TP 2

L. Naert – T. Ferragut – E. Lemonnier

5 juillet 2024

Objectifs du TP

- Comprendre la boucle
- Construire une boucle simple avec une seule condition de sortie et sans imbrication

Exercice 1 (reprise du TD 2) (*)

1. Écrire un programme qui demande à l'utilisateur de saisir une suite de nombres et les affiche. La saisie se termine par l'entrée de -1 ;

   Rendre le code du programme avec une version `while` et une version `do-while` et le résultat de leur exécution.
2. Modifier le programme (version `while`) pour qu'il affiche la moyenne des nombres saisis. Le -1 n'est pas comptabilisé.

   Rendre le code du programme et des résultats de son exécution.

Exercice 2 (*)

1. Tester le programme suivant :

```
/**
* Rôle à deviner
* @author M.Adam
*/
class ADeviner {

    void principal() {
        int val1;
        int val2;

        val1 = SimpleInput.getInt ("Première valeur : ");
        val2 = SimpleInput.getInt ("Deuxième valeur : ");

        while (val1 != val2) {
            if (val1 > val2) {
                val1 = val1 - val2;
            } else {
                val2 = val2 - val1;
            }
        }
        System.out.println("Le résultat est : " + val1);
    }
}
```

   Rendre :

   - des exécutions qui se terminent

   - des exécutions qui ne se terminent pas

   - le rôle du programme.

2. Modifier le programme pour que les nombres saisis soient strictement positifs. La saisie se poursuit tant que la valeur n'est pas correcte. Faire une version avec `while` et une version `do-while`.

   Rendre le code corrigé du programme et le résultat de son exécution.

Exercice 3 (**)

Écrire un programme qui demande la saisie de nombres et qui s'arrête si le nombre saisi est inférieur au précédent. Par exemple, si le nombre saisi précédemment est 20, la saisie s'arrête, entre autre, avec 10.

Rendre le code du programme et les tests d'exécution.

Exercice 4 (**)

Écrire un programme simulant un distributeur automatique fonctionnant de la façon suivante :

1. L'utilisateur donne le prix de ce qu'il veut acheter en euros.
2. L'utilisateur donne le montant entré dans la machine.
3. Le programme réagit :
   (a) Si le montant entré est inférieur au prix, le programme redemande un montant à l'utilisateur. Ce montant sera additionné au montant précédent. Quand le montant est supérieur ou égal au prix, il affiche "Produit acheté".
   (b) Si le montant entré est égal au prix, il affiche en plus "Pas de rendu de monnaie".
   (c) Si le montant entré est supérieur strictement au prix, le programme rend la monnaie en détaillant les pièces/billets rendus. Attention, le programme doit rendre le nombre minimal possible de pièces et de billets et ne peut piocher que dans les types de pièces/billets suivants : 1, 2, 5, 10, 20 et 50.

Note : ce programme sera basé sur un nombre entier d'euros. On ne considère pas les centimes.

Rendre le code du programme et les tests d'exécution.

Exercice 5 (**)

L'expression `(int) (Math.random() * 100)` renvoie un entier tiré aléatoirement entre 0 (inclus) et 100 (exclus).

Écrire un programme qui "choisi" un entier entre 0 et 100 et demande à l'utilisateur de deviner le nombre. Le programme ne s'arrête que quand l'utilisateur a trouvé ce nombre. Le programe indique "`Trop grand !`" ou "`Trop petit !`" en fonction de la valeur saisie par l'utilisateur. A la fin, le programme indique le nombre d'essais de l'utilisateur.

Rendre le code du programme et les tests d'exécution.

Exercice 6 (***)

L'objectif de cet exercice est de coder et de tester un programme qui devine un chiffre entre 0 et 1000 choisi par l'utilisateur. Le principe est le suivant :

- l'utilisateur pense à un nombre `x` entre 0 et 1000.
- le programme fait une proposition.
- l'utilisateur répond + si le nombre proposé est trop petit (plus petit que `x`), - si le nombre proposé est trop grand (plus grand que `x`) et = si le nombre proposé est correct (égal à `x`)
- le programme propose des nombres jusqu'à avoir trouvé le nombre choisi.

1. Écrire le programme qui devine le nombre choisi.

   Rendre le code du programme et les tests d'exécution.
2. (Facultatif) Modifier le programme précédent pour qu'il annonce quand le nombre est forcément deviné. Par exemple, pour deviner le nombre 701, si le programme propose 700 et 702, il doit en déduire la bonne réponse.

   Rendre le code du programme et les tests d'exécution.