TP 2 : Les boucles

I. La boucle Pour

La syntaxe *Java* de la boucle `Pour` est la suivante :

```
for( initialisation ; condition_continuité ; incrémentation ) {
  /* instructions */
}
```

1. Écrivez un programme qui affiche 10 fois `Bonjour` à l'écran avec une boucle `for`.
2. Modifiez votre programme pour afficher `Bonjour 1`, `Bonjour 2`, ..., `Bonjour 10`.
3. Sans modifier l'affichage (le contenu du `System.out.println` ou `System.out.print`), affichez maintenant `Bonjour 0`, `Bonjour 3`, ..., `Bonjour 30`.

L'idée est de modifier uniquement l'initialisation, la condition de continuité et l'incrémentation de la boucle.

4. Même chose, mais cette fois-ci, affichez `Bonjour 10`, `Bonjour 9`, ..., `Bonjour 1`.

Vous maîtrisez maintenant la syntaxe et l'utilisation de la boucle `Pour` en *Java*.

II. La boucle Tant Que

La syntaxe *Java* de la boucle `Tant Que` est la suivante :

```
while( condition ) {
  /* instructions */
}
```

1. Écrivez un programme qui affiche 10 fois `Bonjour` à l'écran avec une boucle `while`.
2. Modifiez votre programme pour afficher `Bonjour 1`, `Bonjour 2`, ..., `Bonjour 10`.
3. Sans modifier l'affichage (le contenu du `System.out.println` ou `System.out.print`), affichez maintenant `Bonjour 0`, `Bonjour 3`, ..., `Bonjour 30`.
4. Même chose, mais cette fois-ci, affichez `Bonjour 10`, `Bonjour 9`, ..., `Bonjour 1`.

Vous maîtrisez maintenant la syntaxe et l'utilisation de la boucle `Tant Que` en *Java*.

III. La factorielle

Écrivez un programme qui demande à l'utilisateur de saisir un nombre et qui calcule et affiche la factorielle de ce nombre. Pour rappel, $n!$ = 1 x 2 x … x n-1 x n. Par exemple, 5! = 1 x 2 x 3 x 4 x 5 = 120.

IV. Les multiples

1. Écrivez un programme qui affiche les multiples d'un nombre, noté n, dans un intervalle donné $[a, b]$. Le programme doit tout d'abord demander à l'utilisateur de saisir les bornes a et b de l'intervalle, puis le nombre n. Ensuite, il doit afficher tous les multiples de n dans cet intervalle.
2. Que se passe-t-il si la valeur saisie pour b est inférieure à a ? Si ce n'est pas déjà fait, modifiez votre programme pour prendre en compte ce cas.
3. Si vous avez utilisé une boucle `Tant Que` pour ce programme, modifiez-le pour utiliser plutôt une boucle `Pour`.

V. Le retour de la calculatrice

Cet exercice est basé sur le programme de la calculatrice réalisé lors du TP 1.

1. Si ce n'est pas déjà fait, terminez le programme de la calculatrice du TP 1.
2. On souhaite maintenant donner la possibilité à l'utilisateur de réaliser plusieurs calculs sans avoir à relancer le programme à chaque fois. Modifiez le programme en conséquence.
3. Modifiez maintenant votre programme pour que le nombre de calculs soit limité à 5.

VI. Mastermind

On souhaite réaliser un jeu de *MasterMind* dans lequel l'ordinateur choisit aléatoirement 4 chiffres et l'utilisateur doit les deviner, dans le même ordre. À chaque tour, l'utilisateur fait une proposition de 4 chiffres et l'ordinateur lui indique combien de chiffres sont bien placés et combien sont mal placés.

Pour générer un chiffre aléatoire entre 0 et 9, on peut utiliser le code suivant :

```
int i = (int)(Math.random()*10);
```

1. Écrivez le programme en supposant que l'utilisateur possède un nombre illimité d'essais.
2. Modifiez le programme pour que deux joueurs puissent jouer simultanément contre l'ordinateur (avec deux combinaisons différentes) en limitant le nombre d'essais à 10 pour chaque joueur. Le premier qui trouve la solution est le gagnant. Prévoyez aussi le cas d'égalité, c'est-à-dire lorsque les deux joueurs trouvent la solution au même tour ou qu'ils perdent au bout des 10 essais.