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

```
public class Premier {
 public static void main(String [] argument) {
    double a;
    System.out.print("Entrer une valeur : ") ;
    a = Lire.d() ;
    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

```
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 ;
   // 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 = Lire.d();
   // Calculer le périmètre du carré
   périmètre = côté * côté ;
   // Afficher le résultat
   System.out.print("Perimetre  : " + périmètre);
 }
}
```

Exercice I-4 : Écrire 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.

```
public class Rectangle  { // Nom à la classe
 public static void main(String [] argument)  {
   // Déclaration des variables
   double surface, largeur, longueur ;
   // Afficher un message à l'écran
   System.out.print("Valeur de la longueur : ");
   // Lire au clavier une valeur
   longueur = Lire.d();
   // Afficher un message à l'écran
   System.out.print("Valeur de la largeur : ");
   // Lire au clavier une valeur
   largeur = Lire.d();
   // 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 entier 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	7f

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 + 7	⇒	15	

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 = 7  Y = 3
```

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

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

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

Corrigé

```
public class Euro {
      public static void main (String [] argument) {
         double F, E = 0.0 ;
         double T = 6.55957 ;
         System.out.print("Nombre de Francs : ") ;
         F = Lire.d( ) ;
         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`. `x` est donc supérieure ou égale 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`. `x` est donc strictement inférieure à `0`. Le programme exécute le bloc `else`, c'est à dire l'instruction `r = Math.sqrt (-(-9))`

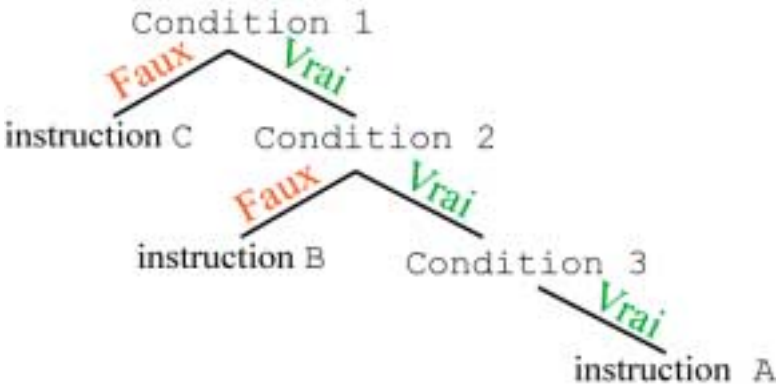
Exercice 3-2 : Construire une arborescence de choix

Corrigé

```
public class Maximum  {
  public static void main (String [] parametre)  {
   int première, deuxième, laPlusGrande ;
   System.out.print("Entrer une valeur :") ;
   première =  Lire.i() ;
   System.out.print("Entrer une deuxieme valeur :") ;
   deuxième =  Lire.i() ;
   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

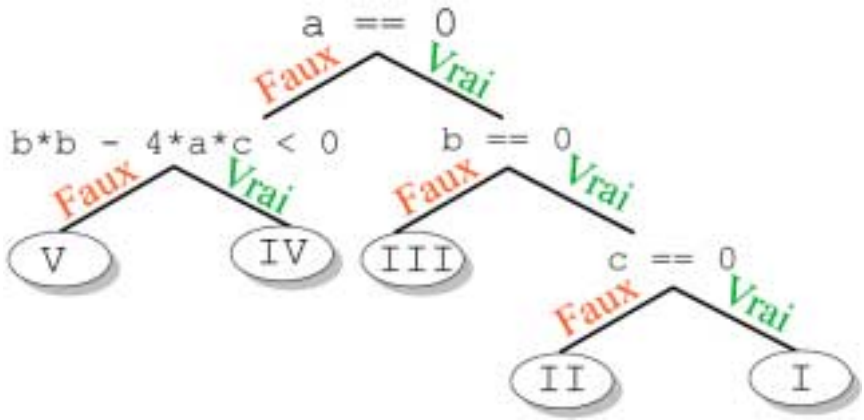


Les deux accolades fermantes situées après l'`instruction B` font que le bloc `else instruction C` est rattaché au premier `if (condition1)`.

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 calculs 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 :

```
public class SecondDegre {
   public static void main (String [] arg)  {
      double a, b, c, delta ;
      double x, x1, x2 ;
      System.out.print("Entrer une valeur pour a : ") ;
```

```
        a =  Lire.d() ;
        System.out.print("Entrer une valeur pour b : ") ;
        b =  Lire.d() ;
        System.out.print("Entrer une valeur pour c : ") ;
        c =  Lire.d() ;
        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` nécessaire 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 :

```
public class Calculette {
 public static void main( String [] argument) {
  int a, b;
  char opérateur;
```

```
  double calcul = 0;
  System.out.print("Entrer la premiere valeur : ");
  a = Lire.i();
  System.out.print("Entrer la seconde valeur : ");
  b = Lire.i();
  System.out.print("Type de l'operation : (+, -, *, /) : ");
  opérateur = Lire.c();
  switch (opérateur ) {
   case '+' : calcul = a + b;
              break;
   case '-' : calcul = a - b;
              break;
   case '/' : calcul = a / b;
              break;
   case '*' : calcul = a * b ;
              break;
    }
   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, à l'étiquette `'/'` de la structure `switch`. Examinons la correction :

```
public class Calculette     {
 public static void main( String [] argument)  {
  int a, b;
  char opérateur;
  double calcul = 0;
  boolean OK = true;
  System.out.print("Entrer la premiere valeur : ");
  a = Lire.i();
  System.out.print("Entrer la seconde valeur : ");
  b = Lire.i();
  System.out.print("Type de l'operation : (+, -, *, /) : ");
```

```
  opérateur = Lire.c();
switch (opérateur )  {
   case '+' : calcul = a + b;
              break;
   case '-' : calcul = a - b;
              break;
   case '/' : if ( b != 0) calcul = a / b;
                else OK = false;
              break;
   case '*' : calcul = a * b ;
              break;
   default : OK = false ;
    }
   if (OK) {
     System.out.print("Cette operation a pour resultat : ");
     System.out.println(a + " " +opérateur+ " "+ b + " =  " + calcul);
   }
   else  System.out.println("Operation non conforme !");
 }
}
```

A l'étiquette `'/'`, le programme vérifie si `b` est non nulle. Si tel est le cas, il exécute normalement l'instruction réalisant la division. Inversement si `b` est nulle, 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. L'image du drapeau provient des boites aux lettres américaines qui ont un drapeau qui change de position lorsque le facteur dépose une lettre dans la boite. Ce changement de position (d'état) signale alors, au destinataire qu'un nouveau courrier est arrivée.

Chapitre 4 : Faire des répétitions

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

Corrigé

```
public class Exercice1 {
 public static void main (String [] argument) {
  int a,b,r;
  System.out.print("Entrer un entier : ");
```

```
  a = Lire.i();
  System.out.print("Entrer un entier : ");
  b = Lire.i();
  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	-
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		

d.			
`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 et 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 :

```
public class Exercice2  {
  public static void main (String [] parametre)  {
   int valeur, laPlusGrande, laPlusPetite, laSomme = 0, leNombre = 0 ;
   double laMoyenne;
   System.out.print("Entrer une valeur :") ;
   valeur =  Lire.i() ;
   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 =  Lire.i() ;
    } 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 somme 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, 4-5, 4-6 et 4-7 : Comprendre la boucle `while`, traduire une marche à suivre en programme Java

Corrigé

1. Traduction de la marche à suivre en Java
   - Tirer au hasard un nombre entre 0 et 10.
     `i = (int) (10*Math.random()) ;`
   - Lire un nombre.
     `nombreLu = Lire.i();`
   - Tant que le nombre lu est différent du nombre tiré au hasard :
     `while ( nombreLu != i)`
     - Lire un nombre
       `nombreLu = Lire.i();`
     - Compter le nombre de boucle.
       `nbBoucle = nbBoucle + 1` (ou encore `nbBoucle++`)
   - Afficher un message de réussite ainsi que le nombre de boucles.
     `System.out.print("Bravo ! ");`
     `System.out.println("vous avez reussi en " + nbBoucle + " fois");`
2. Le code source complet :

```
public class Devinette {
 public static void main (String [] parametre) {
      int i, nombreLu = -1, nbBoucle = 0;
      i = (int) (10*Math.random());
      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 =  Lire.i();
```

```
         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.

3. Quelques améliorations :

```
public class Jeu  {
  public static void main (String [] parametre)     {
      int i, nombreLu = -1, nbBoucle = 0;
      // 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 =  Lire.i();
         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 4-8 : Comprendre la boucle `for`

Corrigé :

```
public class Exercice4 {
 public static void main (String [] paramètre) {
  long i, b = 1;
  int a;
  System.out.print("Entrer un entier :");
  a = Lire.i();
  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

d.			
`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 4-9 : 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 sucre au clavier, mettre la valeur dans `nombreSucre`.

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

c. Le paramètre qui permet de sucré 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 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()`

```
public class Exercice5 {
 public static void main (String [] arg)  {
   int nbCB, nbCheque, nbVirement, nbDebit;
   double résultat;
   System.out.print(" Nombre d'achat Cartes Bleues : ");
   nbCB = Lire.i();
   System.out.print(" Nombre de cheques emis : ");
   nbCheque = Lire.i();
   System.out.print(" Nombre de virements automatiques : ");
   nbVirement = Lire.i();
   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 ;
 }
}
```

Chapitre 6 : Fonctions, notions avancées

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

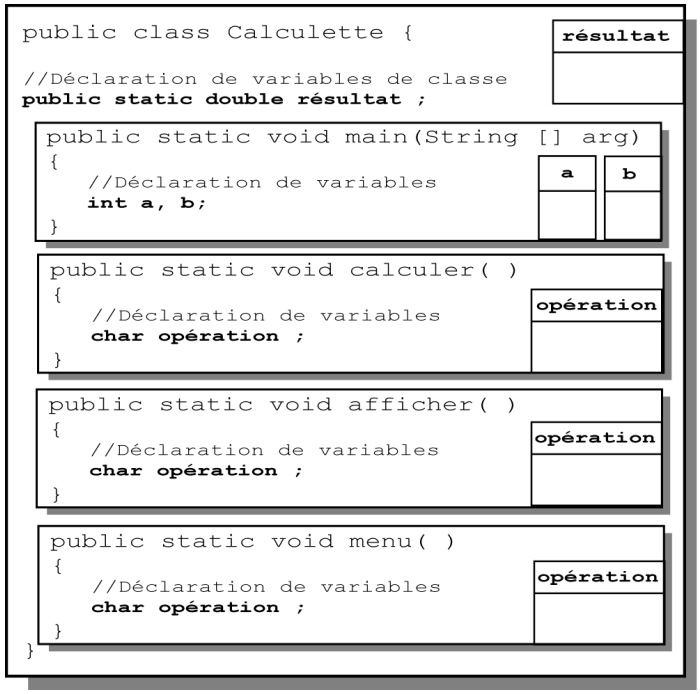
Corrigé :

```
public class Calculette {
 public static double résultat ;
 public static void main( String [] argument)  {
  int a, b;
   menu();
```

```
  System.out.println("Entrer la premiere valeur ");
  a = Lire.i();
  System.out.println("Entrer la seconde valeur ");
  b = Lire.i();
  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 ;
  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 = Lire.c() ;
 }
}
```

a. Les fonctions de la classe `Calculette` sont au nombre de 4 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 :

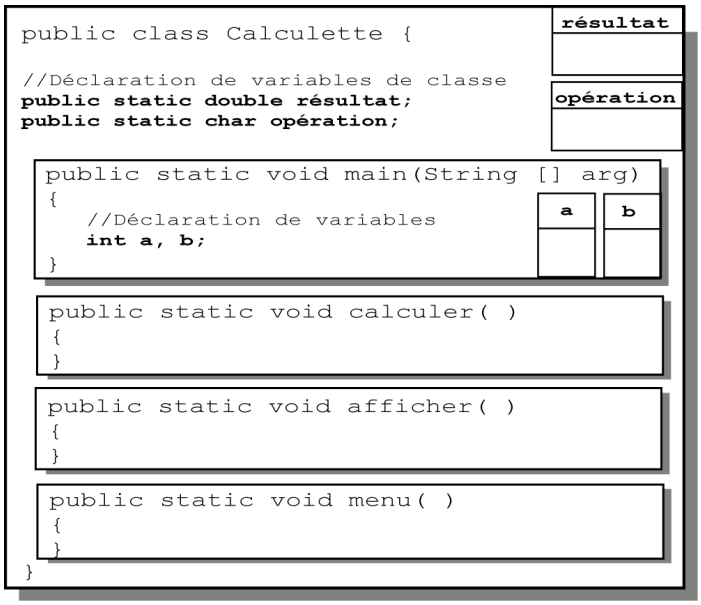
```
Calculette.java:16: Variable opération may not have been initialized.
Calculette.java:17: Undefined variable: a
Calculette.java:17: Undefined variable: b
Calculette.java:19: Undefined variable: a
Calculette.java:19: Undefined variable: b
```

```
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 confusions 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*).

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.

```
public class Exercice3 {
 public static void main( String [] argument){
  int a, b;
  char opérateur;
  double calcul;
  opérateur = menu();
  System.out.println("Entrer la premiere valeur ");
  a = Lire.i();
  System.out.println("Entrer la seconde valeur ");
  b = Lire.i();
  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 ;
 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 = Lire.c();
 return opération ;
 }
}
```

Chapitre 7 : Les classes et les objets

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

Corrigé :

```
public class Exercice1 {
   public static void main(String [] argument) {
      String s1 = "", s2 = "", s3 = "", s4 = "";
      int nbA = 0;
      // a. demande la saisie d'une phrase
      System.out.print("Entrez une phrase : ");
      s1 = Lire.S();
      // 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 = Lire.S();
        // 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 cette exercice, la démarche est identique. Seules, les techniques de comparaison diffèrent puisque les variables utilisées ne sont plus numériques mais, alphabétiques.

```
public class Exercice2 {
   public static void main(String [] argument) {
      String s1 = "", sPlusGrand = "", sPlusPetit = "";
      System.out.print("Entrez un mot : ");
      s1 = Lire.S();
      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 = Lire.S();
      } 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 : Créer une classe, des objets

Corrigé :

a. La classe `Personne` :

```
public class Personne  {
  public int âge;
  public String nom, prénom;
}
```

b. La classe `MesAmis` :

```
public class MesAmis {
   public static void main(String [] argument) {
      Personne Untel = new Personne();
      System.out.print ("Entrez votre nom : ");
      Untel.nom = Lire.S();
      System.out.print ("Entrez votre prenom : ");
      Untel.prénom = Lire.S();
      System.out.print ("Entrez votre age : ");
      Untel.âge = Lire.i();
      System.out.print("Vous vous appelez " + Untel.prénom + " ");
      System.out.print(Untel.nom + ". Vous avez " + Untel.âge + " ans.");
   }
}
```

Exercice 7-4 : Consulter les variables d'instance

Corrigé :

La classe `Personne` :

```
public class Personne   {
  public int âge;
  public String nom, prénom;
   // a. Affiche les caractéristiques d'une personne
   public void présentezVous() {
      System.out.print("Je m'appelle " + prénom + " " + nom);
      System.out.println( ". J'ai " + âge + " ans.");
   }
   // c. Retourne le nom suivi du prénom
   public String quelEstVotreNom() {
      return prénom + " " + nom;
   }
   / d.  Retourne l'âge
   public int quelEstVotreAge() {
      return âge;
   }
}
```

La classe `MesAmis` :

```
public class MesAmis {
   public static void main(String [] argument) {
      Personne Untel = new Personne();
      System.out.print ("Entrez votre nom : ");
      Untel.nom = Lire.S();
      System.out.print ("Entrez votre prenom : ");
```

```
        Untel.prénom = Lire.S();
        System.out.print ("Entrez votre age : ");
        Untel.âge = Lire.i();
        // b. Affiche les caractéristiques de l'objet Untel
        Untel.présentezVous();
        // e. Récupére les nom et prénom de l'objet Untel
        String n = Untel.quelEstVotreNom();
        // e. Récupére l'âge de l'objet Untel
        int a = Untel.quelEstVotreAge();
        // e. Affiche le nom et l'age d'Untel
        System.out.println(" Nom : " + n + " Age : " + a);
    }
}
```

Exercice 7-5 : 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 `FaireDesPoints`.

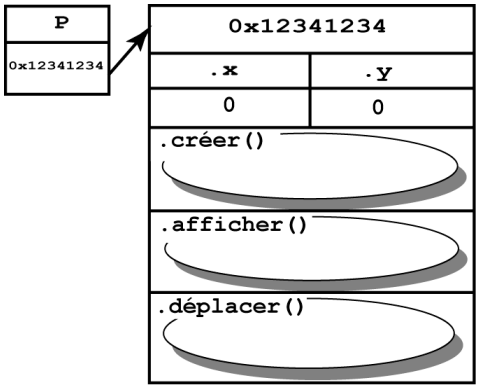
b. Le type `Point` est défini par la classe `Point`, décrite dans le fichier `Point.java`.

c. Les attributs de la classe `Point` sont les coordonnées (x, y) d'un point. Ils ont pour nom respectivement `x` et `y` déclarés comme variables d'instance de type `int`.

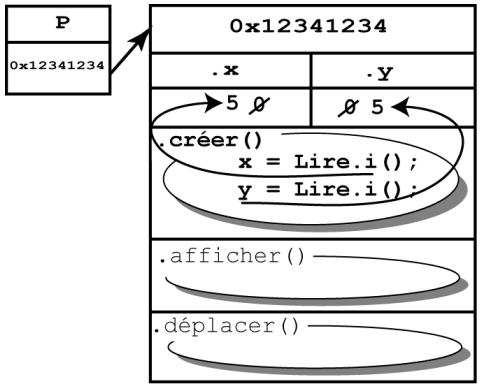
d. Trois méthodes sont définies dans la classe `Point`. Elles ont pour nom : `créer()`, `afficher()` et `déplacer()`.

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`.

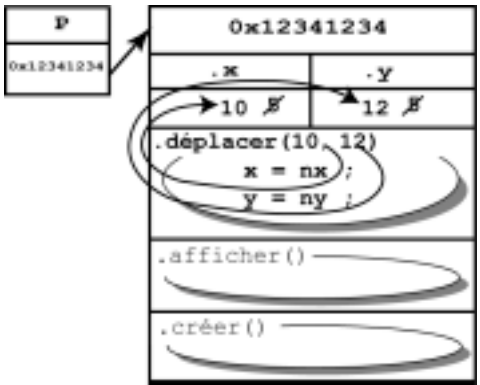
f. L'objet `P` est représenté de la façon suivante :



g. La méthode `créer()` est appelée par l'intermédiaire de l'objet `P`. Ce sont donc les données `x` et `y` de l'objet `P` qui mémorisent les valeurs saisies au clavier.



h. De la même façon, la méthode `déplacer()` est appelée par l'intermédiaire de l'objet `P`. Ce sont donc les données `x` et `y` de l'objet `P` qui prennent les valeurs transmises en paramètre de la méthode.



i. Le résultat final est le suivant :

```
x : 0 y : 0
Entrez l'abscisse : 5
Entrez l'ordonnée : 5
x : 5 y : 5
x : 10 y : 12
```

Chapitre 8 : Les principes du concept objet

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 :

```
valeur[0] = 1valeur[1] = 3valeur[2] = 5valeur[3] = 7valeur[4] =
9valeur[5] = 11
```

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 FaireDesCercles {
 public static void main(String [] arg) {
   System.out.print("Combien de Cercles : ");
   int nbCercle = Lire.i();

   if (nbCercle < 3) nbCercle = 4;
   // a. Créé 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 9-4 : Les tableaux à deux dimensions

Corrigé :

```
public class Exercice4 {
 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 9-5 : Pour mieux comprendre le mécanisme des boucles imbriquées (for – for)

Corrigé :

```
public class Exercice5 {
 public static void main (String [] parametre) {
  int i,j, N = 5;
  char C;
  System.out.print("Entrer un  caractere : ");
  C = Lire.c();
  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

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

d. Ce programme a pour résultat :

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

Exercice 9-6 : 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 Vecteurs

Corrigé :

```
import java.util.*;
import java.lang.Number.*;
public class Etudiant     {
   private String nom, prénom;
   // a. Définiton du vecteur notes
   private Vector notes;
   private double moyenne;
   public Etudiant()    {
      System.out.print("Entrer le nom de l'etudiant : ");
      nom = Lire.S();
      System.out.print("Entrer le prenom de l'etudiant : ");
      prénom = Lire.S();
      System.out.print("Combien de notes pour l'etudiant  ");
```

```
        System.out.print( prénom + " " + nom + " :  ");
        int nombre = Lire.i();
        notes = new Vector();
        for (int i = 0; i < nombre; i ++)  {
           System.out.print("Entrer la note  n° "+ (i+1)+ " :  ");
           // b. Mise en place des notes dans le vecteur
           //    Les notes sont préalablement transformées en objet Double
           notes.addElement(new Double(Lire.d()));
        }
        moyenne = calculMoyenne();
     }
     public void afficheUnEtudiant()  {
     // d. Mofification de l'affichage des notes
        System.out.print(" Les notes  de "+prénom+" "+nom+ " sont : ");
        int nbnotes = notes.size();
        for (int i = 0; i < nbnotes; i ++)
           System.out.print(" "+ notes.elementAt(i));
        System.out.println();
        System.out.println("Sa moyenne vaut  "+ moyenne);
     }
     private double calculMoyenne()  {
        double somme = 0.0;
        int nbnotes = notes.size();
        for(int i = 0; i < nbnotes; i++) {
     // c. Calcul de la moyenne, les notes sont transformées en type simple
     //    double, avant d'être accumulées dans la variable somme.
           Double d = (Double) notes.elementAt(i);
           double note = d.doubleValue();
           somme = somme + note ;
        }
        return somme/nbnotes;
     }
     public double quelleMoyenne() {
        return moyenne;
     }
  }
```

Exercice 10-2 : Comprendre les dictionnaires

Corrigé

a., b. et c. Cette méthode est à insérer dans la classe `Classe`

```
// 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 ! ");
}
```

Le second constructeur de la classe `Etudiant` se présente de la façon suivante :

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

d. La nouvelle classe `GestionClasse`

```
public class GestionClasse  {
 public static void main (String [] argument) {
    byte choix = 0 ;
    Classe C = new Classe();
    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");
     System.out.println("5. Modifie un etudiant");
     System.out.println("6. Sortir");
     System.out.println();
     System.out.print("Votre choix : ");
     choix  = Lire.b();
     switch (choix) {
       case 1 : C.ajouteUnEtudiant();
```

```
        break;
        case 2 :
         System.out.print("Entrer le prenom de l'etudiant a supprimer : ");
         prénom = Lire.S();
         System.out.print("Entrer le nom de l'etudiant a supprimer : ");
         nom = Lire.S();
         C.supprimeUnEtudiant(prénom, nom);
        break;
        case 3 : C.afficheLesEtudiants();
        break;
        case 4 :
         System.out.print("Entrer le prenom de l'etudiant recherche : ");
         prénom = Lire.S();
         System.out.print("Entrer le nom de l'etudiant recherche : ");
         nom = Lire.S();
         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 = Lire.S();
         System.out.print("Entrer le nom de l'etudiant a modifier : ");
         nom = Lire.S();
         C.modifieUnEtudiant(prénom, nom);
        break;
        case 6 : System.exit(0) ;
        default : System.out.println("Cette option n'existe pas ");
    }
   } while (choix != 6);
 }
}
```

Exercice 10-3 : 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)
```

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. Définir une case à cocher
  Checkbox CaseCoche = new Checkbox("Taille Fixe");
// b. Ajouter l'écouteur d'événement
  CaseCoche.addItemListener(new GestionAction(0, d));
// a. Ajouter 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();
  }
 }
```

Le projet : Gestion d'un compte bancaire

Chapitre 1 : Stocker une information

Déterminer les variables nécessaires au programme

Corrigé

Compte tenu des remarques précédentes, les variables utilisées pour le projets peuvent 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 :

```
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 ;
  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 = Lire.b();
//Option 1
System.out.print("Type du compte [Types possibles : ");
System.out.print("C(ourant), J(oint), E(pargne)] :");
typeCpte = Lire.c();
System.out.print("Numero du compte :");
numéroCpte = Lire.l();
System.out.print("Premiere valeur creditee  :");
val_courante = Lire.d();
//Si compte épargne
System.out.print("Taux de placement :     ");
taux = Lire.d();
//Option 2
//demande à l'utilisateur de saisir le numéro du compte à afficher
System.out.print ( " Quel compte souhaitez vous afficher ? : ");
numéroLu = Lire.l();
// 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("epargne  dont 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 alors, de la façon suivante :

```
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 ;
  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 = Lire.b();
  switch (choix)  {
    case 1 :
     System.out.print("Type du compte [Types possibles :" );
     System.out.print("C(ourant), J(oint), E(pargne)] :");
     typeCpte = Lire.c();
     System.out.print("Numero du compte :");
     numéroCpte = Lire.l();
     System.out.print("Premiere valeur creditee :");
     val_courante = Lire.d();
    //c. Si le type est un compte epargnele programme demande automatiquement le taux
     if ( typeCpte == 'E')   {
       System.out.print("Taux de placement :     ");
       taux = Lire.d();
     }
    break;
    case 2 :
      //demande à l'utilisateur de saisir le numéro du compte à afficher
      System.out.print ( " Quel compte souhaitez vous afficher ? : ");
      numéroLu = Lire.l();
```

```
     // 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("epargne  dont le taux est  " + taux);
        }
        System.out.println(" Premiere valeur creditee : " + val_courante);
      }
      else  {
        //d. Sinon, affiche un message indiquant que numéro du compte non valide.
        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 :
      //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

```
      if ( typeCpte == 'E')     {
         System.out.println("Taux de placement :     ");
         taux = Lire.d();
      }
    break;
    case 2 :
        System.out.print ( " Quel compte souhaitez vous afficher ? : ");
        numéroLu = Lire.l();
        // vérifie que le numéro du compte existe,
        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(" epargne  dont 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 chapitre précédents.

```
byte tmp;
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  = Lire.b();
```

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 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 reprend les instructions de sortie de programme utilisées dans le programme développé au cours des chapitre 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 reprend les instructions d'affichage utilisées dans le programme développé au cours des chapitre 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 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 ;
 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 = Lire.c();
      } while ( typeCpte != 'C' && typeCpte != 'J' && typeCpte != 'E');
      System.out.print("Numero du compte :");
      numéroCpte = Lire.l();
      System.out.print("Premiere valeur creditee :");
      val_courante = Lire.d();
      if ( typeCpte == 'E')  {
      System.out.print("Taux de placement :     ");
      taux = Lire.d();
      }
      break;
      case 2 :
      System.out.print ( " Quel compte souhaitez vous afficher ? : ");
      numéroLu = Lire.l();
      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(" epargne  dont 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

```
public class ProjetCh6 { //deClasse
 // 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
  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 = Lire.c();
         } while ( typeCpte != 'C' && typeCpte != 'J' && typeCpte != 'E');
      System.out.print("Numero du compte :");
      numéroCpte = Lire.l();
      System.out.print("Premiere valeur creditee :");
      val_courante  = Lire.d();
      if (typeCpte == 'E') {
         System.out.print("Taux de placement :  ");
         taux = Lire.d();
      }
      break;
      case 2 :
         System.out.print ( " Quel compte souhaitez vous afficher ? : ");
         numéroLu = Lire.l();
         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(" epargne  dont 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;
    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  = Lire.b();
    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 4 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(" epargne  dont 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 4 (le type, le numéro, le taux et la valeur courante). La fonction `créerCpte()` doit donc retourner ces quatre valeurs à la fonction `main()`.

b. Ces 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 changement 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 = Lire.c();
} 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 = Lire.S();
```

```
if (typeCpte.equalsIgnoreCase("Epargne")) {
   System.out.print("Taux de placement :     ");
   taux = Lire.d();
}
```

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;
   do {
         System.out.print("Type du compte [Types possibles :" );
         System.out.print("C(ourant), J(oint), E(pargne)] :");
         tmp = Lire.c();
   } 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 = Lire.S();
   if ( typeCpte.equalsIgnoreCase("Epargne"))  {
      System.out.print("Taux de placement :     ");
      taux = Lire.d();
   }
   System.out.print("Valeur initiale du compte : ");
   val_courante = Lire.d();
}
```

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

```
public  void 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

```
public class ProjetCh7  {
 // La fonction principale
 public static void main (String [] argument)  {
   byte choix = 0 ;
   String numéroLu = "";
// 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 = Lire.S();
      // 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;
    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  = Lire.b();
    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() {
       System.out.print("Entrer la valeur à créditer (+) ou débiter (-) : ");
       valeur = Lire.d();
       System.out.print("Date de l'opération [jj/mm/an] ");
```

```
        date = Lire.S();
        System.out.print("Motif de l'operation [S(alaire),");
        System.out.print(" L(oyer), A(limenation), D(ivers)] : ");
        motif =  Lire.S();
        System.out.print("Mode [C(B), N(° Cheque), V(irement ) ]  : ");
        mode = Lire.S();
    }
```

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

```
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;
   do {
         System.out.print("Type du compte [Types possibles :" );
         System.out.print("C(ourant), J(oint), E(pargne)] :");
         tmp = Lire.c();
   } 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 = Lire.S();
   if ( typeCpte.equalsIgnoreCase("Epargne"))  {
```

```
        System.out.print("Taux de placement :     ");
        taux = Lire.d();
    }
    System.out.print("Valeur initiale du compte : ");
    val_courante = Lire.d();
    }
    // 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;
    }
    public  void 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

```
public class ProjetCh7  {
 public static void main (String [] argument)  {
   byte choix = 0 ;
   String numéroLu = "";
   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 = Lire.S();
         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 = Lire.S();
         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 fonction  menuPrincipal()
  // La fonction sortir( ) {
  // La fonction alAide( ) {
}
```

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

b. Si l'utilisateur affiche les données d'un compte sans avoir créer 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.

```
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;
   do {
         System.out.print("Type du compte [Types possibles :" );
         System.out.print("C(ourant), J(oint), E(pargne)] : ");
```

```
            tmp = Lire.c();
      } 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 = Lire.S();
      if ( typeCpte.equalsIgnoreCase("Epargne"))  {
         System.out.print("Taux de placement :     ");
         taux = Lire.d();
      }
      System.out.print("Valeur initiale du compte : ");
      val_courante = Lire.d();
// 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;
      }
      public  void 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`

```
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()   {
      char tmp;
      do {
         System.out.print("Type du compte [Types possibles :" );
         System.out.print("C(ourant), J(oint), E(pargne)] : ");
         tmp = Lire.c();
      } 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 = Lire.S();
    if ( typeCpte.equalsIgnoreCase("Epargne"))  {
       System.out.print("Taux de placement :     ");
       taux = Lire.d();
    }
    System.out.print("Valeur initiale du compte : ");
    val_courante = Lire.d();
    nbLigneRéel = 0;
}
    public void créerLigne()  {
       ligne = new LigneComptable();
       ligne.créerLigneComptable();
       nbLigneRéel = 1;
       val_courante = val_courante + ligne.valeur;
    }
    public  void 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`

```
public class LigneComptable  {
    private double valeur;
    private String date;
    private String motif;
    private String mode;
    public void créerLigneComptable()  {
       System.out.print("Entrer la valeur à créditer (+) ou débiter (-) : ");
       valeur = Lire.d();
       System.out.print("Date de l'opération [jj/mm/an] ");
       date = Lire.S();
       System.out.print("Motif de l'operation [S(alaire),");
       System.out.print(" L(oyer), A(limenation), D(ivers)] : ");
       motif = Lire.S();
       System.out.print("Entrer le mode [C(B), N(° Cheque), V(irement ) ] : ");
       mode = Lire.S();
    }
```

```
    // 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 = Lire.S();
    // 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 = Lire.S();
       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;
        do {
            System.out.print("Valeur initiale du compte : ");
            tmpval= Lire.d();
        } while ( tmpval <= 0);
        return tmpval;
    }
```

b. La méthode `contrôleMode()` :

```
    private String contrôleMode() {
        String tmpS = "";
        char tmpc ;
        do {
            System.out.print("Mode [C(B), N(° Cheque), V(irement ) ]  : ");
            tmpc = Lire.c();
        } 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 ;
        do {
            System.out.print("Motif de l'operation [S(alaire),");
            System.out.print(" L(oyer), A(limenation), D(ivers)] : ");
            tmpc = Lire.c();
        } while ( tmpc != 'S' && tmpc != 'L' && tmpc != 'A' && tmpc != 'D');
```

```
        switch (tmpc) {
           case 'S' : tmpS = "Salaire";
           break;
           case 'L' : tmpS = "Loyer";
           break;
           case 'A' : tmpS = "Alimenation";
           break;
           case 'D' : tmpS = "Divers";
           break;
        }
        return tmpS;
     }
```

c. La méthode `créerCpte()` :

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

La méthode `créerLigneComptable()` :

```
     public void créerLigneComptable()  {
        System.out.print("Entrer la valeur à créditer (+) ou débiter (-) : ");
        valeur = Lire.d();
        System.out.print("Date de l'opération [jj/mm/an] ");
        date = Lire.S();
        // 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() {
   System.out.print("Entrer la valeur à créditer (+) ou débiter (-) : ");
   valeur = Lire.d();
   System.out.print("Date de l'opération [jj/mm/an] ");
   date = Lire.S();
   motif = contrôleMotif();
   mode = contrôleMode();
}
```

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

```
public Compte () {
   typeCpte = contrôleType();
   System.out.print("Numéro du compte : ");
   numéroCpte = Lire.S();
   if ( typeCpte.equalsIgnoreCase("Epargne"))   {
      System.out.print("Taux de placement :     ");
      taux = Lire.d();
   }
   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 ="";
   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 second constructeur, comme suit :

```
public static void main (String [] argument) {
    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 premier constructeur. 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 double  taux ;
    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 :

```
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 ;
   public  Compte ()    {
      typeCpte = contrôleType();
      System.out.print("Numéro du compte : ");
      numéroCpte = Lire.S();
      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()    {
      char tmpc;
      String tmpS = "";
      do {
            System.out.print("Type du compte [Types possibles :" );
            System.out.print("C(ourant), J(oint)] : ");
            tmpc = Lire.c();
      } 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()  {
        double tmp, tmpval;
        do {
            System.out.print("Valeur initiale du compte : ");
            tmpval= Lire.d();
        } while ( tmpval <= 0);
        return tmpval;
    }
    public void créerLigne()   {
        ligne = new LigneComptable();
        nbLigneRéel = 1 ;
        val_courante = val_courante + ligne.quelleValeur();
    }
    public  void 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() {
        double tmp;
        do {
            System.out.print("Taux de placement :     ");
            tmp = Lire.d();
        } 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 modifier de sorte qu'il gère la création d'un compte épargne. Cette gestion est réalisée comme suit :

```
public Compte(String type)  {
   if (type.equalsIgnoreCase("Epargne")) {
      typeCpte = type;
      System.out.print("Numéro du compte : ");
      numéroCpte = Lire.S();
      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 (Lire.c() == '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 ()  {
   typeCpte = contrôleType();
   System.out.print("Numéro du compte : ");
   numéroCpte = Lire.S();
   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;
}

    public  Compte( String type)  {
    if (type.equalsIgnoreCase("Epargne")) {
       typeCpte = type;
       System.out.print("Numéro du compte : ");
       numéroCpte = Lire.S();
       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 ligne à chaque ligne créée
       nbLigneRéel++;
       // b. Si le nombre de ligne est < 10, création d'une nouvelle ligne
       if (nbLigneRéel < NBLigne)
          ligne [nbLigneRéel] = new LigneComptable();
       // c. Si le nombre de ligne 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 Hashtable liste;
   // a. Le constructeur de la classe ListeCompte fait appel au constructeur
   //    de la classe Hashtable
   public ListeCompte()   {
      liste = new Hashtable();
   }
   // 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)   {
          Enumeration enumCompte = liste.keys();
          while (enumCompte.hasMoreElements()) {
             String clé = (String) enumCompte.nextElement();
             Compte c = (Compte) liste.get(clé);
             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

```
public class ProjetCh10 {
   public static void main (String [] argument) {
      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 (Lire.c() == 'o')    C.ajouteUnCompte("E") ;
              else     C.ajouteUnCompte("A");
           break;
           case 2 :
              System.out.print ( "Quel compte souhaitez vous afficher ? : ");
              numéroLu = Lire.S();
              // 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 = Lire.S();
              // a. ajouter une ligne à un compte
              C.ajouteUneLigne(numéroLu);
           break;
           case 5 :
              System.out.print ( "Quel compte souhaitez vous supprimer ? : ");
              numéroLu = Lire.S();
              // 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;
   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  = Lire.b();
   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
  private  String nomDuFichier = "Compte.dat";
  private   ObjectOutputStream fWo;
  private    ObjectInputStream 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+" : Erreur  de 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 (*Écrire la méthode* `contrôleDate()`)

Corrigé

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

```
    private String contrôleDate() {
       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(Lire.S());
          }
          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 besoins 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;
   }
```

Elles 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 stockées dans des variables de double précision
  private double prctDiv, prctLoy, prctAli;
  public Stat(Compte c) {
    cpte = c;
  }
  private  double pourcentage(double nb, double t){
     double    prct = 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()` :

```
public class Projet  {
public static void main (String [] argument) {
   byte choix = 0 ;
   String numéroLu = "";
   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 = Lire.S();
    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 du  premier 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");
   show();
  }
}
```

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 :

De plus, lorsque le numéro est 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.show();
   }
   public void actionPerformed(ActionEvent evt) {
```

```
        String numéro = réponse.getText();
        Compte cpte =  new Compte("");
        // Si le numéro saisi existe dans la liste des compte
        cpte = lc.quelCompte(numéro);
        if (cpte != null) {
        // Si le numéro saisi existe dans la liste des compte
        // 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);
     }
  }
```