NFA031 – TP4

Exercices sur les boucles (1)

24 octobre 2017

Consignes

Pour tous les programmes de ce TP vous devez :

— *Documenter le programme* : ajouter au moins un commentaire dans son entête qui décrit son comportément en termes des entrées et sorties (il faut répondre à la question *quoi* et non pas à la question *comment*). Ajouter également votre nom, vous êtes l'auteur. Inspirez vous des programmes fournis dans l'exercice 1 pour rédiger cette documentation.

— *Produisez un jeu de tests* : suffisamment répresentatif de tous les cas des entrées et sorties du programme.

— Utilisez votre jeu de tests pour testert votre programme. N'oubliez pas de refaire **tous** vos tests après toute modification : ce que marchait avant, peut ne plus marcher après vos changements.

Exercice 1 : Tester, modifier des boucles `while`, réecrire en boucles `for`

Récupérez les 2 programmes fournis `Multiples10.java` et `Puissances2N.java`. Un commentaire dans leur entête explique ce qu'ils font : ces commentaires sont leur *documentation*.

1. Commencez par identifier les éléments importants des boucles dans ces deux programmes. Pour rappel, vous devez identifier les initialisations avant chaque boucle, la condition qui est testée par la boucle, et dans le corps, quelles modifications sont effectuées sur les variables testées par cette condition. Répondez ensuite aux questons suivantes : chaque boucle fait-elle au moins un tour ? Chaque boucle termine ? Que calcule-t-elle ?
2. Compilez et testez le programme `Multiples10`.
3. Modifiez le programme `Multiples10` afin d'afficher tous les nombres sur une même ligne et separés par une une virgule. Il ne doit pas y avoir de virgule sans deux nombres qui l'entourent.
4. Produisez une version équivalente de `Multiples10` mais écrite cette fois avec une boucle `for`. Testez cette nouvelle version.
5. Produisez une table d'exemples pour `PuissancesDe2` ? Compilez et testez les à l'aide de vos jeux d'exemples.
6. Modifiez le programme `PuissancesDe2` afin de calculer et afficher les N premières puissances de 2 pour un nombre N (supposé positif) et lu au clavier.

Exercice 2 : Maximum des nombres lus

Dans cet exercice on veut écrire un programme `MaxNombres` qui lit une suite de nombres entiers au clavier, et calcule puis affiche le maximum d'entre eux.

1. Dans sa première version vous lirez en tout 10 nombres, en utilisant une boucle `for`.
2. Dans une deuxième version, le calcul se fera sur une suite de nombres entiers non nuls. La lecture s'arrête dès qu'un 0 est lu. Question à élucider avant de coder : que fera votre programme si le premier nombre lu est 0 ? Si celui-ci est pour vous un cas spécial, votre documentation doit l'expliciter.

Exercice 3 : lecture validée, non terminaison

On souhaite écrire un programme qui lit ses entrées de manière « sécurisée ». Le programme doit lire un entier x dans une boucle et **assurer** qu'en sortie de boucle, x appartient à un intervalle `[inf,sup]` donné. Si x n'est pas dans l'intervalle, le programme affiche un message d'erreur et demande à nouveau l'entrée d'un entier. La boucle s'arrête lorsque l'entier lu est bien dans l'intervalle donné. Le programme finit par afficher la valeur de x, qui a été validée.

Question 1

Le fichier `DansIntervalleValide.java` contient une version de ce programme où l'on suppose que l'intervalle lu est non vide (valide), à savoir `inf <= sup`.

1. Testez ce programme. Vous ne testerez que les cas d'intervalles non vides.
2. Supposez maintenant que la contrainte `inf <= sup` n'est pas respectée. Que se passe-t-il si pendant l'exécution l'utilisateur entre des valeurs de inf et sup telles que `inf>sup` ?. Expliquez le problème et produisez un exemple d'exécution qui l'illustre.

Question 2

Ce programme n'est pas suffisamment robuste : on peut tenter de lire x, alors qu'on ignore si l'intervalle `[inf,sup]` est lui même valide (autrement dit, non vide). Modifiez ce programme de manière à lire les bornes de l'intervalle de manière sécurisé. N'oubliez pas de changer les commentaires de la documentation : votre programme fonctionne correctement même si l'utilisateur entre des valeurs invalides pour l'intervalle.

Exercice 4 : étude/correction de code

Le programme suivant est censé lire une suite de nombres entiers non nuls, puis calculer et afficher leur moyenne au moyen d'un nombre à virgule. La lecture doit s'arrêter dès qu'un 0 est lu. Si aucun nombre non nul est lu, le programme indique qu'il est impossible de calculer une moyenne. Ce programme est **incorrect** : certains de ses résultats ne correspondent pas à ce qui est spécifié par la documentation du programme (celle-ci est donnée sous forme de commentaires à l'entête du fichier).

— Donnez au moins deux cas de test (valeurs des entrées, résultat attendu pour ces entrées, résultat éronné obtenu) qui montrent que le programme ne se comporte pas comment attendu.
— Identifiez l'erreur de programmation ou de logique dans le code.
— Proposez une correction et modifiez ce code pour qu'il se comporte correctement.

```java
/**
 * Ce programme lit une suite d'entiers non nuls et calcule leur moyenne.
 * Le programme signale l'impossibilité de calcul si aucune valeur
 *  non nulle n'est entrée.
 */

public class MoyenneNombresEntiersErrone{
  public static void main(String[] args){
    int n;
    double somme =0;
    int nbentrees = 0;
    do {
       Terminal.ecrireString("Entrez un entier (fin avec 0): ");
       n = Terminal.lireInt();
       somme = somme + n;
       nbentrees++;
    } while (n!=0);
    if (nbentrees==0){
      Terminal.ecrireStringln("Pas de moyenne car aucune valeur entrée.");
    }else {
      Terminal.ecrireStringln("La moyenne est: "+ (somme/nbentrees));
    }
  }
}
```