Chapitre 2

Les instructions de contrôle

Connaissances requises

- Instructions simples, instructions structurées, instructions composées (bloc)
- L'instruction *if* ; cas des *if* imbriqués
- L'instruction *switch* ; l'étiquette *default*
- L'instruction *do while*
- L'instruction *while*
- L'instruction *for* ; initialisation avec éventuelle déclaration, condition d'arrêt, incrémentation
- Les instructions de branchement inconditionnel *break* et *continue* avec ou sans étiquette

Note : on suppose qu'on dispose d'une classe nommée *Clavier*, comportant (entre autres) des méthodes (statiques) de lecture au clavier d'informations de type *int* (*lireInt*), *float* (*lireFloat*), *double* (*lireDouble*) et *char* (*lireChar*). Cette classe est présente sur le site Web d'accompagnement et sa liste est fournie en Annexe D.

11 Syntaxe de if et de switch

Quelles erreurs ont été commises dans chacun des groupes d'instructions suivants. On suppose que les variables concernées sont d'un type primitif numérique et qu'elles ont été correctement déclarées (un groupe ne comporte aucune erreur) :

```
// groupe 1
if (a < b) System.out.println ("ascendant")
      else System.out.println ("non ascendant") ;

// groupe 2
if (a < b) { System.out.println ("ascendant) ; max = b }

// groupe 3
int n, p ;
   .....
switch (n) { case 2 : System.out.println ("petit") ;  break ;
             case p : System.out.println ("limite") ; break ;
           }

// groupe 4
int n ;
final int LIMITE = 20 ;
   .....
switch (n) { case LIMITE-1 : System.out.println ("un peu trop petit") ; break ;
             case LIMITE   : System.out.println ("OK") ;                 break ;
             case LIMITE+1 : System.out.println ("un peu trop grand") ; break ;
           }
```

Solution

Groupe 1

Il manque un point-virgule à la fin du premier appel de *System.out.println* :

```
if (a < b) System.out.println ("ascendant") ;
      else System.out.println ("non ascendant") ;
```

Groupe 2

Il manque un point-virgule à la fin de la deuxième instruction du bloc :

```
if (a < b) { System.out.println ("ascendant) ; max = b ; }
```

Groupe 3

Les valeurs utilisées dans les étiquettes de la forme *case xxx* doivent être des expressions constantes, ce qui n'est pas le cas de *p*.

Groupe 4

Aucune erreur. Les expressions telles que *LIMITE-1* étant bien cette fois des expressions constantes.

12 Rôle de l'instruction switch

Soit le programme suivant[a] :

```
public class ExoII2
{ public static void main(String[] args)
  { int n ;
    n = Clavier.lireInt() ;
    switch (n)
    { case 0 :  System.out.println ("Nul") ;
      case 1 :
      case 2 :  System.out.println ("Petit") ;
                break ;
      case 3 :
      case 4 :
      case 5 :  System.out.println ("Moyen") ;
      default : System.out.println ("Grand") ;
    }
  }
}
```

Quels résultats affiche-t-il lorsqu'on lui fournit en donnée :

1. la valeur 0,
2. la valeur 1,
3. la valeur 4,
4. la valeur 10,
5. la valeur -5.

a. Il utilise la classe *Clavier* (voir note en début de chapitre).

Solution

```
// avec la valeur 0
Nul
Petit

// avec la valeur 1
Petit

// avec la valeur 4
Moyen
Grand

// avec la valeur 10
Grand

// avec la valeur -5
Grand
```

13 Syntaxe des boucles

Quelles erreurs ont été commises dans chacune des instructions suivantes ?

```
do n++ while (n<10) ;                          // instruction 1
do while ( (n = Clavier.lireInt()) != 10) ;    // instruction 2
do ; while (true) ;                            // instruction 3
do {} while (false) ;                          // instruction 4
```

Solution

Instruction 1

Il manque un point-virgule :

```
do n++ ; while (n<10) ;
```

Instruction 2

Il manque une instruction (même vide) après le mot *do*, par exemple :

```
do ; while ( (n = Clavier.lireInt()) != 10) ;
```

ou :

```
do {} while ( (n = Clavier.lireInt()) != 10) ;
```

Instruction 3

Aucune erreur de compilation ne sera détectée. Mais on est en présence d'une boucle infinie.

Instruction 4

Aucune erreur de compilation ne sera détectée. Mais l'instruction ne sert à rien.

14 Comparaison entre for, while et do... while

Soit le programme suivant[a] :

```
public class ExoII4a
{ public static void main(String[] args)
  { int i, n, som ;
    som = 0 ;
    for (i=0 ; i<4 ; i++)
    { System.out.println ("donnez un entier ") ;
      n = Clavier.lireInt() ;
      som += n ;
    }
    System.out.println ("Somme : " + som) ;
  }
}
```

Écrire un programme réalisant la même chose en employant à la place de l'instruction *for* :

1. une instruction *while*,
2. une instruction *do... while*.

a. Il utilise la classe *Clavier* (voir note en début de chapitre).

Solution 1 Avec une instruction *while* :

```
public class ExoII4b
{ public static void main(String[] args)
  { int i, n, som ;
    som = 0 ;
    i = 0 ;
    while (i<4)
    { System.out.println ("donnez un entier ") ;
      n = Clavier.lireInt() ;
      som += n ;
      i++ ;
    }
    System.out.println ("Somme : " + som) ;
  }
}
```

Solution 2 Avec une instruction *do... while* :

```
public class ExoII4c
{ public static void main(String[] args)
  { int i, n, som ;
    som = 0 ;
```

```
    i = 0 ;
   do
    { System.out.println ("donnez un entier ") ;
      n = Clavier.lireInt() ;
      som += n ;
      i++ ;
    }
    while (i<4) ;
    System.out.println ("Somme : " + som) ;
  }
}
```

15 Rupture de séquence avec break et continue

Quels résultats fournit le programme suivant ?

```
public class ExoII5
{ public static void main(String[] args)
  { int n=0 ;
    do
    { if (n%2==0) { System.out.println (n + " est pair") ;
                    n += 3 ;
                    continue ;
                  }
      if (n%3==0) { System.out.println (n + " est multiple de 3") ;
                    n += 5 ;
                  }
      if (n%5==0) { System.out.println (n + " est multiple de 5") ;
                    break ;
                  }
      n += 1 ;
    }
    while (true) ;
  }
}
```

Solution

```
0 est pair
3 est multiple de 3
9 est multiple de 3
15 est multiple de 3
20 est multiple de 5
```

16 Boucle while, opérateurs d'affectation élargie et d'incrémentation (1)

Quels résultats fournit le programme suivant ?

```
public class ExoII6
{ public static void main(String[] args)
  { int n, p ;

    n = 0 ;
    while (n<=5) n++ ;
    System.out.println ("A : n = " + n) ;

    n = p = 0 ;
    while (n<=8) n += p++ ;
    System.out.println ("B : n = " + n) ;

    n = p = 0 ;
    while (n<=8) n += ++p ;
    System.out.println ("C : n = " + n) ;

    n = p = 0 ;
    while (p<=5) n += p++ ;
    System.out.println ("D : n = " + n) ;

    n = p = 0 ;
    while (p<=5) n+= ++p ;
    System.out.println ("D : n = " + n) ;
  }
}
```

Solution

```
A : n = 6
B : n = 10
C : n = 10
D : n = 15
D : n = 21
```

17 Boucle while, opérateurs d'affectation élargie et d'incrémentation (2)

Quels résultats fournit le programme suivant ?

```
public class ExoII7
{ public static void main(String[] args)
  { int n, p ;

    n=p=0 ;
    while (n<5) n+=2 ; p++ ;
    System.out.println ("A : n = " + n + ", p = " + p) ;

    n=p=0 ;
    while (n<5) { n+=2 ; p++ ; }
    System.out.println ("B : n = " + n + ", p = " + p) ;
  }
}
```

Solution

```
A : n = 6, p = 1
B : n = 6, p = 3
```

18 Syntaxe générale des trois parties d'une boucle for

Quels résultats fournit le programme suivant ?

```
public class ExoII8
{ public static void main (String[] args)
  { int i, n ;

    for (i=0, n=0 ; i<5 ; i++) n++ ;
    System.out.println ("A : i = " + i + ", n = " + n) ;

    for (i=0, n=0 ; i<5 ; i++, n++) {}
    System.out.println ("B : i = " + i + ", n = " + n) ;
```

```
        for (i=0, n=50 ; n>10 ; i++, n-= i ) {}
        System.out.println ("C : i = " + i + ", n = " + n) ;

        for (i=0, n=0 ;
          i<3 ; i++, n+=i, System.out.println ("D : i = " + i + ", n = " + n)) ;
        System.out.println ("E : i = " + i + ", n = " + n) ;
      }
    }
```

Solution

```
A : i = 5, n = 5
B : i = 5, n = 5
C : i = 9, n = 5
D : i = 1, n = 1
D : i = 2, n = 3
D : i = 3, n = 6
E : i = 3, n = 6
```

19 Synthèse : calcul d'une suite de racines carrées

Écrire un programme qui calcule les racines carrées de nombres fournis en donnée. Il s'arrêtera lorsqu'on lui fournira la valeur 0[a]. Il refusera les valeurs négatives. Son exécution se présentera ainsi :

```
donnez un nombre positif : 2
sa racine carree est : 1.4142135623730951
donnez un nombre positif : -3
svp positif
donnez un nombre positif : 5
sa racine carree est : 2.23606797749979
donnez un nombre positif : 0
```

a. Rappelons que la méthode *Math.sqrt* fournit un résultat de type *double* correspondant à la valeur de type *double* fournie en argument.

Solution Il existe beaucoup de rédactions possibles. En voici trois :

```
public class RacCara
{ public static void main (String[] args)
  { double x ;
    do
      {  System.out.print ("donnez un nombre positif : ") ;
         x = Clavier.lireDouble () ;
         if (x < 0) System.out.println ("svp positif") ;
         if (x <=0) continue ;
         System.out.println ("sa racine carree est : " + Math.sqrt (x) ) ;
      }
    while (x != 0) ;
  }
}

public class RacCarb
{ public static void main (String[] args)
  { double x ;
    do
    { System.out.print ("donnez un nombre positif : ") ;
      x = Clavier.lireDouble() ;
      if (x < 0) { System.out.println ("svp positif") ;
                   continue ;
                 }
     if (x>0) System.out.println ("sa racine carree est : " + Math.sqrt (x) ) ;
    }
    while (x != 0) ;
  }
}

public class RacCarc
{ public static void main (String[] args)
  { double x ;
    do
    { System.out.print ("donnez un nombre positif : ") ;
      x = Clavier.lireDouble() ;
      if (x < 0) { System.out.println ("svp positif ") ;
                   continue ;
                 }
      if (x>0) System.out.println ("sa racine carree est : " + Math.sqrt (x)) ;
      if (x==0) break ;
    }
    while (true) ;
  }
}
```

20 Synthèse : calcul de la valeur d'une série

Écrire un programme calculant la somme des *n* premiers termes de la "série harmonique", c'est-à-dire la somme :

1 + 1/2 + 1/3 + 1/4 + ... + 1/n

La valeur de *n* sera lue en donnée[a].

a. On pourra utiliser la classe *Clavier* (voir note en début de chapitre).

Solution

```
public class Serie
{ public static void main (String[] args)
  { int nt ;          // nombre de termes de la serie harmonique
    float som ;       // pour la somme de la serie
    int i ;

    do
      { System.out.print ("combien de termes : ") ;
        nt = Clavier.lireInt() ;
      }
    while (nt<1) ;
    for (i=1, som=0 ; i<=nt ; i++) som += (float)1/i ;
    System.out.println ("Somme des " + nt + " premiers termes = " + som) ;
  }
}
```

Remarque

1. Rappelons que dans :

   ```
   som += (float)1/i
   ```

   l'opérateur *float* porte sur l'entier 1. Le premier opérande de l'opérateur / est donc de type *float* ; par conséquent, son second opérande sera soumis à une promotion numérique en *float*, avant qu'on ne procède à la division.

   Notez qu'il faut éviter d'écrire :

   ```
   som += 1/i
   ```

   En effet dans ce cas l'opérateur / porterait sur deux entiers et correspondrait à la division entière. Le résultat serait toujours nul (sauf pour *i = 1*).

   De même, en écrivant :

   ```
   som += (float)(1/i)
   ```

   le résultat ne serait pas plus satisfaisant puisque la conversion en flottant n'aurait lieu qu'après la division (en entier).

Remarque

En revanche, on pourrait écrire :

```
som += 1.0f/i
```

2. On peut améliorer la précision du résultat en effectuant la somme "à l'envers", c'est-à-dire en allant de *n* vers 1 et non pas de 1 vers *n*. La différence ne deviendra cependant perceptible que pour de grandes valeurs de *n*.

21 Synthèse : dessin d'un triangle en mode texte

Écrire un programme qui affiche un triangle isocèle formé d'étoiles. La hauteur du triangle (c'est-à-dire son nombre de lignes) sera fourni en donnée[a], comme dans l'exemple ci-dessous. On s'arrangera pour que la dernière ligne du triangle s'affiche sur le bord gauche de l'écran.

```
combien de lignes ? 8
       *
      ***
     *****
    *******
   *********
  ***********
 *************
***************
```

a. On pourra utiliser la classe *Clavier* (voir note en début de chapitre).

Solution

```
public class Dessin
{
  public static void main (String[] args)
  {
    int nLignes ;              // nombre total de lignes
    int numLigne ;             // compteur de ligne
    int nEspaces ;             // nombre d'espaces precedent une etoile
    final char cRempli = '*' ;  // caractere de remplissage (ici etoile)
    int j ;

    System.out.print ("combien de lignes ? ") ;
    nLignes = Clavier.lireInt () ;
```

```
    for (numLigne=0 ; numLigne<nLignes ; numLigne++)
    { nEspaces = nLignes - numLigne - 1 ;
      for (j=0 ; j<nEspaces ; j++)  System.out.print (' ') ;
      for (j=0 ; j<2*numLigne+1 ; j++) System.out.print (cRempli) ;
      System.out.println () ;
    }
  }
}
```

22 Synthèse : calcul de combinaisons

Écrire un programme qui affiche toutes les manières possibles d'obtenir un franc avec des pièces de 2 centimes, 5 centimes et 10 centimes. Dire combien de possibilités ont ainsi été trouvées. Les résultats seront présentés ainsi :

```
1 F = 50 X 2c
1 F = 45 X 2c + 2 X 5c
1 F = 40 X 2c + 4 X 5c
1 F = 35 X 2c + 6 X 5c
1 F = 30 X 2c + 8 X 5c
1 F = 25 X 2c + 10 X 5c
1 F = 20 X 2c + 12 X 5c
1 F = 15 X 2c + 14 X 5c
   .....
1 F = 15 X 2c + 7 X 10c
1 F = 10 X 2c + 2 X 5c + 7 X 10c
1 F = 5 X 2c + 4 X 5c + 7 X 10c
1 F = 6 X 5c + 7 X 10c
1 F = 10 X 2c + 8 X 10c
1 F = 5 X 2c + 2 X 5c + 8 X 10c
1 F = 4 X 5c + 8 X 10c
1 F = 5 X 2c + 9 X 10c
1 F = 2 X 5c + 9 X 10c
1 F = 10 X 10c
En tout, il y a 66 facons de faire 1 F
```

Solution

```
public class Combis
{
  public static void main (String[] args)
  {
    int nbf ;        /* compteur du nombre de façons de faire 1 F */
    int n10 ;        /* nombre de pièces de 10 centimes */
    int n5 ;         /* nombre de pièces de 5 centimes */
    int n2 ;         /* nombre de pièces de 2 centimes */

    nbf = 0 ;
    for (n10=0 ; n10<=10 ; n10++)
      for (n5=0 ; n5<=20 ; n5++)
      for (n2=0 ; n2<=50 ; n2++)
      if ( 2*n2 + 5*n5 + 10*n10 == 100)
         { nbf ++ ;
           System.out.print ("1 F = ") ;
           if (n2 != 0)    System.out.print (n2 + " X 2c") ;
           if (n5 != 0)  { if (n2 != 0) System.out.print (" + ") ;
                           System.out.print (n5 + " X 5c")  ;
                         }
          if (n10 != 0) { if ((n2 != 0) || (n5 != 0)) System.out.print (" + ") ;
                           System.out.print (n10 + " 10c") ;
                         }
           System.out.println () ;
         }
    System.out.println ("En tout, il y a " + nbf + " facons de faire 1 F") ;
  }
}
```