Exercices simples sur les boucles

Exemple préliminaire : exécution d'un programme

Dans cet exemple, nous allons suivre pas à pas l'exécution d'un programme. Les instructions d'un programme peuvent changer la mémoire, c'est à dire les valeurs des variables, et avoir des effets, notamment via les entrées-sorties écran-clavier. Les déclarations pour leur part, ont pour conséquence de créer de nouvelles variables. Nous allons retracer dans un tableau synthétique l'évolution pas à pas d'un programme, en notant l'effet produit par les déclarations et instructions. Ce tableau va comporter une première colonne référant aux numéros de ligne du programme. Une colonne servira à noter la valeur des conditions des instructions conditionnelles et des boucles. Ensuite, il y aura une colonne pour chaque variables et pour finir une colonne pour les entrées clavier et une colonne pour les sorties à l'écran.

Nous noterons qu'une variable n'est pas définie en inscrivant le sigle `NEP` (pour *N'Existe Pas*) dans la colonne correspondante. Un point d'interrogation est utilisé quand la variable existe (elle a été déclarée), mais elle n'a pas été initialisée.

```
1  class Exemple5_0{
2      public static void main(String[] args){
3          int total = 0;
4          int x;
5          Terminal.ecrireString("Entrez␣le␣multiplicateur:␣");
6          x = Terminal.lireInt();
7          for (int i=1; i<=4; i++){
8              total = total + (i*x);
9          }
10         Terminal.ecrireString("La␣somme␣des␣4␣premiers␣multiples␣est:␣");
11         Terminal.ecrireInt(total);
12         Terminal.sautDeLigne();
13     }
14 }
```

Et voici le tableau qui retrace l'exécution de ce programme :

nb	test	total	x	i	clavier	écran
2		NEP	NEP	NEP		
3		0	NEP	NEP		
4		0	?	NEP		
5		0	?	NEP		Entrez le multiplicateur :
6		0	3	NEP	3	
7.init		0	3	1		
7.test	true	0	3	1		
8		3	3	1		
7.modif		3	3	2		
7.test	true	3	3	2		
8		9	3	2		
7.modif		9	3	3		
7.test	true	9	3	3		
8		18	3	3		
7.modif		18	3	4		
7.test	true	18	3	4		
8		30	3	4		
7.modif		30	3	5		
7.test	false	30	3	5		
9		30	3	NEP		
10		30	3	NEP		La somme des 4 premiers multiples est :
11		30	3	NEP		30
12		30	3	NEP		

La première ligne du tableau montre l'état initial avant exécution. L'instruction `for` a trois morceaux intervenant à des moments différents, c'est pourquoi on adistingué les effets de l'initialisation, du test de la condition `i<=4` (quand la condition vaut `true`, la boucle se poursuit, quand elle vaut `false`, la boucle s'arrête), et de la modification de la variable `i`. Les autres types de boucle (`while` et `do ... while`) ne comportent qu'un partie test.

La ligne 9 est prise en compte en ceci que l'accolade ferme un bloc, ce qui met fin à l'existence de la variable `i` qui est locale à ce bloc. On aurait pu de même prendre en compte la ligne 13 qui met fin aux variables `total` et `x`.

Un tel tableau relate une exécution donnée. Si on change la valeur `x` entrée ligne 6, il faut changer tout le tableau. On voit sur cet exemple que le corps de la boucle, la ligne 8, est exécutée 4 fois au cours du programme.

Et pour finir, voici les affichages seuls pour l'exécution retracée dans le tableau précédent, c.a.d, celle où l'on saisit au clavier la valeur 3 :

```
> java Exemple4_0
Entrez le multiplicateur: 3
La somme des 4 premiers multiples est: 30
```

Exercice 3.1.1 *déroulement d'une boucle for*

Avec un tableau comme celui donné dans l'exemple 4.0, retracez une exécution du programme suivant dans laquelle on entre au clavier la valeur 5.

```
1  class Exo5_1{
2      public static void main(String[] args){
3          int x;
4          Terminal.ecrireString("Un␣entier␣svp:␣");
5          x = Terminal.lireInt();
6          for (int i = 0; i<4; i++){
7              Terminal.ecrireInt(x+i);
8              Terminal.sautDeLigne();
9          }
10         Terminal.ecrireStringln("Fini");
11     }
12 }
```

Exercice 3.1.2 *déroulement d'une boucle while*

```
1  class Exo6_2{
2      public static void main(String[] args){
3          int puis = 1;
4          int x, res;
5          Terminal.ecrireString("Un␣entier␣svp:␣");
6          x = Terminal.lireInt();
7          res = x;
8          while (res < 1000){
9              res = res *x;
10             puis = puis+1;
11         }
12         Terminal.ecrireString("Le␣résultat␣est␣");
13         Terminal.ecrireInt(puis);
14         Terminal.sautDeLigne();
15     }
16 }
```

1. Que calcule ce programme ?
2. Avec un tableau comme celui donné dans l'exemple 4.0, retracez une exécution du programme suivant dans laquelle on entre au clavier la valeur 8.

Exercice 3.1.3 *calculs*

1. Écrivez un programme qui affiche la table de multiplication d'un chiffre. Ce chiffre sera entré par l'utilisateur. Par exemple, si le chiffre est 3, le programme affiche :

```
1 x 3 = 3
2 x 3 = 6
3 x 3 = 9
```

```
4 x 3 = 12
5 x 3 = 15
6 x 3 = 18
7 x 3 = 21
8 x 3 = 24
9 x 3 = 27
```

2. Si vous ne l'avez pas déjà fait, modifiez votre programme pour qu'il vérifie que le nombre entré par l'utilisateur est bien un chiffre (c'est à dire un nombre compris entre 1 et 9).
3. Écrivez un programme qui calcule x^y où x et y sont deux entiers saisis au clavier. Pour cela il faut multiplier x fois 1 par y. Par exemple $2^3 = 1 * 2 * 2 * 2$.
4. Écrivez un programme qui affiche la valeur de la fonction x^2 (la fonction qui à un entier associe son carré) pour les dix premiers entiers positifs.

Exercice 3.1.4 *conversion en dollars*

Le programme suivant est une version du programme Conversion des notes de cours, modifié afin de calculer la conversion en dollars d'une somme en euros saisie au clavier.

```
public class Exo2_1_2 {
    public static void main (String[] args) {
        double euros, dollar, cours;

        Terminal.ecrireStringln("Cours␣du␣dollar␣(valeur␣de␣1␣dollar)?␣");
        cours = Terminal.lireDouble();
        Terminal.ecrireStringln("Somme␣en␣euros?␣");
        euros = Terminal.lireDouble();
        dollar = euros / cours;
        Terminal.ecrireStringln("La␣somme␣en␣francs:␣"+ dollar);
    }
}
```

Modifiez ce programme pour que pour un cours du dollar donné (et qui ne change pas pendant l'exécution du programme), on puisse convertir plusieurs sommes. Prévoyez un moyen d'arrêter l'exécution du programme.

Exercice 3.1.5 *date correcte*

Écrivez un programme qui saisit une date correcte sous la forme de trois entiers (jour, mois et année). Le programme doit tester si la date est correcte, et si ce n'est pas le cas, doit signaler le type d'erreur rencontrée, puis demander une nouvelle saisie. Le programme finit lorsqu'une date correcte est enfin saisie, avec l'affichage de celle-ci. Dans le cas où le mois de la date est février, votre programme devra calculer si l'année est bissextile. De manière générale, il devra calculer le nombre de jours maximal du le mois de la date saisie, de manière à valider le numéro de jour qui a été saisi.