

NFA031 – TP3

Exercices sur les boucles

4 novembre 2015

Consignes

Ce TP utilise trois programmes que vous pouvez récupérer en cliquant sur leur nom dans la liste suivante :

- DansIntervalleValide.java
- Mutiples10.java
- PuissancesDe2.java

Pour tous les programmes de ce TP vous devez :

- *Documenter le programme* : ajouter au moins un commentaire dans son entête qui décrit son comportement 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 représentatif 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**, réécrire en boucles **for**

Récupérez les 2 programmes fournis `Multiplies10.java` et `Puissances2N.java`. Un commentaire dans leur entête explique ce qu’ils font : ces commentaires sont leur *documentation*.

1. Produisez une table d’exemples pour chaque programme. Quelle est la particularité du jeu d’exemples pour `Multiplies10` ? Compilez et testez les à l’aide de vos jeux d’exemples.
2. Modifiez le programme `Multiplies10` afin d’afficher tous les nombres sur une même ligne et séparés par une virgule. Il ne doit pas y avoir de virgule sans deux nombres qui l’entourent.
3. Produisez une version équivalente de `Multiplies10` mais écrite cette fois avec une boucle `for`.
4. Même question pour le programme `Puissances2N`.

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'expliquer.

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 $inf \leq sup$.

1. Testez ce programme. Vous ne testerez que les cas d'intervalles non vides.
2. Supposez maintenant que la contrainte $inf \leq sup$ 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 sécurisée. 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.