Université Denis Diderot - DEUG MIAS/MASS - IF121
Corrigé du partiel du 27/11/2004

Exercice 1.

L'exécution du programme :

```
class Affectations{
   public static void main (String[ ] st) {
       int x; int y = 4; int z = 6;
       double r1, r2;
       x = y + 2; Deug.println(x);
       x = x + 4; Deug.println(x);
       x = z / y; Deug.println(x);
       r1 = z / y; Deug.println(r1);
       r1 = y; r2 = z / r1; Deug.println(r2);
   }
}
```

donne

```
$ java Affectations
6    <-- x prend  comme valeur la valeur de y (4) augmentée de 2, soit 6
10   <-- x prend comme valeur celle actuelle de x (6) augmentée de 4, soit 10
1    <-- x a comme valeur le quotient de la division entière de 6 par 4, soit 1
1.0  <-- on fait la même division entière, mais le résultat est de type double
1.5  <-- la division est faite sur le type double (6/4 = 1.5) et affectée à r2
         qui est une variable de type double
$
```

Exercice 2.

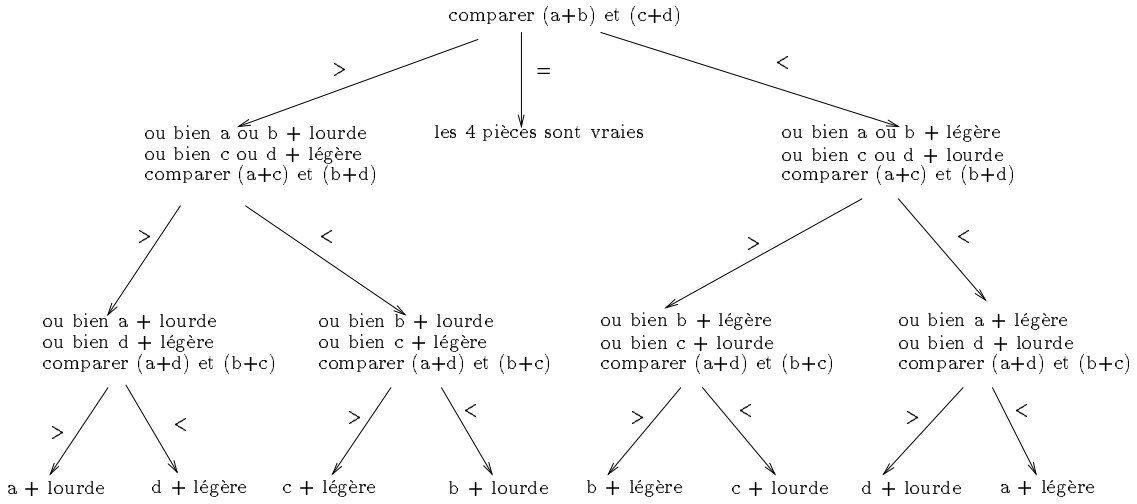
2.1. Trouver si une pièce parmi trois est fausse.
Première formulation :

```
on compare deux pièces, par exemple a et b
si a est égal à b
   alors on compare c à a (ou à b)
         si a est égal à c
            alors il n'y a pas de fausse pièce
            sinon c est une fausse pièce
   sinon a ou b est une fausse pièce et c est donc une vraie pièce
         on compare c à a (ou à b)
         si c est égale à a
            alors b est une fausse pièce
            sinon a est une fausse pièce
```

Deuxième formulation sous forme d'une séquence Java:

```
   ........
  if(a == b)
     if(a == c)
        Deug.println("les trois pièces sont vraies");
     else
        Deug.println("c est une fausse pièce");
  else
     if(a == c)
        Deug.println("b est une fausse pièce");
     else
        Deug.println("a est une fausse pièce");
   ........
```

2.2. Trouver si une pièce est fausse parmi 4 par pesées de paires de pièces.
Solution formulée sous forme d'un arbre de décision décrivant une suite logique de comparaisons et les conclusions qu'on peut tirer de leurs résultats:



Exercice 3.

3.1. Exécution du programme Java:

```
import fr.jussieu.script.*;
class Exo3 {
    public static void main(String[ ] st){
       byte b1 = 108, b2 = 79;
       Deug.println(b1 + b2); }
}
```

Les valeurs des deux variables `b1` et `b2` sont converties en valeurs de type `int` et le résultat est de type `int`. On obtient donc l'affichage de `187`.

3.2. Soit le programme Java :

```
import fr.jussieu.script.*;
class Exo3bis {
    public static void main(String[ ] st){
       byte b, b1 = 108, b2 = 79;
       b = b1 + b2;   // ligne provoquant une erreur de compilation
       Deug.println(b); }
}
```

a) Sa compilation donne :

```
$ javac Exo3bis.java
exo3bis.java:5: possible loss of precision
found   : int
required: byte
       b = b1 + b2;   // ligne provoquant une erreur de compilation
              ^
1 error
$
```

La somme `b1 + b2` est réalisée sur des valeurs entières et donne une valeur entière. Elle est susceptible de conduire à une valeur inadaptée au type de la variable `b` (type `byte`).

b) Pour rendre la compilation possible, il faut forcer le type du résultat à celui de la variable recevant la valeur. La ligne doit donc être modifiée en :

```
       b = (byte) (b1 + b2);
```

La compilation se passe bien :

```
$ javac Exo3bis.java
$
```

et son exécution produit :

```
$ java exo3bis
-69
$
```

On obtient une valeur négative car la somme `b1 + b2` donne un nombre compris entre `128` et `255`, qui correspond à une valeur négative (bit le plus à gauche égal à `1`). Cette valeur correspond de la représentation d'un nombre négatif égal à `-(256-(b1+b2))`, c'est-à-dire ici à `-(256-187)`, soit `-69`.

Exercice 4.

4.1. Fonction testant l'appartenance d'un point à une grille :

```
   static boolean estSurLaGrille(int x0, int y0, int longueur, int hauteur,
                                 int x, int y)
   {
       return ((x >= x0) && (x <= x0 + longueur)
                && (y >= y0) && (y <= y0 + hauteur));
   }
```

4.2. Fonction testant si un point est sur la bordure d'une grille :

```
static boolean estAuBord(int x0, int y0, int longueur, int hauteur,
                                 int x, int y)
{
  return(    estSurLaGrille(x0, y0, longueur, hauteur, x, y)
          && ( (x == x0) || (x == x0 + longueur) ||
               (y == y0) || (y == y0 + hauteur)
             )
        );
}
```

4.3. Fonction testant si un point est un des coins d'une grille :

```
static boolean estUnCoin(int x0, int y0, int longueur, int hauteur,
                                 int x, int y)
{
 if ((x == x0) || (x == x0 + longueur))
     return  (y == y0) || (y == y0 + hauteur);
 else
     return false;
// return ((x==x0)||(x==x0+longueur)) && ((y==y0)||(y == y0 + hauteur));
}
```

4.4. Fonction renvoyant le nombre de voisins sur une grille d'un point sur cette grille :

```
static int nombreDeVoisins(int x0, int y0, int longueur, int ,
                                   int x, int y)
{
  if(!estSurLaGrille(x0, y0, longueur, hauteur, x, y)) return 0;
  if (estUnCoin(x0, y0, longueur, hauteur, x, y)) return 3;
  if (estAuBord(x0, y0, longueur, hauteur, x, y)) return 5;
  return 8;
}
```

4.5. Programme lisant la définition d'une grille et les coordonnées d'un point et affichant son nombre de voisins sur la grille :

```
public static void main(String[ ] st)
{
  int x0, y0, x, y, longueur, hauteur;
  Deug.print ("Entrer x0 : "); x0 = Deug.readInt();
  Deug.print ("Entrer y0 : "); y0 = Deug.readInt();
  Deug.print ("Entrer longueur : "); longueur = Deug.readInt();
  Deug.print ("Entrer hauteur : "); hauteur = Deug.readInt();
  Deug.print ("Entrer x : "); x = Deug.readInt();
  Deug.print ("Entrer y : "); y = Deug.readInt();
  Deug.print("Le point (" + x + "," +y + ") a ");
  Deug.print(nombreDeVoisins(x0,y0,longueur,hauteur,x,y));
  Deug.println(" voisin(s) sur la grille donnée");
}
```

Exercice 5.

Fonction testant si un nombre est la médiane des éléments d'un tableau :

```
static boolean estMediane(int[ ] tab, int x)
{
   int pluspetits = 0; // nombre d'éléments plus petits que x
   int plusgrands = 0; // nombre d'éléments plus grands que x
   boolean trouve = false; // x a été trouvé dans le tableau
   for(int i = 0; i < tab.length; i++)
      if (tab[i] == x) trouve = true;  // x est dans le tableau
      else if (tab[i] > x) plusgrands++; // un élément > x
      else pluspetits++; // un élément < x
   return trouve && (plusgrands == pluspetits);
}
```

Exercice 6.

La fonction `begaie` :

```
static char[ ] begaie(char[ ] tab)
{
   char[ ] t = new char[2*tab.length];
   for(int i = 0; i < tab.length; i++)
   {
      t[2*i] = tab[i];
      t[2*i+1] = tab[i];
   }
   return t;
}
```