Corrigé du contrôle continu n° 2

Informatique Fondamentale (IF121) — groupe A13

Contrôle du 17 décembre 2003

Barème : 5 points pour l'exercice 1a, 3 points pour 1b et 1c, 5 points pour l'exercice 2. 8 points pour les bases de la programmation (note « prog. ») ; 8/8 signifie que les bases sont acquises (structure générale, méthodes, variables, boucles, parité d'un entier, entrées-sorties). Le total sur 21 est compté sur 20.

Exercice 1

La justification du programme est donnée sous forme de commentaires dans le programme. La démarche générale est de traduire sous forme mathématique les raisons qui ont poussé à écrire le programme de cette façon. Un invariant relie en général les valeurs des variables que le programme modifie avec les données du problème initial.

```
import fr.jussieu.script.Deug;
class ExponentiationRapide {
    static double exponentiationRapide (double x, int n) {
        double y = x;
        double r = 1;
        int k = n;
        while (k > 0) {
            /* Invariant : y = x^(2^i) ; r = x^(n % 2^i) ; k = n / 2^i
               où i est le nombre de fois que la boucle a été exécutée */
            if (k % 2 == 1) {
                r = r * y;
            }
            // Si k est impair : n % 2^(i+1) = n % 2^i + 2^i, et
            //     r = x^(n % 2^i) * x^(2^i) = x^(n % 2^i + 2^i) = x^(2^(i+1))
            // Si k est pair : n % 2^(i+1) = n % 2^i, et r = x^(n % 2^i) = x^(n % 2^(i+1))
            y = y * y;
            k = k / 2;
            // y = (x^(2^i))^2 = x^(2^(i+1))
            // k = (n / 2^i) / 2 = n / 2^(i+1)
            // Terminaison : parce que k est entier, diminue strictement, et reste positif
        }
        // Sortie de la boucle ==> k = 0
        // On a n / 2^i = 0, donc n % 2^i = n, d'où r = x^n.
        return r;
    }

    static void main (String[] args) {    // (la méthode main n'était pas demandée)
        Deug.println("Élévation d'un nombre x à une puissance n.");
        Deug.print("x = ");  double x = Deug.readDouble();
        Deug.print("n = ");  int n = Deug.readInt();
        double r = exponentiationRapide(x, n);
        Deug.println("x^n = " + r);
    }
}
```

Exercice 2

```
import fr.jussieu.script.Deug;
class Trajectoire {
    static void main (String[] args) {
        double g = 9.81, a1 = 1, b1 = 0.2, a2 = 5, b2 = 2;
        Deug.print("Entrez la masse m : "); double m = Deug.readDouble();
        Deug.print("Entrez la durée avant ouverture D : "); double D = Deug.readDouble();
        Deug.print("Entrez l'altitude de largage : "); double z = Deug.readDouble();
        double v = 0, dt = 0.001, t = 0;
        // On suit l'évolution de v et z au cours du temps.
        while (z > 0) {    // On continue tant que le parachutiste n'a pas atteint le
                           // sol, c'est-à-dire tant que son altitude z est positive
            double a, b;
            if (t > D) {    // a et b changent à l'ouverture du parachute
                a = a1; b = b1;
            } else {
                a = a2; b = b2;
            }
            z -= v * dt;
            v += (g - (a + b / m * v) * v) * dt;
            t += dt;
        }
        Deug.println("Durée de chute : " + t + " s");
        Deug.println("Vitesse d'impact : " + v + " m/s");
    }
}
```

Remarques générales

- Lire l'énoncé.
- Répartition des tâches (déjà dit en TP) : le premier exercice demande d'écrire une méthode qui effectue un calcul (calcul de x^n à partir de x et de n). Cette méthode n'a donc pas à lire les données du problème (elles sont passées en argument) ni à afficher le résultat (il doit être retourné par la méthode). C'est la méthode `main` (non demandée) qui se charge de la lecture des entrées et de l'affichage du résultat.
- Test de parité (déjà vu en TP). Soit n un entier. Par définition, n est pair si et seulement si il est multiple de 2, c'est-à-dire qu'il existe un entier k tel que $n = 2 \times k$. On ne peut pas exprimer cette propriété directement en Java. Une caractérisation équivalente est que n est pair si et seulement si le reste de la division de n par 2 est nul. Ceci s'écrit en Java : `n % 2 == 0`. Donc pour tester si le contenu de la variable `n` est pair en Java :
  ```
  if (n % 2 == 0) { /* code pour n pair */ } else { /* code pour n impair */ }
  ```
- Dans une conditionnelle `if (condition) ... else ...` , la condition est forcément fausse quand on exécute la branche « else » (« else » signifie « sinon »). Donc plutôt que d'écrire
  ```
  if (condition) { ... } else if (condition contraire) { ... }
  ```
  on peut écrire directement
  ```
  if (condition) { ... } else { ... }
  ```
- Élévation au carré : x^2 s'écrit en Java `x*x`. (Si `x` est un entier, `x ^ 2` est défini en Java, mais vaut le « ou » exclusif bit à bit de `x` et 2).
- Lire l'énoncé.