La récursivité - Révisions

Les méthodes qui traitent des entiers ou tableau d'entiers pourront être définies dans une classe nommée, par exemple, ***OutilEntierRecursif*** (exercices 1, 3, 4, 5, 6).

Exercice 1 : Suite de Fibonnacci

La suite de Fibonnacci est définie de la manière suivante :

$U_0 = 1$
$U_1 = 1$
$U_n = U_{n-1} + U_{n-2}$ pour $n \geq 2$

Ecrire une méthode récursive qui calcule la valeur du terme de rang ***n***. Tester cette méthode. Attention, si vous cherchez des valeurs de U_n sur Internet : il existe plusieurs version de la suite, l'une avec un premier terme égal à 0, une autre avec un premier terme égal à 1.

Exercice 2 : Figures géométriques - Tester les méthodes au fur et à mesure

1) Que fait la méthode suivante ? Essayer de répondre sans tester la méthode. Puis vérifiez votre réponse en recopiant le code dans une classe que vous créerez (classe ***FigureGeometrique*** par exemple). Vous aurez besoin de cette méthode dans la suite de l'exercice.

```
        public static void blanc(int n) {
            if (n > 0) {
                System.out.print(' ');
                blanc(n - 1);
            }
        }
```

2) Ecrire une méthode récursive qui aura en argument un entier *n*. Son rôle sera d'afficher d'abord sur une ligne *n* fois le caractère '*', **puis d'amener le curseur au début de la ligne suivante**. L'entier *n* devra être positif ou nul. Sinon la méthode sera sans effet.

3) Ecrire une méthode récursive qui aura en argument un entier *n*. Son rôle sera d'afficher à l'écran un triangle rectangle formé avec le caractère '* ' et de hauteur *n*.

Exemple – Si n est égal à 4, on obtiendra le triangle suivant (il ne faut pas afficher le quadrillage).

*			
*	*		
*	*	*	
*	*	*	*

4) Ecrire une méthode récursive qui aura en argument un entier *n*. Son rôle sera d'afficher à l'écran un triangle isocèle formé avec le caractère '* ' et de hauteur *n.*

Exemple – Si *n* est égal à 5, on obtiendra le triangle suivant (il ne faut pas afficher le quadrillage).

				*				
			*	*	*			
		*	*	*	*	*		
	*	*	*	*	*	*	*	
*	*	*	*	*	*	*	*	*

5) Facultatif : Ecrire une méthode récursive qui aura en argument un entier *n*. Son rôle sera d'afficher à l'écran un losange formé avec le caractère '* '. *n* sera le nombre d'étoiles de la diagonale du losange. Il doit être impair.

Exemple – Si *n* est égal à 7, on obtiendra le losange suivant (il ne faut pas afficher le quadrillage).

			*			
		*	*	*		
	*	*	*	*	*	
*	*	*	*	*	*	*
	*	*	*	*	*	
		*	*	*		
			*			

6) Facultatif : Ecrire une méthode récursive permettant de tracer une spirale. Le nombre d'enroulements sera donné en paramètre. Attention : la hauteur des lignes sur l'interface console est plus grande que la largeur des colonnes. La spirale n'entrera pas dans un carré, mais sera étirée dans le sens de la hauteur.

Exemples d'exécution du programme :

```
Spirale avec 1 enroulement(s) :

  ***
  * *
  * *
    *
*****
```

```
Spirale avec 4 enroulement(s) :

  ***************
  *             *
  * *********** *
  * *         * *
  * * ******* * *
  * * *     * * *
  * * * *** * * *
  * * * * * * * *
  * * * * * * * *
  * * *   * * * *
  * * ***** * * *
  * *       * * *
  * ********* * *
  *           * *
  ************* *
                *
*****************
```

Exercice 3 : Traitements sur les tableaux d'entiers - Tester au fur et à mesure

1) Ecrire une méthode récursive qui initialise toutes les cases d'un tableau d'entiers avec une valeur donnée. Le tableau à initialiser sera passé en paramètre. On suppose qu'il a été créé au préalable. La valeur initiale sera elle aussi passée en paramètre.

Par exemple, si la valeur initiale souhaitée est -1, on obtient le tableau suivant :

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

2) Ecrire une méthode récursive qui détermine si un entier donné est présent ou pas dans un tableau d'entiers.

Par exemple, l'entier 2 est présent dans le tableau ci-dessous, mais pas l'entier 22 :

3	10	5	9	4	2	1	8	14	-6

3) Facultatif : Ecrire une méthode récursive qui détermine l'indice de la case contenant le maximum du tableau. Si le maximum est présent plusieurs fois, c'est l'indice de la première occurrence qui sera renvoyé par la méthode.

Dans le tableau ci-dessous, l'indice à renvoyer est égal à 8.

3	10	5	9	4	2	1	8	14	-6

Dans le tableau ci-dessous, l'indice à renvoyer est égal à 1.

3	10	5	9	4	2	1	8	10	-6

Exercice 4 : Afficher les chiffres d'un entier dans l'ordre inverse

Ecrire une méthode récursive qui affiche à l'écran tous les chiffres d'un nombre entier, dans l'ordre inverse et séparés par un espace. Chaque chiffre sera séparé du suivant pas un espace. Il est inutile de convertir l'entier en chaîne de caractères. Si l'entier est négatif, le signe - ne sera pas affiché.

Par exemple, si le nombre entier est 1736, la méthode devra afficher : 6 3 7 1
Attention, il ne faut pas afficher d'espace en plus, ni avant premier, ni après le dernier chiffre.

Variante facultative : La méthode récursive devra renvoyer le résultat sous la forme d'une chaîne de caractères. Le résultat ne sera pas affiché par la méthode.

Exercice 5 : Programme mystère

Que fait la méthode *traitement* de la classe suivante ? Si vous ne trouvez pas la réponse, écrivez le programme et testez-le. Tentez ensuite de comprendre son fonctionnement.

```
package recursivite;
import java.util.Scanner;
/**
 * Programme mystère
 * @author INFO2
 */
public class Mystere {

    /** Saisie sur l'entrée standard  */
    private static Scanner entree = new Scanner(System.in);

    /** Méthode récursive */
    public static void traitement() {
        char caract; // contient le caractère entré par l'utilisateur
        System.out.print(" ? ");

        // on stocke le premier caractère entré par l'utilisateur
        caract = entree.nextLine().charAt(0);
        if (caract != '.') {
            traitement();   // appel récursif
        }
        System.out.print(caract);
    }
```

```
    /**
     * Programme principal
     * @param args  argument non utilisé
     */
    public static void main(String args[]) {
        System.out.println(
                "Lorsque le programme affiche '?', l'utilisateur doit entrer un "
                + "caractere. \nPour terminer le programme, il doit entrer le "
                + "caractere + '.'. \n");
        traitement();
        System.out.println();
    }
}
```

Exercice 6 : La fonction d'Ackermann

La fonction d'Ackermann est définie récursivement de la manière suivante :

$Ack(m, n) = Ack(m-1, Ack(m, n-1))$ pour $m > 0$ et $n > 0$

$Ack(0, n) = n + 1$ pour $n > 0$

$Ack(m, 0) = Ack(m-1, 1)$ pour $m > 0$

Ecrire une méthode récursive qui calcule la valeur de la fonction d'Ackermann, si celle-ci est définie. Dans le cas contraire, la méthode renverra -1.
Exemples : *Ack(0,4) = 5* *Ack(4,0) = 13* *Ack(3,4) = 125*

Exercice 7 : Calcul du PGCD de 2 entiers (2 méthodes différentes)

Ecrire une méthode récursive qui calcule le PGCD (Plus Grand Commun Diviseur) de deux entiers strictement positifs.

Par exemple, le PGCD de 42 et 60 est 6. Celui de 1155 et 9724 est 11. Celui de 960 et 108 est 12.

a) On écrira une première méthode basée sur la méthode des divisions successives. Elle utilise la propriété suivante :

pgcd (a , b) = a si a = b
pgcd (a , b) = pgcd (a - b , b) si a > b
pgcd (a , b) = pgcd (a , b - a) si a < b

b) On écrira une deuxième méthode basée sur la méthode d'Euclide. L'idée est la suivante : on veut calculer le pgcd entre a et b, et on sait que a est plus grand que b. On calcule le reste *r* de la division a par b. Le pgcd entre a et b est égal au pgcd entre *r* et b. On recommence jusqu'à obtenir un reste nul. Le pgcd est alors égal au dernier nombre utilisé comme diviseur.

Par exemple, si on veut calculer pgcd(960, 108), on calculera d'abord le reste de la division de 960 par 108. On obtient 96. Il faut maintenant calculer pgcd(108, 96). Le reste de la division entre 108 et 96 est égal à 12. On doit ensuite calculer pgcd(96, 12). On divise 96 par 12. Le reste de la division est égal à 0. Le pgcd recherché est donc égal à 12.

c) Facultatif : Comparer l'efficacité des 2 méthodes. Il faut constituer un grand ensemble de données (éventuellement des valeurs générées aléatoirement, et on calcule le pgcd entre chaque d'entre elles et toutes les autres valeurs) et compter, pour chaque méthode, le nombre d'appels récursifs.