d. Si oui, affiche le numéro du compte, le type, la valeur initiale
               System.out.print("Le compte n° : " + numéroCpte + " est un compte ");
               if (typeCpte == 'C') System.out.println(" courant");
               else if (typeCpte == 'J') System.out.println(" joint");
               else if (typeCpte == 'E'){
                  // c. affiche son taux dans le cas d'un compte épargne.
                  System.out.println("epargnedont le taux est" + taux);
               }
               System.out.println(" Premiere valeur creditee : " + val_courante);
            }
         else{
            // d. Sinon, affiche un message indiquant que le numéro du compte est non
            // valide.
            System.out.println("Le systeme ne connait pas le compte " + numéroLu);
            }
         break;
         // Si l'option vaut 3
         case 3 :
            System.out.println("Option non programmee");
         break;
            // Si l'option vaut 4
         case 4 :
            System.out.println("Au revoir et a bientot");
            System.exit(0) ;
         break;
         // Si l'option vaut 5
         case 5 :
            // le programme affiche une ligne d'explication pour chaque option.
```

```
            System.out.println("Option 1. Pour creer un compte Courant entrer C ");
            System.out.println("Pour creer un compte Joint entrer J ");
            System.out.println("Pour creer un compte Epargne entrer E");
            System.out.print("Puis, entrer le numero du compte, et");
            System.out.println(" sa premiere valeur creditee ");
            System.out.println("Dans le cas d'un compte epargne, entrer le taux ");
            System.out.println("Option 2. Le systeme affiche les donnees du compte
choisi ");
            System.out.println("Option 3. Ecrire une ligne comptable");
            System.out.println("Option 4. Pour quitter le programme");
            System.out.println("Option 5. Pour afficher de l'aide");
         }
      }
}
```

Chapitre 4 : Faire des répétitions

Rendre le menu interactif

Corrigé

a. Tant que l'utilisateur ne choisit pas l'option `Sortir`, répéter l'affichage du menu et la structure `switch`.

b. La boucle `do…while`, est la structure la plus appropriée puisqu'elle permet d'entrer et d'exécuter au moins une fois la boucle avant la saisie du choix de l'utilisateur. La boucle s'écrit :

```
do {
   // programme
   choix = lectureClavier.nextByte();
while (choix != 4);
```

La valeur `4` étant celle associée à l'option `Sortir` .

c. La traduction en Java, de cette marche à suivre s'écrit :

```
import java.util.*;
public class ProjetCh4  {
 public static void main (String [] argument)    {
      byte choix;
      char typeCpte ='\0';
      double val_courante = 0.0, taux = 0.0;
      long numéroCpte = 0, numéroLu = 0 ;
      Scanner lectureClavier = new Scanner(System.in);
      do{
         System.out.println("1. Creation d'un compte");
         System.out.println("2. Affichage d'un compte");
         System.out.println("3. Ecrire une ligne comptable");
         System.out.println("4. Sortir");
         System.out.println("5. De l'aide");
```

```
System.out.println();
System.out.print("Votre choix : ");
choix = lectureClavier.nextByte();
switch (choix){
   case 1 :
   // Tant que le caractère saisi n'est pas C, E ou J, le programme demande
   // à nouveau la saisie. Le type ainsi saisi correspond aux types proposés.
      do {
      System.out.print("Type du compte [Types possibles :" );
      System.out.print("C(ourant), J(oint), E(pargne)] :");
      typeCpte = lectureClavier.next().charAt(0);
   } while ( typeCpte != 'C' && typeCpte != 'J' && typeCpte != 'E');
   System.out.print("Numero du compte : ");
   numéroCpte = lectureClavier.nextLong();
   System.out.print("Premiere valeur creditee : ");
   val_courante = lectureClavier.nextDouble();
   if ( typeCpte == 'E') {
       System.out.println("Taux de placement : ");
       taux = lectureClavier.nextDouble();
   }
   break;
   case 2 :
      System.out.print ( " Quel compte souhaitez vous afficher ? : ");
      numéroLu = lectureClavier.nextLong();
      if ( numéroLu == numéroCpte) {
         System.out.print("Le compte n° : " + numéroCpte + " est un compte ");
         if (typeCpte == 'C') System.out.println(" courant");
         else if (typeCpte == 'J') System.out.println(" joint");
         else if (typeCpte == 'E')
         System.out.println(" epargnedont le taux est" + taux);
         System.out.println(" Valeur initiale : " + val_courante);
   }
   else
      System.out.println("Le systeme ne connait pas le compte " + numéroLu);
   break;
   case 3 :
      System.out.println("Option non programmee");
   break;
   case 4 :
      System.out.println("Au revoir et a bientot");
      System.exit(0) ;
   break;
```

```
            case 5 :
               System.out.println("Option 1. Pour creer un compte Courant entrer C ");
               System.out.println("Pour creer un compte Joint entrer J ");
               System.out.println("Pour creer un compte Epargne entrer E");
               System.out.print("Puis, entrer le numero du compte, et");
               System.out.println(" sa premiere valeur creditee ");
               System.out.println(" Dans le cas d'un compte epargne, entrer le taux ");
               System.out.println("Option 2. Le systeme affiche les donnees du compte
                                   choisi ");
               System.out.println("Option 3. Ecrire une ligne comptable");
               System.out.println("Option 4. Pour quitter le programme");
               System.out.println("Option 5. Pour afficher de l'aide");
            break;
            default :
                System.out.println("Cette option n'existe pas ");
            }
         } while (choix != 4);
      }
}
```

Chapitre 5 : De l'algorithme paramètré à l'écriture de fonctions

Définir une fonction (*Les fonctions sans paramètre avec résultat*)

Corrigé :

a. L'entête de la fonction `menuPrincipal()` s'écrit :

```
public static byte menuPrincipal()
```

En effet, aucun paramètre n'est nécessaire à la bonne marche de la fonction. Les deux parenthèses, sans aucune variable à l'intérieur, suffisent. Par contre la fonction communique le choix de l'utilisateur à la fonction `main()`. Cette valeur étant de type `byte`, la fonction est définie également de type `byte`.

b. Le corps de la fonction reprend les instructions d'affichage utilisées dans le programme développé au cours des chapitres précédents.

```
byte tmp;
Scanner lectureClavier = new Scanner(System.in);
System.out.println("1. Creation d'un compte");
System.out.println("2. Affichage d'un compte");
System.out.println("3. Ecrire une ligne comptable");
System.out.println("4. Sortir");
System.out.println("5. De l'aide");
System.out.println();
System.out.println("Votre choix : ");
tmp= lectureClavier.nextByte();
```

La variable `choix` est remplacée volontairement par la variable `tmp`, pour bien les différencier. Une variable `choix` déclarée à la fois dans la fonction `main()` et dans la fonction `menuPrincipal()` utilisent deux cases mémoires distinctes ! Elles ne représentent pas la même variable.

c. Pour communiquer la valeur correspondant au choix de l'utilisateur, la variable `tmp` est placée dans une instruction `return`, comme suit :

```
return tmp;
```

Définir une fonction (*Les fonctions sans paramètre ni résultat*)

Corrigé

a. La fonction `sortir()` ne fournit pas de résultat, elle est donc déclarée de type `void`. L'entête s'écrit :

```
public static void sortir( )
```

b. Le corps de la fonction `sortir()` reprend les instructions de sortie de programme utilisées dans le programme développé au cours des chapitres précédents.

```
System.out.println("Au revoir et a bientot");
System.exit(0) ;
```

c. Tout comme la fonction `sortir()`, la fonction `alAide()` ne fournit pas de résultat, l'entête s'écrit :

```
public static void alAide( )
```

d. Le corps de la fonction `alAide()` reprend les instructions d'affichage utilisées dans le programme développé au cours des chapitres précédents.

```
System.out.println("Option 1. Pour creer un compte Courant entrer C ");
System.out.println("Pour creer un compte Joint entrer J ");
System.out.println("Pour creer un compte Epargne entrer E");
System.out.print("Puis, entrer le numero du compte, et");
System.out.println(" sa premiere valeur creditee ");
System.out.println(" Dans le cas d'un compte epargne, entrer le taux ");
System.out.println("Option 2. Le systeme affiche les donnees du compte choisi ");
System.out.println("Option 3. Ecrire une ligne comptable");
System.out.println("Option 4. Pour quitter le programme");
System.out.println("Option 5. Pour afficher de l'aide");
```

Appeler une fonction

Corrigé

La fonction `main()` fait appel aux fonctions `alAide()`, `sortir()` et `menuPrincipal()`, de la façon suivante :

```
public static void main (String [] argument) {
 byte choix;
 char typeCpte ='\0';
 double val_courante = 0.0, taux = 0.0;
 long numéroCpte = 0, numéroLu = 0 ;
 Scanner lectureClavier = new Scanner(System.in);
 do{
   choix = menuPrincipal();
   switch (choix) {
      case 1 :
      do {
```

```
            System.out.print("Type du compte [Types possibles :" );
            System.out.print("C(ourant), J(oint), E(pargne)] :");
            typeCpte = lectureClavier.next().charAt(0);
         } while ( typeCpte != 'C' && typeCpte != 'J' && typeCpte != 'E');
         System.out.print("Numero du compte :");
         numéroCpte = lectureClavier.nextLong();
         System.out.print("Premiere valeur creditee :");
         val_courante = lectureClavier.nextDouble();
         if ( typeCpte == 'E'){
         System.out.print("Taux de placement : ");
         taux = lectureClavier.nextDouble();
         }
         break;
         case 2 :
         System.out.print ( " Quel compte souhaitez vous afficher ? : ");
         numéroLu = lectureClavier.nextLong();
         if ( numéroLu == numéroCpte){
            System.out.print("Le compte n° : " + numéroCpte + " est un compte ");
            if (typeCpte == 'C') System.out.println(" courant");
            else if (typeCpte == 'J') System.out.println(" joint");
            else if (typeCpte == 'E')
               System.out.println(" epargnedont le taux est" + taux);
            System.out.println(" Valeur initiale : " + val_courante);
         }
         else
            System.out.println("Le systeme ne connait pas le compte " + numéroLu);
         break;
         case 3 :System.out.println("Option non programmee");
         break;
         case 4 :       sortir();
         break;
         case 5 :       alAide();
         break;
         default :System.out.println("Cette option n'existe pas ");
      }
    } while (choix != 4);
  }
```

Chapitre 6 : Fonctions, notions avancées

Comprendre la visibilité des variables (*Les variables locales*)

Corrigé

Aucune valeur n'est affichée par ce programme, le compilateur détectant des erreurs du type :

```
ProjetCh6.java:86: Variable numéroCpte may not have been initialized.
ProjetCh6.java:87: Variable typeCpte may not have been initialized.
ProjetCh6.java:92: Variable taux may not have been initialized.
ProjetCh6.java:94: Variable val_courante may not have been initialized.
```

En effet, les variables `numéroCpte`, `typeCpte`, `taux` et `val_courante` sont déclarées à l'intérieur de la fonction `afficherCpte()`. Ce sont donc des variables locales. Les valeurs étant saisies dans la fonction `créerCpte()`, aucune valeur n'est placée dans les cases mémoire respectives de la fonction `afficherCpte()`. L'interpréteur ne peut pas réaliser d'affichage.

Comprendre la visibilité des variables (*Les variables de classe*)

Corrigé

```
import java.util.*;
public class ProjetCh6 { // Variables de classe
 // a. Les variables caractérisant un compte sont déclarées comme variables de classe
 static byte choix;
 static char typeCpte ='\0';
 static double val_courante = 0.0, taux = 0.0;
 static long numéroCpte = 0, numéroLu = 0 ;
 public static void main (String [] argument) {
// b. Aucune variable n'est déclarée localement à cette fonction
Scanner lectureClavier = new Scanner(System.in);
do {
   choix = menuPrincipal();
   switch (choix){
      case 1 :
         do {
         System.out.print("Type du compte [Types possibles :" );
         System.out.print("C(ourant), J(oint), E(pargne)] :");
         typeCpte = lectureClavier.next().charAt(0);
         } while ( typeCpte != 'C' && typeCpte != 'J' && typeCpte != 'E');
      System.out.print("Numero du compte :");
      numéroCpte = lectureClavier.nextLong();
      System.out.print("Premiere valeur creditee :");
      val_courante= lectureClavier.nextDouble();
      if (typeCpte == 'E') {
         System.out.print("Taux de placement :");
         taux = lectureClavier.nextDouble();
      }
      break;
      case 2 :
         System.out.print ( " Quel compte souhaitez vous afficher ? : ");
         numéroLu = lectureClavier.nextLong();
```

```
            if (numéroLu == numéroCpte)afficherCpte( );
            else
             System.out.println("Le systeme ne connait pas le compte " + numéroLu);
        break;
        case 3 : System.out.println("Option non programmee");
        break;
        case 4 : sortir();
        break;
        case 5 : alAide();
        break;
        default : System.out.println("Cette option n'existe pas ");
      }
    } while (choix != 4);
  }

   // Affiche le compte
   public static void afficherCpte( ){
  // b. Aucune variable n'est déclarée localement à cette fonction
     System.out.print("Le compte n° : " + numéroCpte + " est un compte ");
     if (typeCpte == 'C') System.out.println(" courant");
     else if (typeCpte == 'J') System.out.println(" joint");
     else if (typeCpte == 'E')
         System.out.println(" epargnedont le taux est" + taux);
     System.out.println("Premiere valeur creditee: " + val_courante);
   }
  // Affiche le menu principal, retourne la valeur de l'option choisie
   public static byte menuPrincipal() {
     byte tmp;
     Scanner lectureClavier = new Scanner(System.in);
     System.out.println("1. Creation d'un compte");
     System.out.println("2. Affichage d'un compte");
     System.out.println("3. Ecrire une ligne comptable");
     System.out.println("4. Sortir");
     System.out.println("5. De l'aide");
     System.out.println();
     System.out.println("Votre choix : ");
     tmp= lectureClavier.nextByte();
     return tmp;
   }
   public static void sortir( ) {
     System.out.println("Au revoir et a bientot");
     System.exit(0) ;
```

```
  }
  public static void alAide( ) {
    System.out.println("Option 1. Pour creer un compte Courant entrer C ");
    System.out.println("Pour creer un compte Joint entrer J ");
    System.out.println("Pour creer un compte Epargne entrer E");
    System.out.print("Puis, entrer le numero du compte, et");
    System.out.println(" sa premiere valeur creditee ");
    System.out.println("Dans le cas d'un compte epargne, entrer le taux ");
    System.out.println("Option 2. Le systeme affiche les donnees du compte choisi ");
    System.out.println("Option 3. Ecrire une ligne comptable");
    System.out.println("Option 4. Pour quitter le programme");
    System.out.println("Option 5. Pour afficher de l'aide");
  }
}
```

Comprendre la visibilité des variables (*Le passage de paramètres par valeur*)

Corrigé

a. L'entête de la fonction utilise quatre paramètres correspondant aux caractéristiques du compte à transmettre à la fonction. Elle s'écrit :

```
public static void afficherCpte( long num, char type, double taux, double val) {
```

b. Le corps de la fonction `afficherCpte()` reprend les instructions utilisées dans le programme développé au cours des chapitre précédents, en prenant soin de remplacer les noms de variables par ceux définis en paramètre, dans l'entête de la fonction.

```
System.out.print("Le compte n° : " + num + " est un compte ");
if (type == 'C') System.out.println(" courant");
else if (type == 'J') System.out.println(" joint");
else if (type == 'E')
System.out.println(" epargnedont le taux est" + taux);
System.out.println("Premiere valeur creditee : " + val);
```

Les limites du retour de résultat

Corrigé

a. Pour créer un compte, les valeurs saisies par l'utilisateur sont au nombre de quatre (le type, le numéro, le taux et la valeur courante). La fonction `créerCpte()` doit donc retourner ces quatre valeurs à la fonction `main()`.

b. Les quatre valeurs sont chacune de type différent (`long`, `double` ou `char`). Il n'est donc pas possible de déterminer le type de la fonction `créerCpte()`. De plus, l'instruction `return` ne peut retourner qu'une seule et unique variable. Il n'est donc pas possible de transmettre le contenu des quatre variables à la fonction `main()`.

Chapitre 7 : Les classes et les objets

Traiter les chaînes de caractères

Corrigé

a. et b. Compte tenu des changements de type des variables `typeCpte` et `numéroCpte`, la saisie des valeurs s'écrit de la façon suivante :

```
public String typeCpte ;
public double taux ;
public String numéroCpte ;
do {
      System.out.print("Type du compte [Types possibles :" );
      System.out.print("C(ourant), J(oint), E(pargne)] :");
      tmp = lectureClavier.next().charAt(0);
} while ( tmp != 'C' && tmp!= 'J' && tmp != 'E');
switch (tmp)   {
         case 'C' : typeCpte = "Courant";
         break;
         case 'J' : typeCpte = "Joint";
         break;
         case 'E' : typeCpte = "Epargne";
         break;
      }
System.out.print("Numero du compte : ");
numéroCpte = lectureClavier.next();
if (typeCpte.equalsIgnoreCase("Epargne")) {
   System.out.print("Taux de placement : ");
   taux = lectureClavier.nextDouble();
}
```

c. Voir corrigé de "Construire l'application `Projet`" ci-après.

Définir le type `Compte`

Corrigé

a. Les données définissant tout compte bancaire sont les suivantes :

```
public String typeCpte ;
public double val_courante, taux ;
public String numéroCpte ;
```

b. La méthode `créerCpte()` est déclarée non `static`, puisqu'elle décrit le comportement de tout compte bancaire. Elle permet l'initialisation par saisie au clavier, des données caractéristiques d'un compte (variables d'instance). Elle s'écrit de la façon suivante :

```
public void créerCpte()    {
   char tmp;
   Scanner lectureClavier = new Scanner(System.in);
   do {
         System.out.print("Type du compte [Types possibles :" );
         System.out.print("C(ourant), J(oint), E(pargne)] :");
         tmp = lectureClavier.next().charAt(0);
   } while ( tmp != 'C' && tmp!= 'J' && tmp != 'E');
   switch (tmp)   {
         case 'C' : typeCpte = "Courant";
```

```
            break;
            case 'J' : typeCpte = "Joint";
            break;
            case 'E' : typeCpte = "Epargne";
            break;
      }
      System.out.print("Numero du compte : ");
      numéroCpte = lectureClavier.next();
      if ( typeCpte.equalsIgnoreCase("Epargne"))   {
         System.out.print("Taux de placement : ");
         taux = lectureClavier.nextDouble();
      }
      System.out.print("Valeur initiale du compte : ");
      val_courante = lectureClavier.nextDouble();
   }
```

La méthode `afficherCpte()` utilise la même technique, en affichant les données caractéristiques de tout compte bancaire.

```
publicvoid afficherCpte() {
   System.out.println("Le compte n° : " + numéroCpte + " est un compte "+typeCpte);
   if ( typeCpte.equalsIgnoreCase("Epargne"))
      System.out.println(" dont le taux est" + taux);
   System.out.println("Valeur courante : " + val_courante);
}
```

Construire l'application `Projet`

Corrigé

```
import java.util.*;
public class ProjetCh7  {
 // La fonction principale
 public static void main (String [] argument)   {
   byte choix = 0 ;
   String numéroLu = "";
   Scanner lectureClavier = new Scanner(System.in);
   // b. Création d'un objet C de type Compte
   Compte   C = new Compte();
   do {
      choix = menuPrincipal();
      switch (choix){
      // c. Appeler la méthode créerCpte() par l'intermédiaire de l'objet C
      case 1 :    C.créerCpte() ;
      break;
      case 2 :
         System.out.print ( "Quel compte souhaitez vous afficher ? : ");
```

```
            numéroLu = lectureClavier.next();
            // c. Appeler la méthode afficherCpte() par l'intermédiaire de l'objet C
            // La vérification du numéro utilise une méthode de la classe String
            if ( numéroLu.equalsIgnoreCase(C.numéroCpte))  C.afficherCpte();
            else
               System.out.println("Le systeme ne connait pas le compte " + numéroLu);
         break;
         case 3 :  //option 3, créer une ligne comptable
            System.out.println("Option non programmee");
         break;
         case 4 :    sortir();
         break;
         case 5 :    alAide();
         break;
         default : System.out.println("Cette option n'existe pas ");
      }
   } while (choix != 4);
    }

    // a. Les fonctions du menu développées au cours des chapitres précédents
    public static byte menuPrincipal() {
      byte tmp;
      Scanner lectureClavier = new Scanner(System.in);
      System.out.println("1. Creation d'un compte");
      System.out.println("2. Affichage d'un compte");
      System.out.println("3. Ecrire une ligne comptable");
      System.out.println("4. Sortir");
      System.out.println("5. De l'aide");
      System.out.println();
      System.out.println("Votre choix : ");
      tmp= lectureClavier.nextByte();
      return tmp;
    }
    public static void sortir( ) {
      System.out.println("Au revoir et a bientot");
      System.exit(0) ;
    }
    public static void alAide( ) {
      System.out.println("Option 1. Pour creer un compte Courant entrer C ");
      System.out.println("Pour creer un compte Joint entrer J ");
      System.out.println("Pour creer un compte Epargne entrer E");
      System.out.print("Puis, entrer le numero du compte, et");
```

```
    System.out.println(" sa premiere valeur creditee ");
    System.out.println("Dans le cas d'un compte epargne, entrer le taux ");
    System.out.println("Option 2. Le systeme affiche les donnees du compte choisi ");
    System.out.println("Option 3. Ecrire une ligne comptable");
    System.out.println("Option 4. Pour quitter le programme");
    System.out.println("Option 5. Pour afficher de l'aide");
  }
}
```

Définir le type `LigneComptable`

Corrigé

a. Les données d'une ligne comptable sont : la valeur à créditer ou débiter, la date de l'opération, le mode de paiement ainsi que le motif. En conséquence, la déclaration des variables d'instance ci-après permet de définir les données du type `LigneComptable`.

```
public double valeur;
public String date;
public String motif;
public String mode;
```

b. La méthode `créerLigneComptable()` :

```
public void créerLigneComptable() {
    Scanner lectureClavier = new Scanner(System.in);
    System.out.print("Entrer la valeur à créditer (+) ou débiter (-) : ");
    valeur = lectureClavier.nextDouble();
    System.out.print("Date de l'opération [jj/mm/an] ");
    date = lectureClavier.next();
    System.out.print("Motif de l'operation [S(alaire),");
    System.out.print(" L(oyer), A(limentation), D(ivers)] : ");
    motif =lectureClavier.next();
    System.out.print("Mode [C(B), N(° Cheque), V(irement ) ]: ");
    mode = lectureClavier.next();
}
```

La méthode `afficherLigne()` :

```
public void afficherLigne()   {
    if (valeur < 0)
        System.out.print("Débiter : " + valeur);
    else
        System.out.print("Créditer : " + valeur);
    System.out.println(" le : "+ date +" motif : " + motif + " mode : " + mode);
}
```

Modifier le type `Compte`

Corrigé

```
import java.util.*;
```

```
public class Compte {
 public String typeCpte ;
 public double val_courante, taux ;
 public String numéroCpte ;
 // a. Déclaration d'un objet ligne de type LigneComptable
 public LigneComptable ligne;
 public void créerCpte()   {
   char tmp;
   Scanner lectureClavier = new Scanner(System.in);
   do {
         System.out.print("Type du compte [Types possibles :" );
         System.out.print("C(ourant), J(oint), E(pargne)] :");
         tmp = lectureClavier.next().charAt(0);
   } while ( tmp != 'C' && tmp!= 'J' && tmp != 'E');
   switch (tmp)   {
         case 'C' : typeCpte = "Courant";
         break;
         case 'J' : typeCpte = "Joint";
         break;
         case 'E' : typeCpte = "Epargne";
         break;
   }
   System.out.print("Numéro du compte : ");
   numéroCpte = lectureClavier.next();
   if ( typeCpte.equalsIgnoreCase("Epargne"))   {
      System.out.print("Taux de placement : ");
      taux = lectureClavier.nextDouble();
   }
   System.out.print("Valeur initiale du compte : ");
   val_courante = lectureClavier.nextDouble();
   }
   // b. Définition de la méthode créerLigne()
   public void créerLigne()   {
      // b. Création en mémoire de l'objet ligne
      ligne = new LigneComptable();
      // b. Appel de la méthode créerLigneComptable() à travers l'objet ligne
      ligne.créerLigneComptable();
      // b. La valeur courante du compte est modifiée
      val_courante = val_courante + ligne.valeur;
   }
   publicvoid afficherCpte() {
      System.out.print("Le compte n° : " + numéroCpte );
```

```
        System.out.println(" est un compte " + typeCpte);
        if ( typeCpte.equalsIgnoreCase("Epargne"))
           System.out.println(" dont le taux est" + taux);
        // c. Affichage des informations relatives à l'objet ligne
        ligne.afficherLigne();
        System.out.println("Valeur courante : " + val_courante);
     }
  }
```

Modifier l'application `Projet`

Corrigé

```
import java.util.*;
public class ProjetCh7  {
 public static void main (String [] argument)  {
   byte choix = 0 ;
   String numéroLu = "";
   Scanner lectureClavier = new Scanner(System.in);
   Compte   C = new Compte();
   do {
      choix = menuPrincipal();
      switch (choix){
      case 1 :    C.créerCpte() ;
      break;
      case 2 :
         System.out.print ( "Quel compte souhaitez vous afficher ? : ");
         numéroLu = lectureClavier.next();
         if ( numéroLu.equalsIgnoreCase(C.numéroCpte))  C.afficherCpte();
         else
            System.out.println("Le systeme ne connait pas le compte " + numéroLu);
      break;
      case 3 :  //a. option 3, créer une ligne comptable
         System.out.print ( "Pour quel compte souhaitez vous créer une ligne ? : ");
         numéroLu = lectureClavier.next();
         if ( numéroLu.equalsIgnoreCase(C.numéroCpte))  C.créerLigne();
         else
            System.out.println("Le systeme ne connait pas le compte " + numéroLu);
      break;
      case 4 :
         sortir();
      break;
      case 5 :    alAide();
      break;
      default : System.out.println("Cette option n'existe pas ");
```

```
    }
} while (choix != 4);
 }
 // La fonctionmenuPrincipal()
 // La fonction sortir( ) {
 // La fonction alAide( ) {
}
```

a. Voir commentaires dans le code source ci-dessus.

b. Si l'utilisateur affiche les données d'un compte sans avoir créé de lignes comptables, le programme stoppe son exécution en affichant l'erreur : `java.lang.NullPointerException`. En effet, la méthode `afficheCpte()` fait appel à la méthode `afficherLigne()` par l'intermédiaire d'un objet `ligne` qui n'a pas été créé en mémoire. L'objet `ligne` n'est créé que si l'utilisateur passe par l'option 3.

c. Pour remédier à cette situation, l'idée est de déclarer une nouvelle variable entière (un drapeau, *voir corrigé de l'exercice 3-5*) qui suivant sa valeur, va indiquer à l'interpréteur si une ligne a été créée ou non.

```
import java.util.*;
public class Compte {
 public String typeCpte ;
 public double val_courante, taux ;
 public String numéroCpte ;
 public LigneComptable ligne;
// Déclaration du drapeau nbLigneRéel
 public int nbLigneRéel ;
 public void créerCpte()   {
   char tmp;
   Scanner lectureClavier = new Scanner(System.in);
   do {
         System.out.print("Type du compte [Types possibles :" );
         System.out.print("C(ourant), J(oint), E(pargne)] : ");
         tmp = lectureClavier.next().charAt(0);
   } while ( tmp != 'C' && tmp!= 'J' && tmp != 'E');
   switch (tmp)   {
         case 'C' : typeCpte = "Courant";
         break;
         case 'J' : typeCpte = "Joint";
         break;
         case 'E' : typeCpte = "Epargne";
         break;
   }
   System.out.print("Numéro du compte : ");
   numéroCpte = lectureClavier.next();
   if ( typeCpte.equalsIgnoreCase("Epargne"))   {
```

```
        System.out.print("Taux de placement : ");
        taux = lectureClavier.nextDouble();
    }
    System.out.print("Valeur initiale du compte : ");
    val_courante = lectureClavier.nextDouble();
// A la création du compte, aucune ligne comptable n'a été saisie
// donc nbLignRéel vaut 0
    nbLigneRéel = 0;
}
    public void créerLigne()   {
        ligne = new LigneComptable();
        ligne.créerLigneComptable();
        // A la création d'une ligne, nbLignRéel vaut 1
        nbLigneRéel = 1;
        val_courante = val_courante + ligne.valeur;
    }
    publicvoid afficherCpte() {
        System.out.print("Le compte n° : " + numéroCpte );
        System.out.println(" est un compte " + typeCpte);
        if ( typeCpte.equalsIgnoreCase("Epargne"))
            System.out.println(" dont le taux est" + taux);
        // Si nbLignRéel est égal à 1 ( > 0), on peut afficher les données
        // d'une ligne comptable
        if (   nbLigneRéel > 0) ligne.afficherLigne();
        System.out.println("Valeur courante : " + val_courante);
    }
}
```

Chapitre 8 : Les principes du concept objet

Encapsuler les données d'un compte bancaire (*La protection* `private` *et l'accès aux données*)

Corrigé

a. En déclarant les données des types `Compte` et `LigneComptable` en mode `private`, le compilateur détecte des erreurs telles que `Variable numéroCpte in class Compte` **`not accessible`** `from class Projet`.

b. Accéder en lecture aux données de la classe `Compte`

```
import java.util.*;
public class Compte {
    private String typeCpte ;
    private double val_courante, taux ;
    private String numéroCpte ;
    private LigneComptable ligne;
    private int nbLigneRéel ;
```

```
// Accéder en lecture aux données de la classe
public String quelTypeDeCompte()    {
   return typeCpte;
}
public String quelNuméroDeCompte()  {
   return numéroCpte;
}
public double quelleValeurCourante()   {
   return val_courante;
}
public double quelTaux()   {
   return taux;
}
public void créerCpte()    {
   Scanner lectureClavier = new Scanner(System.in);
   char tmp;
   do {
      System.out.print("Type du compte [Types possibles :" );
      System.out.print("C(ourant), J(oint), E(pargne)] : ");
      tmp = lectureClavier.next().charAt(0);
   } while ( tmp != 'C' && tmp!= 'J' && tmp != 'E');
   switch (tmp)   {
      case 'C' : typeCpte = "Courant";
      break;
      case 'J' : typeCpte = "Joint";
      break;
      case 'E' : typeCpte = "Epargne";
      break;
   }
   System.out.print("Numéro du compte : ");
   numéroCpte = lectureClavier.next();
   if ( typeCpte.equalsIgnoreCase("Epargne"))   {
      System.out.print("Taux de placement : ");
      taux = lectureClavier.nextDouble();
   }
   System.out.print("Valeur initiale du compte : ");
   val_courante = lectureClavier.nextDouble();
   nbLigneRéel = 0;
}
public void créerLigne()   {
   ligne = new LigneComptable();
   ligne.créerLigneComptable();
```

```
        nbLigneRéel = 1;
        val_courante = val_courante + ligne.valeur;
    }
    publicvoid afficherCpte() {
        System.out.print("Le compte n° : " + numéroCpte );
        System.out.println(" est un compte " + typeCpte);
        if ( typeCpte.equalsIgnoreCase("Epargne"))
           System.out.println(" dont le taux est" + taux);
        if (   nbLigneRéel > 0) ligne.afficherLigne();
        System.out.println("Valeur courante : " + val_courante);
    }
}
```

Accéder en lecture aux données de la classe `LigneComptable`

```
import java.util.*;
public class LigneComptable   {
    private double valeur;
    private String date;
    private String motif;
    private String mode;
    public void créerLigneComptable()   {
        Scanner lectureClavier = new Scanner(System.in);
        System.out.print("Entrer la valeur à créditer (+) ou débiter (-) : ");
        valeur = lectureClavier.nextDouble();
        System.out.print("Date de l'opération [jj/mm/an] ");
        date = lectureClavier.next();
        System.out.print("Motif de l'operation [S(alaire),");
        System.out.print(" L(oyer), A(limentation), D(ivers)] : ");
        motif = lectureClavier.next();
        System.out.print("Entrer le mode [C(B), N(° Cheque), V(irement ) ] : ");
        mode = lectureClavier.next();
    }
    // Accéder en lecture aux données de la classe
    public double quelleValeur() {
        return valeur ;
    }
    public String quelMotif(){
        return motif ;
    }
    public String quelMode(){
        return mode ;
    }
    public String quelleDate(){
```

```
        return date ;
    }
    public void afficherLigne()   {
        if (valeur < 0)
        System.out.print("Débiter : " + valeur);
        else
        System.out.print("Créditer : " + valeur);
        System.out.println(" le : " + date + " motif: " + motif + " mode : " + mode);
    }
}
```

c. Dans la classe `Projet`, seules les instructions faisant appel directement aux données de la classe `Compte` sont à modifier. C'est à dire pour les options 2 et 3 :

```
    case 2 :
        System.out.print ( "Quel compte souhaitez vous afficher ? : ");
        numéroLu = lectureClavier.next();
    // if ( numéroLu.equalsIgnoreCase(C.numéroCpte))
    // -> numéroCpte inaccessible car private dans Compte
    if ( numéroLu.equalsIgnoreCase(C.quelNuméroDeCompte()))
        C.afficherCpte();
    case 3 :
        System.out.print ( "Pour quel compte souhaitez vous créer une ligne ? : ");
        numéroLu = lectureClavier.next();
        if ( numéroLu.equalsIgnoreCase(C.quelNuméroDeCompte()))
            C.créerLigne();
        else
            System.out.println("Le systeme ne connait pas le compte " + numéroLu);
    break;
```

d. Dans la classe `Compte`, seules les instructions faisant appel directement aux données de la classe `LigneComptable` sont à modifier. C'est à dire pour la méthode `créerLigne()` .

```
    public void créerLigne()   {
        ligne = new LigneComptable();
        nbLigneRéel = 1 ;
        val_courante = val_courante + ligne.quelleValeur();
    }
```

Encapsuler les données d'un compte bancaire (*Le contrôle des données*)

Corrigé

a. La méthode `contrôleValinit()` :

```
    private double contrôleValinit()  {
        double tmp, tmpval;
        Scanner lectureClavier = new Scanner(System.in);
        do {
            System.out.print("Valeur initiale du compte : ");
```

```
            tmpval= lectureClavier.nextDouble();
        } while ( tmpval <= 0);
        return tmpval;
    }
```

b. La méthode `contrôleMode()` :

```
    private String contrôleMode() {
        String tmpS = "";
        char tmpc ;
        Scanner lectureClavier = new Scanner(System.in);
        do {
            System.out.print("Mode [C(B), N(° Cheque), V(irement ) ]: ");
            tmpc = lectureClavier.next().charAt(0);
        } while ( tmpc != 'C' && tmpc != 'N' && tmpc != 'V');
        switch (tmpc) {
            case 'C' : tmpS = "CB";
            break;
            case 'N' : tmpS = "Cheque";
            break;
            case 'V' : tmpS = "Virement";
            break;
        }
        return tmpS;
    }
```

La méthode `contrôleMotif()` :

```
    private String contrôleMotif() {
        String tmpS = "";
        char tmpc ;
        Scanner lectureClavier = new Scanner(System.in);
        do {
            System.out.print("Motif de l'operation [S(alaire),");
            System.out.print(" L(oyer), A(limentation), D(ivers)] : ");
            tmpc = lectureClavier.next().charAt(0);
        } while ( tmpc != 'S' && tmpc != 'L' && tmpc != 'A' && tmpc != 'D');
        switch (tmpc) {
            case 'S' : tmpS = "Salaire";
            break;
            case 'L' : tmpS = "Loyer";
            break;
            case 'A' : tmpS = "Alimentation";
            break;
            case 'D' : tmpS = "Divers";
            break;
```

```
        }
        return tmpS;
    }
```

c. La méthode `créerCpte()` :

```
    public void créerCpte()    {
        Scanner lectureClavier = new Scanner(System.in);
        // Le type du compte est contrôlé
        typeCpte = contrôleType();
        System.out.print("Numéro du compte : ");
        numéroCpte = lectureClavier.next();
        if ( typeCpte.equalsIgnoreCase("Epargne")) {
            System.out.print("Taux de placement : ");
            taux = lectureClavier.nextDouble();
        }
        // La valeur courante du compte est contrôlée
        val_courante = contrôleValinit();
    }
```

La méthode `créerLigneComptable()` :

```
    public void créerLigneComptable()   {
        Scanner lectureClavier = new Scanner(System.in);
        System.out.print("Entrer la valeur à créditer (+) ou débiter (-) : ");
        valeur = lectureClavier.nextDouble();
        System.out.print("Date de l'opération [jj/mm/an] ");
        date = lectureClavier.next();
        // Le motif de l'opération est contrôlé
        motif = contrôleMotif();
        // Le mode de l'opération est contrôlé
        mode = contrôleMode();
    }
```

Encapsuler les données d'un compte bancaire (*Les constructeurs de classe*)

Corrigé

a. Le constructeur de la classe `LigneComptable` s'écrit en modifiant l'entête de la méthode `créerLigneComptable()`, comme suit :

```
    public LigneComptable() {
        Scanner lectureClavier = new Scanner(System.in);
        System.out.print("Entrer la valeur à créditer (+) ou débiter (-) : ");
        valeur = lectureClavier.nextDouble();
        System.out.print("Date de l'opération [jj/mm/an] ");
        date = lectureClavier.next();
        motif = contrôleMotif();
```

```
        mode = contrôleMode();
    }
```

Le constructeur de la classe `Compte` s'écrit en modifiant l'entête de la méthode `créerCompte()`, comme suit :

```
    public Compte ()  {
        Scanner lectureClavier = new Scanner(System.in);
        typeCpte = contrôleType();
        System.out.print("Numéro du compte : ");
        numéroCpte = lectureClavier.next();
        if ( typeCpte.equalsIgnoreCase("Epargne")) {
            System.out.print("Taux de placement : ");
            taux = lectureClavier.nextDouble();
        }
        val_courante = contrôleValinit();
        nbLigneRéel = 0 ;
    }
```

Cette dernière transformation fait que l'interpéteur n'exécute plus le constructeur par défaut qui initialisait automatiquement toutes les données du compte à `0` ou à `null` mais, le nouveau constructeur. Ainsi, à l'exécution du programme, la demande de saisie d'un numéro de compte est affichée au lieu du menu. En effet, l'appel du constructeur est réalisé avant l'affichage du menu.

b. Pour corriger ce défaut, la première idée est de déplacer l'appel du constructeur dans l'option1 du programme. Ainsi, l'affichage du menu sera réalisé avant la demande de saisie d'un numéro de compte.

```
public static void main (String [] argument) {
    byte choix = 0 ;
    String numéroLu = "", vide ="";
    Scanner lectureClavier = new Scanner(System.in);
    Compte   C ;
    do {
        choix = menuPrincipal();
        switch (choix){
            case 1 :    C = new Compte();
            break;
        // etc..
```

Malheureusement, ce déplacement est mal perçu par le compilateur, ce dernier détectant une erreur du type `Variable C may not have been initialized.` En effet, l'objet `C` n'est construit qu'en option `1`. Si l'utilisateur choisit l'option `2` avant de créer le compte, l'interpréteur sera dans l'impossibilité de l'afficher, puisqu'il n'existera pas en mémoire.

c. En surchargeant le constructeur `Compte()` de la façon suivante :

```
    public Compte(String num){
        numéroCpte = num;
        nbLigneRéel = 0;
    }
```

d. Nous pouvons construire un premier objet C, qui fait appel au constructeur avec paramètre, comme suit :

```
public static void main (String [] argument) {
   Scanner lectureClavier = new Scanner(System.in);
   byte choix = 0 ;
   String numéroLu = "", vide ="";
   Compte   C = new Compte(vide);
   do {
      choix = menuPrincipal();
      switch (choix){
         case 1 : C = new Compte();
         break;
      // etc..
```

De cette façon, l'objet C est construit en mémoire dès le début de l'exécution du programme. Les données de l'objet C sont initialisées à 0 ou null. L'utilisateur peut donc théoriquement afficher le compte avant de le créer. Ensuite, les véritables valeurs du compte sont saisies en option 1 du programme, en faisant appel au constructeur par défaut. Remarquez qu'il ne s'agit pas d'une nouvelle déclaration mais, bien d'une initialisation. Dans le cas d'une déclaration en option 1, nous aurions eu :

```
case 1 : Compte C = new Compte();
```

Dans la classe Compte, l'appel au constructeur LigneComptable() est réalisé dans la méthode créerLigne(), comme suit :

```
public void créerLigne()   {
   ligne = new LigneComptable();
   nbLigneRéel = 1 ;
   val_courante = val_courante + ligne.quelleValeur();
}
```

Comprendre l'héritage (*Protection des données héritées*)

Corrigé

a. La classe CpteEpargne s'écrit comme suit :

```
public class CpteEpargne extends Compte {
   private doubletaux ;
   public void afficherCpte() {
      System.out.println(" Taux d'epargne du compte :" + taux);
   }
   public double quelTaux()   {
      return taux;
   }
}
```

b. et c. La classe Compte est modifiée de la façon suivante :

```
import java.util.*;
public class Compte  {
   private String typeCpte ;
```

```
    // c. Seule la donnée val_courante doit être accessible par la classe CpteEpargne
    protected double val_courante;
    private String numéroCpte ;
    private LigneComptable ligne;
    private int nbLigneRéel ;
    // b. Suppression de la déclaration private double taux ;
    publicCompte ()   {
       Scanner lectureClavier = new Scanner(System.in);
       typeCpte = contrôleType();
       System.out.print("Numéro du compte : ");
       numéroCpte = lectureClavier.next();
       val_courante = contrôleValinit();
       nbLigneRéel = 0; ;
// b. Suppression de la saisie du taux ;
    }
    public String quelTypeDeCompte()    {
       return typeCpte;
    }
    public String quelNuméroDeCompte()  {
       return numéroCpte;
    }
// b. Suppression de la méthode d'accès en lecture quelTaux()
    public double quelleValeurCourante()   {
       return val_courante;
    }
    public LigneComptable quelleLigneCompable(){
       return ligne;
    }
    private String contrôleType()    {
       Scanner lectureClavier = new Scanner(System.in);
       char tmpc;
       String tmpS = "";
       do {
             System.out.print("Type du compte [Types possibles :" );
             System.out.print("C(ourant), J(oint)] : ");
             tmpc = lectureClavier.next().charAt(0);
       } while ( tmpc != 'C' && tmpc != 'J' );
       switch (tmpc) {
          case 'C' : tmpS = "Courant";
          break;
          case 'J' : tmpS = "Joint";
          break;
```

```
// b. Suppression du case 'E'
      }
      return tmpS;
   }
   private double contrôleValinit()  {
      Scanner lectureClavier = new Scanner(System.in);
      double tmp, tmpval;
      do {
         System.out.print("Valeur initiale du compte : ");
         tmpval= lectureClavier.nextDouble();
      } while ( tmpval <= 0);
      return tmpval;
   }
   public void créerLigne()   {
      ligne = new LigneComptable();
      nbLigneRéel = 1 ;
      val_courante = val_courante + ligne.quelleValeur();
   }
   publicvoid afficherCpte() {
      System.out.print("Le compte n° : " + numéroCpte );
      System.out.println(" est un compte " + typeCpte);
      // b. Suppression de l'affichage du taux ;
      if (   nbLigneRéel > 0) ligne.afficherLigne();
      System.out.println("Valeur courante : " + val_courante);
      if (val_courante < 0)
         System.out.println("Attention compte débiteur ... !!!");
   }
}
```

Comprendre l'héritage (*Le contrôle des données d'un compte d'épargne*)

Corrigé

Pour contrôler la validité du taux, insérer la méthode `contrôleTaux()` dans la classe `CpteEpargne`.

```
   private double contrôleTaux() {
      Scanner lectureClavier = new Scanner(System.in);
      double tmp;
      do {
         System.out.print("Taux de placement : ");
         tmp = lectureClavier.nextDouble();
      } while (tmp < 0);
      return tmp;
   }
```

Comprendre l'héritage (*Le constructeur d'une classe dérivée*)

Corrigé

a. L'instruction `super("Epargne");` fait appel au constructeur de la classe `Compte` possédant en paramètre un `String`, le paramètre de `super()` étant un `String`.

b. Le constructeur de la classe `Compte` possédant en paramètre un `String` doit être modifié de sorte qu'il gère la création d'un compte épargne. Cette gestion est réalisée comme suit :

```
public Compte(String type){
   Scanner lectureClavier = new Scanner(System.in);
   if (type.equalsIgnoreCase("Epargne")) {
      typeCpte = type;
      System.out.print("Numéro du compte : ");
      numéroCpte = lectureClavier.next();
      val_courante = contrôleValinit();
      nbLigneRéel = 0 ;
   }
}
```

Comprendre l'héritage (*Le polymorphisme*)

Corrigé

a. Pour afficher la totalité des informations relatives à un compte épargne, la méthode `afficherCpte()` de la classe `CpteEpargne`, doit faire appel à la méthode `afficherCpte()` de la classe `Compte`, comme suit :

```
public void afficherCpte() {
   super.afficherCpte();
   System.out.println(" Taux d'epargne du compte :" + taux);
}
```

b. L'option `1` de l'application `Projet` s'écrit comme suit :

```
case 1 :
   System.out.print (" Compte Epargne (o/n) : ");
   if (lectureClavier.next().charAt(0) == 'o')     C = new CpteEpargne();
   else      C = new Compte();
   break;
```

Chapitre 9 : Collectionner un nombre fixe d'objets

Traiter dix lignes comptables (*Transformer les constructeurs*)

Corrigé

```
public Compte ()  {
   Scanner lectureClavier = new Scanner(System.in);
   typeCpte = contrôleType();
   System.out.print("Numéro du compte : ");
   numéroCpte = lectureClavier.next();
   val_courante = contrôleValinit();
   // a. Création en mémoire de la donnée ligne sous la forme d'un tableau
```

```
    ligne = new LigneComptable[NBLigne];
    // b. Initialisation
    nbLigneRéel = -1;
 }
 publicCompte( String type){
    Scanner lectureClavier = new Scanner(System.in);
    if (type.equalsIgnoreCase("Epargne")) {
       typeCpte = type;
       System.out.print("Numéro du compte : ");
       numéroCpte = lectureClavier.next();
       val_courante = contrôleValinit();
    // a. Création en mémoire de la donnée ligne sous la forme d'un tableau
       ligne = new LigneComptable[NBLigne];
    // b. Initialisation
       nbLigneRéel = -1;
    }
 }
```

Traiter dix lignes comptables (*Transformer la méthode* `créerLigne()`)

Corrigé

```
    // La méthode créerLigne()
    public void créerLigne() {
    // a. incrémente le nombre de lignes à chaque ligne créée
       nbLigneRéel++;
       // b. Si le nombre de lignes est < 10, création d'une nouvelle ligne
       if (nbLigneRéel < NBLigne)
          ligne [nbLigneRéel] = new LigneComptable();
       // c. Si le nombre de lignes est > 10, décaler toutes les lignes vers le haut
       else {
          nbLigneRéel--;
          décalerLesLignes();
          ligne [nbLigneRéel] = new LigneComptable();
       }
       // d. Modifier la valeur courante du compte
       val_courante = val_courante + ligne[nbLigneRéel].quelleValeur();
    }
    private void décalerLesLignes() {
       for(int i = 1; i < NBLigne ; i++)
       ligne[i-1] = ligne[i];
    }
```

Traiter dix lignes comptables (*Transformer la méthode* `afficherCompte()`)

Corrigé

```
    Public void afficherCpte() {
       System.out.print("Le compte n° : " + numéroCpte );
       System.out.println(" est un compte "+typeCpte);
       // Si une ligne comptable a été créée, l'afficher
       if (nbLigneRéel >=0) {
          for (int i = 0; i <= nbLigneRéel; i++) ligne[i].afficherLigne();
       }
       System.out.println("Valeur courante : " + val_courante);
       if (val_courante < 0) System.out.println("Attention compte débiteur ... !!!");
    }
```

Chapitre 10 : Collectionner un nombre indéterminé d'objets

Les comptes sous forme de dictionnaire (*La classe* `ListeCompte`)

Corrigé

```
import java.util.*;
import java.io.*;
public class ListeCompte {
   private HashMap<String, Compte> liste;
   // a. Le constructeur de la classe ListeCompte fait appel au constructeur
   //de la classe HashMap
   public ListeCompte() {
      liste = new HashMap<String, Compte>();
   }
   // b. La méthode ajouteUnCompte()
   public void ajouteUnCompte(String t) {
      Compte nouveau = new Compte("");
   // b. Si le paramètre vaut "A" un compte est créé
      if (t.equalsIgnoreCase("A")) nouveau = new Compte();
   // b. Sinon un compte épargne est créé
      else if (t.equalsIgnoreCase("E"))nouveau = new CpteEpargne();
      String clé = nouveau.quelNuméroDeCompte();
   // b. Une fois créé, le compte est inséré dans le dictionnaire
      if (liste.get(clé) == null) liste.put(clé, nouveau);
      else System.out.println("Ce compte existe deja !");
   }
   // c. Créer une ligne pour un compte donné
   public void ajouteUneLigne(String n) {
      String clé = n;
      Compte c = (Compte) liste.get(clé);
      // c. Si le compte existe, une ligne est créée
```

```
        if (c != null)c.créerLigne();
        else System.out.println("Le systeme ne connait pas le compte "+n);
    }
    // d. retoune un objet Compte
    public Compte quelCompte(String n){
        String clé = n;
        Compte c = (Compte) liste.get(clé);
        if (c == null)
        System.out.println("Le systeme ne connait pas le compte "+n);
        return(c);
    }
    // d. recherche un compte à partir du numéro passé en paramètre
    public void rechercheUnCompte (String n) {
        String clé = n;
        Compte c = (Compte) liste.get(clé);
        if (c != null)    c.afficherCpte();
        else System.out.println("Le systeme ne connait pas le compte "+n);
    }
    // d. supprime un compte à partir du numéro passé en paramètre
    public void supprimeUnCompte(String n) {
        String clé = n;
        Compte c = (Compte) liste.get(clé);
        if (c != null){
            liste.remove(clé);
            System.out.println(n + " a été supprimé ");
        }
        else System.out.println(n + " est inconnu ! ");
    }
    // d. affiche tous les comptes stockés dans le dictionnaire
    public void afficheLesComptes()  {
        if(liste.size() != 0)   {
            Collection<Compte> colCompte = liste.values();
            for (Compte c: colCompte )c.afficherCpte();
        }
        else System.out.println("Aucun compte n'a ete cree, dans cette liste");
    }
}
```

Les comptes sous forme de dictionnaire (*L'application* `Projet`)

Corrigé

```
import java.util.*;
public class ProjetCh10 {
    public static void main (String [] argument) {
```

```
        Scanner lectureClavier = new Scanner(System.in);      byte choix = 0 ;
        String numéroLu = "";
        // Créer une liste de Compte
        ListeCompte C = new ListeCompte();
        do {
           choix = menuPrincipal();
           switch (choix) {
           case 1 :
              System.out.print (" Compte Epargne (o/n) : ");
           // b. Créer un compte épargne ou un autre type de compte
              if (lectureClavier.next().charAt(0) == 'o')     C.ajouteUnCompte("E") ;
              else      C.ajouteUnCompte("A");
           break;
           case 2 :
              System.out.print ( "Quel compte souhaitez vous afficher ? : ");
              numéroLu = lectureClavier.next();
              // a. rechercher et afficher un compte
              C.rechercheUnCompte(numéroLu);
           break;
           case 3 :
              // a. afficher tous les comptes
              C.afficheLesComptes();
           break;
           case 4 :
              System.out.print("Pour quel compte souhaitez vous créer une ligne ? : ");
              numéroLu = lectureClavier.next();
              // a. ajouter une ligne à un compte
              C.ajouteUneLigne(numéroLu);
           break;
           case 5 :
              System.out.print ( "Quel compte souhaitez vous supprimer ? : ");
              numéroLu = lectureClavier.next();
              // a. supprimer un compte
              C.supprimeUnCompte(numéroLu);
           break;
           case 6 :    sortir();
           break;
           case 7 :    alAide();
           break;
           default :   System.out.println("Cette option n'existe pas ");
        }
     } while (choix != 6);
```

```
  }
// c. modifier l'affichage du menu principal
 public static byte menuPrincipal() {
   byte tmp;
   Scanner lectureClavier = new Scanner(System.in);
   System.out.println("1. Création d'un compte");
   System.out.println("2. Affichage d'un compte");
   System.out.println("3. Affichage de tous les comptes");
   System.out.println("4. Ecrire une ligne comptable");
   System.out.println("5. Supprimer un compte ");
   System.out.println("6. Sortir");
   System.out.println("7. De l'aide");
   System.out.println();
   System.out.print("Votre choix : ");
   tmp= lectureClavier.nextByte();
   return tmp;
 }
 public static void sortir( ) {
   System.out.println("Au revoir et a bientot");
   System.exit(0) ;
 }
 public static void alAide( ) {
}
```

La sauvegarde des comptes bancaires (*La classe* `FichierCompte`)

Corrigé

```
import java.io.*;
public class FichierCompte{
   // a. Le fichier de sauvegarde s'appelle Compte.dat
   privateString nomDuFichier = "Compte.dat";
   private ObjectOutputStream fWo;
   privateObjectInputStream fRo;
   private char mode;
   public boolean ouvrir(String s) {
      try {
         mode = (s.toUpperCase()).charAt(0);
         if (mode == 'R' || mode == 'L')
            fRo= new ObjectInputStream(new FileInputStream(nomDuFichier));
         else     if (mode == 'W' || mode == 'E')
            fWo= new ObjectOutputStream(new FileOutputStream(nomDuFichier));
         return true;
      }
```

```
        catch (IOException e)   {
            return false;
        }
    }
    public void fermer() {
        try {
            if (mode == 'R' || mode == 'L') fRo.close();
            else  if (mode == 'W' || mode == 'E')fWo.close();
        }
        catch (IOException e){
            System.out.println(nomDuFichier+" : Erreur à la fermeture ");
        }
    }
    // b. La méthode lire retourne un objet de type ListeCompte
    public ListeCompte lire() {
        try {
            ListeCompte tmp = (ListeCompte) fRo.readObject();
            return tmp;
        }
        catch (IOException e) {
            System.out.println(nomDuFichier+" : Erreurde lecture ");
        }
        catch (ClassNotFoundException e) {
            System.out.println(nomDuFichier+" n'est pas du bon format ");
        }
        return null;
    }
    // b. La méthode ecrire sauvegarde un objet de type ListeCompte
    public void ecrire(ListeCompte tmp) {
        try {
            if (tmp != null)fWo.writeObject(tmp);
        }
        catch (IOException e) {
            System.out.println(nomDuFichier+" : Erreur en cours d'écriture ");
        }
    }
}
```

c. Les en-têtes des classes `Compte`, `CpteEpargne`, `ListeCompte` et `LigneComptable` doivent utiliser le terme `implements Serializable`

La sauvegarde des comptes bancaires (*L'application* `Projet`)

Corrigé

a. La lecture du fichier « `Compte.dat` » est réalisée de la façon suivante :

```
FichierCompte F = new FichierCompte();
if (F.ouvrir("L")) {
   C = F.lire();
   F.fermer();
}
```

Ces instructions sont à insérer dans le fichier `Projet.java`, juste avant l'entrée dans la boucle `do…while();`

b. La sauvegarde automatique s'effectue à l'option `6`, comme suit :

```
case 6 :
   System.out.println("Sauvegarde des données dans Compte.dat");
   F.ouvrir("E");
   F.ecrire(C);
   F.fermer();
   sortir();
break;
```

La mise en place des dates dans les lignes comptables (*Rechercher des méthodes dans les différents packages*)

Corrigé

a. et b. Les différentes recherches proposées nous montrent qu'il existe une classe nommée `SimpleDateFormat` qui permet la vérification de la validité d'une date. La date est traduite en objet `Date` à partir d'une chaîne de caractères comprenant le jour, le mois et l'année. Pour séparer chacune de ces valeurs, il est nécessaire de définir un caractère appelé séparateur de champs.

La mise en place des dates dans les lignes comptables (*Ecrire la méthode* `contrôleDate()`)

Corrigé

Le contrôle de la validité d'une date est réalisé de la façon suivante :

```
private String contrôleDate() {
   Scanner lectureClavier = new Scanner(System.in);
   int nb = 0;
   Date d = null;
   // Choix du format de la date, le séparateur est le caractère '/'
   SimpleDateFormat formatIn=new SimpleDateFormat("dd/MM/yyyy");
   String sdate;
   // d. Tant que d n'est pas correctement initialisée
   while (d == null){
      try {
         System.out.print("Entrer une date (jj/mm/aaaa): ");
         // a. Saisie de la chaine correspondant à la date
         // b. Et traduction en objet de type Date
         //Si la traduction ne peut se réaliser
         //La chaîne n'est pas au bon format, d vaut null
         d = formatIn.parse(lectureClavier.next());
      }
```

```
            catch(ParseException p) {
               // c. Si la traduction ne peut se réaliser, une exception est détectée
               // d. On incrémente un compteur
               nb++;
               // d. Si le compteur >= 3, la date est initialisée à la date placée en
               //mémoire de l'ordinateur
               if (nb >= 3) d = new Date();
            }
         }
         // e. Lorsque la date est au bon format, on la transforme en String
         sdate = formatIn.format(d);
         return sdate;
      }
```

Chapitre 11 : Dessiner des objets

Calcul de statistiques (*Les classes* `ListeCompte` *et* `Compte`)

Corrigé

a. Pour réaliser les statistiques, nous avons besoin de connaître pour chaque ligne comptable saisie, la valeur et le motif de l'opération.

b. La valeur et le motif sont accesibles de la la classe `Compte` grâce aux méthodes `quelleValeur()` et `quelMotif()`. Par contre, le tableau des lignes comptables n'est pas accessible puisqu'il est déclaré en `private`. Il est donc nécessaire de créer des méthodes d'accès en consultation au tableau `ligne`, ainsi qu'au nombre de lignes effectivement créées. Ces méthodes sont définies comme suit :

```
   public LigneComptable quelleLigne(int n) {
      return ligne[n];
   }
   public int combienLignes() {
      return nbLigneRéel;
   }
```

Ces deux méthodes sont à insérer dans la classe `Compte`.

Calcul de statistiques (*La méthode* `statParMotif()`)

Corrigé

```
public class Stat{
   // b. Le calcul des statistiques est réalisé à partir d'un objet Compte
   public Compte cpte;
   // b. Les resultats sont stockés dans des variables de double précision
   private double prctDiv, prctLoy, prctAli;
   public Stat(Compte c) {
       cpte = c;
   }
   privatedouble pourcentage(double nb, double t){
      doubleprct = 0;
```

```
        if (t > 0) prct = (double) nb / t* 100;
        return prct;
    }
// a. La méthode calcule les statistiques en fonction du motif de l'opération
public void statParMotif() {
        double totCredit=0;
        double totDiv=0, totLoy=0, totAli=0;
        // a. Pour chaque ligne comptable enregistrée
        for(int i = 0; i <= cpte.combienLignes() ; i++){
           // a. S'il s'agit d'un crédit, en faire la somme
           if (cpte.quelleLigne(i).quelleValeur() > 0)
              totCredit = totCredit +cpte.quelleLigne(i).quelleValeur();
           // a. Si le motif est "Divers" en faire la somme
           if (cpte.quelleLigne(i).quelMotif().equalsIgnoreCase("Divers"))
              totDiv = totDiv + Math.abs(cpte.quelleLigne(i).quelleValeur());
           // a. Si le motif est "Loyer" en faire la somme
           if (cpte.quelleLigne(i).quelMotif().equalsIgnoreCase("Loyer"))
              totLoy = totLoy +Math.abs(cpte.quelleLigne(i).quelleValeur());
           // a. Si le motif est "Alimentation" en faire la somme
           if (cpte.quelleLigne(i).quelMotif().equalsIgnoreCase("Alimentation"))
              totAli = totAli +Math.abs(cpte.quelleLigne(i).quelleValeur());
        }
        // a. Calculer le pourcentage pour chacun des motifs
        prctAli = pourcentage(totAli, totCredit);
        prctLoy = pourcentage(totLoy, totCredit);
        prctDiv = pourcentage(totDiv, totCredit);
        // d. Afficher le résultat
        System.out.print("A : " + prctAli + "L : " + prctLoy + "D : "+prctDiv);
}
```

c. Pour afficher simplement le résultat du calcul réalisé par la méthode `statParMotif()` :

```
import java.util.*;
public class Projet  {
public static void main (String [] argument) {
   byte choix = 0 ;
   String numéroLu = "";
   Scanner lectureClavier = new Scanner(System.in);
   ListeCompte liste = new ListeCompte();
   FichierCompte F = new FichierCompte();
   // Lecture du fichier Compte.dat
   if (F.ouvrir("L")) {
      liste = F.lire();
      F.fermer();
```

```
    }
    System.out.println("Affichage des statistiques");
    System.out.print ( "Quel compte souhaitez vous afficher ? : ");
    numéroLu = lectureClavier.next();
    Compte cpte =  new Compte("");
    // Le compte existe-t-il dans la liste ?
    cpte = liste.quelCompte(numéroLu);
    // si Oui
    if (cpte != null) {
       Stat s = new Stat(cpte);
       // Afficher les statistiques
       s.statParMotif();
    }
 }
}
```

L'interface graphique (*L'affichage de l'histogramme*)

Corrigé

a. La méthode `dessine()` à insérer dans la classe `Stat`

```
public void dessine(Graphics g){
      statParMotif();
      // Affichage en gris du n° du compte ainsi que du texte Crédit et Débit au
      // dessus de chaque colonne de l'histogramme
      g.setColor(Color.darkGray);
      g.drawString("Compte n° : " + cpte.quelNuméroDeCompte(), 100, 50);
      g.drawString("Crédit", 105, 220);
      g.drawString("Débit", 160, 220);
      // Affichage en bleu dupremier rectangle correspondant à la colonne Crédit
      g.setColor(Color.blue);
      g.fillRect(100,100,50,100);
      // L'affichage est réalisé par rapport au coin supérieur gauche de la fenêtre
      // La variable reste permet de décaler les rectangles de la colonne Débit
      // vers le bas, de sorte que les deux colonnes soient ajustées horizontalement
      int reste = (int) (100 - prctLoy - prctDiv - prctAli);
      // Affichage en vert du rectangle correspondant au motif Loyer
      g.setColor(Color.green.darker().darker());
      g.fillRect(150, 100 + reste, 50, (int)prctLoy);
      g.drawString("Loyer", 210, 95 + (int)prctLoy+reste);
      // Affichage en magenta du rectangle correspondant au motif Alimentation
      g.setColor(Color.magenta);
      g.fillRect(150, 100 + (int)prctLoy+reste, 50, (int)prctAli);
      g.drawString("Alimentation", 210, 95 + (int)(prctLoy+prctAli)+reste);
```

```
        // Affichage en rouge du rectangle correspondant au motif Divers
        g.setColor(Color.red);
        g.fillRect(150, 100 + (int)(prctLoy + prctAli) + reste, 50, (int)prctDiv);
        g.drawString("Divers", 210, 95 + (int)(prctLoy + prctAli + prctDiv) + reste);
    }
}
```

b. La fenêtre principale :

```
import java.awt.*;
class Fenetre extends Frame {
    public   final static int HT = 300;
    public   final static int LG = 300;
    public Fenetre(Stat s){
        setTitle("Les statistiques du compte " );
        setSize(HT, LG);
        setBackground(Color.darkGray);
        addWindowListener(new GestionQuitter());
        // La zone de dessin
        Dessin page = new Dessin(s);
        add(page , "Center");
        // Le bouton quitter
        Button bQuitter = new Button ("Quitter");
        bQuitter.addActionListener(new GestionQuitter());
        add(bQuitter, "South");
        setVisible(true);
    }
}
```

c. La classe `Dessin` :

```
import java.awt.*;
public class Dessin extends Canvas {
    private Color couleur = Color.green;
    public final static Color couleurFond = Color.white;
    public Stat s;
    public Dessin(Stat s)   {
        setBackground(couleurFond);
        setForeground(couleur);
        setCursor(new Cursor(Cursor.CROSSHAIR_CURSOR));
        this.s = s;
    }
    // L'affichage de l'histogramme
    public void paint (Graphics g){
        s.dessine(g);
    }
```

```
}
```

d. La classe `GetsionQuitter` :

```
import java.awt.event.*;
   // Gère les écouteurs d'événements de fenêtres et d'actions
   public class GestionQuitter extends WindowAdapter implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         System.exit(0);
      }
      public void windowClosing(WindowEvent e) {
         System.exit(0);
      }
   }
```

L'interface graphique (*La saisie d'un numéro de compte*)

Corrigé

La classe `Saisie` définit un type d'objet qui permet la saisie d'une valeur par l'intermédiaire d'une fenêtre graphique. La valeur est écrite dans un objet appelé `TextField` et est récupérée grâce à la méthode `getText()` lorsqu'une action est réalisée dans cet objet (par exemple lorsque l'utilisateur valide la saisie de la valeur en tapant sur la touche "`Entrée`".

Pour notre projet la fenêtre de saisie doit s'afficher de la façon suivante :

Une fois le numéro validé, l'application doit afficher l'histogramme dans une nouvelle fenêtre. Par conséquent, la classe `Saisie` s'écrit comme suit :

```
import java.awt.*;
import java.awt.event.*;
public class Saisie implements ActionListener {
   ListeCompte lc;
   TextField réponse;
    public Saisie (ListeCompte tmp_lc) {
      // La liste des comptes est passé en paramètre du constructeur
      lc = tmp_lc;
      // Titre de la fenêtre de saisie
      Frame f = new Frame ("Saisie du n° de Compte");
      f.setSize(300, 50);
      f.setBackground(Color.white);
      f.setLayout(new BorderLayout());
      // Affichage du texte dans la fenêtre
      f.add (new Label("Numero du compte :"), "West");
      réponse = new TextField(10);
      f.add(réponse, "East");
      réponse.addActionListener(this);
```

```
        f.setVisible(true);
    }
    public void actionPerformed(ActionEvent evt) {
        String numéro = réponse.getText();
        Compte cpte =  new Compte("");
        // Si le numéro saisi existe dans la liste des comptes
        cpte = lc.quelCompte(numéro);
        if (cpte != null) {
        // Si le numéro saisi existe dans la liste des comptes
        // Calcul des statitstiques
            Stat s = new Stat(cpte);
        // Et affichage dans la nouvelle fenêtre
            new Fenetre (s);
        }
    }
}
```

L'application `Projet` fait appel à la classe `Saisie` de la façon suivante :

```
public class ProjetIG{
    public static void main (String [] argument){
        ListeCompte liste = new ListeCompte();
        FichierCompte F = new FichierCompte();
        // Lecture du fichier Compte.dat
        if (F.ouvrir("L")) {
            liste = F.lire();
            F.fermer();
        }
        // Si la liste n'est pas vide, affichage de la fenêtre de saisie
        if (liste != null)      new Saisie(liste);
    }
}
```

Chapitre 12 : Créer une interface graphique

Mise en place des éléments graphiques

Corrigé

Voir fenêtre Design des classes du projet `ProjetChapitre12` :

- `Main`
- `CompteDialogue`
- `LigneDialogue`
- `CompteEdit`
- `LigneEdit`

Les fichiers associées se trouvent dans le répertoire `Sources/Projet/Chapitre12/NetBeansProjects/EditeurExercice/src/Projet` du CD-Rom livré avec l'ouvrage.

Définition des comportements

Corrigé :

Dans la classe `Main` :

```
// Définition des propriétés choixAction et numeroCompte
String choixAction = "Creer";
String numeroCompte = " " ;

private void modifierRdBtnActionPerformed(java.awt.event.ActionEvent evt) {
   // Initialiser l'action à "Modifier"
   choixAction = "Modifier";
}

private void creerRdBtnActionPerformed(java.awt.event.ActionEvent evt) {
   // Initialiser l'action à "Creer"
   choixAction = "Creer";
}
private void editerRdBtnActionPerformed(java.awt.event.ActionEvent evt) {
   // Initialiser l'action à "Modifier"
   choixAction = "Editer";
}
private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
   // Récupérer le numéro du compte
   numeroCompte = numeroAsaisir.getText();
   // Si l'action est "Creer"
   if(choixAction.equals("Creer")){
      // Ouvrir la boîte de dialogue CompteDaliogue
      new CompteDialogue(numeroCompte);
   }
   // Si l'action est "Modifier"
   else  if(choixAction.equals("Modifier")){
      // Lire le fichier associé au compte
      FichierCompte fc = new FichierCompte();
       if (fc.ouvrir("Comptes/"+numeroCompte+".dat","R")) {
      // Si le fichier existe, stocker les informations lues dans un objet unCompte
         Compte unCompte = fc.lire();
         fc.fermer();
         // Ouvrir la boîte de dialogue LigneDialogue en passant en paramètre les
         // informations lues
          new LigneDialogue(unCompte);
```

```
        }
        // Gestion des erreurs
        // Si le fichier n'existe pas, afficher un message d'alerte
        else new Message("Le compte : ", numeroCompte , " n'existe pas");
    }
    // Si l'action est "Editer"
    else  if(choixAction.equals("Editer")){
        // Lire le fichier associé au compte
        FichierCompte fc = new FichierCompte();
        if (fc.ouvrir("Comptes/"+numeroCompte+".dat","R")){
        // Si le fichier existe, stocker les informations lues dans un objet unCompte
            Compte unCompte = fc.lire();
            fc.fermer();
            // Ouvrir la boîte de dialogue CompteEdit en passant en paramètre les
            // informations lues
            new CompteEdit(unCompte);
        }
        // Gestion des erreurs
        // Si le fichier n'existe pas, afficher un message d'alerte
        else new Message("Le compte : ", numeroCompte , " n'existe pas");
    }
}
```

Dans la classe `CompteDialogue` :

```
// Définition des propriétés
String typeCompte="Courant";
String numeroCompte = " " ;
public CompteDialogue(String cpt) {
    initComponents();
    // Affichage du numéro de compte dans le cadre du composant
    Border cadre = BorderFactory.createTitledBorder(" Compte n° : " + cpt + "  ");
    boiteMenu.setBorder(cadre);
    setBounds(500, 130,470, 390);
    setVisible(true);
    // Affichage d'une image en fond de fenêtre
    ImageIcon iconPhoto = new ImageIcon("Ressources/FondCompte.png");
    photoFond.setIcon(iconPhoto);
    creerGroupeBox();
    numeroCompte = cpt;
}
private void entrepriseActionPerformed(java.awt.event.ActionEvent evt) {
    // Initialiser le type du compte à "Entreprise"
    typeCompte="Entreprise";
```

```
}
private void courantActionPerformed(java.awt.event.ActionEvent evt) {
   // Initialiser le type du compte à "Courant"
   typeCompte="Courant";
}
private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
   // Récupérer la valeur du montant déposé à l'ouverture du compte
   double valeurInitiale = Double.parseDouble(valeurAsaisir.getText());
   // Créer un objet de type Compte à l'aide de son constructeur.
   Compte unCompte = new Compte(numeroCompte, typeCompte, valeurInitiale) ;
   // Enregistrer le compte dans un fichier objet dont le nom porte le numéro du
   // compte suivi de l'extension .dat
   FichierCompte fc = new FichierCompte();
   fc.ouvrir("Comptes/"+numeroCompte+".dat","W");
   fc.ecrire(unCompte);
   fc.fermer();
   // Fermer la fenêtre en cours sans quitter l'application.
   this.dispose();
}
```

Dans la classe `LigneDialogue` :

```
// Définition des propriétés
private String motif = "Divers", transaction = "C.B.";
private Compte compte;
public LigneDialogue(Compte cpt) {
   initComponents();
   compte = cpt;
   // Afficher le type, le numéro et le solde du compte dans le cadre du composant
   Border cadre = BorderFactory.createTitledBorder("Compte " + cpt.getType()+ "n° " +
                                                   cpt.getNumero()+ " : " +
                                                   cpt.getSolde() + " Euros  ");
   boiteMenu.setBorder(cadre);
   setBounds(200, 320,500, 485);
   setVisible(true);
   // Afficher une image en fond de fenêtre
   ImageIcon iconPhoto = new ImageIcon("Ressources/FondCompte.png");
   photoFond.setIcon(iconPhoto);
}
private void debitBtnActionPerformed(java.awt.event.ActionEvent evt) {
   // Récupérer la valeur du montant de la transaction
   // Le montant devient négatif
   double valeur =  - Double.parseDouble(valeurAsaisir.getText());
   // Créer un objet de type LigneComptable
```

```
        LigneComptable tmp = new LigneComptable(valeur, motif, transaction);
        // Modifier la propriété ligne du compte en cours de traitement
        compte.setLigne(tmp);
        // Enregistrer le compte dans un fichier objet dont le nom porte le numéro du
        // compte
        FichierCompte fc = new FichierCompte();
        fc.ouvrir("Comptes/"+compte.getNumero()+".dat","W");
        fc.ecrire(compte);
        fc.fermer();
        // Fermer la fenêtre en cours sans quitter l'application
        this.dispose();
    }
    private void creditBtnActionPerformed(java.awt.event.ActionEvent evt) {
        // Récupérer la valeur du montant de la transaction
        double valeur =  Double.parseDouble(valeurAsaisir.getText());
        // Créer un objet de type LigneComptable
        LigneComptable tmp = new LigneComptable(valeur, motif, transaction);
        // Modifier la propriété ligne du compte en cours de traitement
        compte.setLigne(tmp);
        // Enregistrer le compte dans un fichier objet dont le nom porte le numéro du
        // compte
        FichierCompte fc = new FichierCompte();
        fc.ouvrir("Comptes/"+compte.getNumero()+".dat","W");
        fc.ecrire(compte);
        fc.fermer();
        // Fermer la fenêtre en cours sans quitter l'application
        this.dispose();
    }
```

Dans la classe `CompteEdit` :

```
    public CompteEdit(Compte cpt) {
        initComponents();
        // Afficher le type, le numéro et le solde du compte dans le cadre du composant
        Border cadre = BorderFactory.createTitledBorder("Compte " + cpt.getType()+ "n° " +
                                                        cpt.getNumero()+ " : " +
                                                        cpt.getSolde() + " Euros  ");
        entete.setBorder(cadre);
        setBounds(500, 230,500, 600);
        setVisible(true);
        // Afficher une image en fond de fenêtre
        ImageIcon iconPhoto = new ImageIcon("Ressources/FondEdit.png");
        photoFond.setIcon(iconPhoto);
        // Récupérer le contenu des lignes comptables
```

```
    ArrayList<LigneComptable> tmp = cpt.getLigne();
    int nbLignes = tmp.size();
    // Récupérer le solde du compte
    double solde = cpt.getSolde();
    // Si le solde est négatif l'afficher en rouge
    if (solde < 0) soldeValeur.setForeground(Color.RED);
    // Sinon l'afficher en noir
    else soldeValeur.setForeground(Color.BLACK);
    soldeValeur.setText(Double.toString(cpt.getSolde()));
    if (nbLignes > 0) {
       for (LigneComptable lc : tmp){
          // Pour chaque ligne comptable ,créer un objet de type LigneEdit
          // Ajouter l'objet au conteneur boiteLigne.
          boiteLigne.add(new LigneEdit(lc));
       }
    }
    else {
       System.out.println("La liste des formes est vide ");
    }
 }
 private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
    // Fermer la fenêtre en cours sans quitter l'application
    this.dispose();
 }
```

Dans la classe `LigneEdit` :

```
 public LigneEdit(LigneComptable tmp) {
    initComponents();
    setBounds(0, 0, 200, 70);
    setVisible(true);
    // Afficher le motif de la transaction dans le composant nommé labelMotif
    labelMotif.setText(tmp.getMotif());
    // Afficher le motif de la transaction dans le composant nommé labelTransaction
    labelTransaction.setText(tmp.getMode());
    // Afficher le montant de la transaction dans le composant nommé labelValeur
    labelValeur.setText(Double.toString(tmp.getValeur()));
 }
```