TD 3 – Récursivité

Structures de données (IF 122)

1 Récursivité simple

Une fonction (ou une procédure) récursive est une fonction qui s'appelle elle-même. Ainsi, la fonction factorielle peut être définie de manière plus concise à l'aide d'une fonction récursive :

```
import fr.jussieu.script.Deug;

class Fact{

    static int fact_imperative(int n){
        int res = 1;
        for (int i=1;i<=n;i++) res=res*i;
        return res;
    }

    static int fact_recursive(int n){
        if (n==0) return 1;
        else return n*fact_recursive(n-1);
    }

    public static void main (String[] args) {
        int p;
        p = Deug.readInt();
        Deug.println(fact_imperative(p) + "   " + fact_recursive(p));
    }
}
```

Il faut faire attention à ce que la fonction ne boucle pas sur la même valeur, auquel cas le programme ne s'arrêterait jamais, comme dans cet exemple :

```
import fr.jussieu.script.Deug;

class Stupide{
    static int boucle(int n){
        return boucle(n);
    }

    public static void main (String[] args) {
        int p;
        p = boucle(0);
        Deug.println("ce message n'arrivera jamais");
    }
}
```

Pour éviter les appels infinis :

- l'appel récursif doit toujours être fait avec un paramètre de valeur différente que l'appel initial, dans la plupart des cas ce paramètre est plus petit
- il doit toujours y avoir un cas d'arrêt, i.e. sans appel récursif.

Exercice 1 (TD) Que fait le programme suivant sur des entrées positives ? Sur des entrées négatives ? Corrigez-le pour qu'il ait un comportement cohérent pour toutes les entrées. Dessiner l'arbre des appels pour ce programme si on rentre la valeur 4. Comment progresse la pile des appels récursifs ?

```
import fr.jussieu.script.Deug;

class Compteur{

    static void f(int n){
        Deug.print(n + " ");
        if (n!=0) f (n-1);
        Deug.print(n + " ");
    }

    public static void main (String[] args) {
        int p = Deug.readInt();
        f(p);
    }
}
```

Exercice 2 (TD et TP)

1. Écrire une fonction `String repete( int n, String s)` renvoyant la chaîne de caractères s répétée n fois : `repete(3, "bla")` donne `"blablabla"`.
2. Écrire une fonction `void pyramide( int n, String s)` qui écrit sur la première ligne, 1 fois la chaîne `s`, sur la deuxième, 2 fois la chaîne `s`, et ainsi de suite jusque la dernière ligne, où il y aura `n` fois la chaîne `s`. Ainsi `pyramide(5, "bla");` donnera

   ```
   bla
   blabla
   blablabla
   blablablabla
   blablablablabla
   ```
3. Quand on lance `pyamide(n, s)`, combien y a-t-il d'appels à `pyramide`, combien y a-t-il d'appels à `repete` ? Pour vous aider à répondre dessiner l'arbre des appels pour `n` $= 3$.

Exercice 3 (TD) Écrire une fonction qui calcule la puissance n^p, n et p étant deux entiers.

Exercice 4 (TP) Écrire une fonction qui calcule la valeur F_n de la suite de Fibonacci en n :

$$F_0 = 0 \qquad F_1 = 1$$
$$F_{n+2} = F_n + F_{n+1} \qquad \text{pour } n \geqslant 0$$

Exercice 5 (TD et TP) On se donne la fonction d'Ackerman :

$$Ack(0, p) = p + 1$$
$$Ack(n, 0) = Ack(n - 1, 1)$$
$$Ack(n, p) = Ack(n - 1, Ack(n, p - 1))$$

Cette fonction utilise-t-elle des appels infinis ? Écrivez un programme qui calcule $Ack(n, p)$, demandez-lui la valeur de $Ack(4, 4)$... Que se passe-t-il ?

2 Récursivité croisée

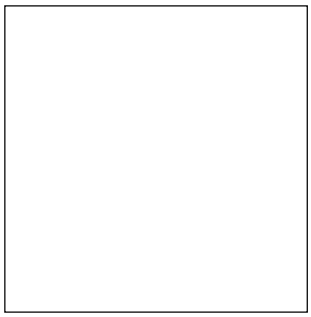
On parle de récursivité *croisée* lorsque deux fonctions s'appellent l'une l'autre récursivement.

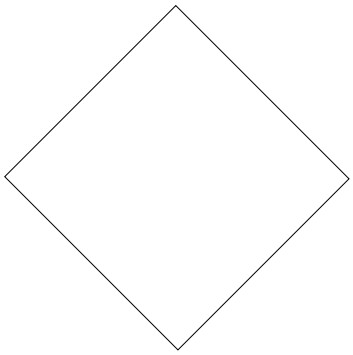
Exercice 6 (TD) Les fonctions suivantes sont censées donner la parité d'un nombre entier : cela sera-t-il le cas pour toutes les valeurs entières positives ?

```
import fr.jussieu.script.Deug;

class PairImpair{

    static boolean pair (int n){
        if (n==0)
            return true;
        else
            return impair(n-1);
    }
    static boolean impair (int n){
        if (n==1)
            return true;
        else
            return pair(n-1);
    }

    public static void main (String[] args) {
        int p = Deug.readInt();
        Deug.println("pair? "+ pair(p) + " impair? " + impair(p));
    }
}
```

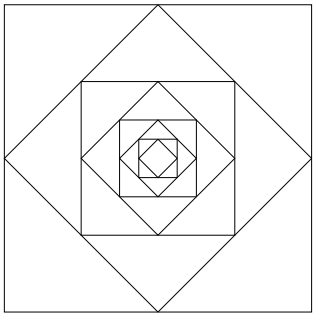
Morale de l'histoire : "Dans les recursions croisées, il vaut mieux que le cas d'arrêt soit le même pour toutes les fonctions".

Exercice 7 (TD et TP) 1. Écrire deux méthodes `CarreDroit` et `CarrePenche` qui permettent, dans un environnement graphique, de construire les deux carrés suivants (on utilisera la fonction `Deug.drawLine(x1,y1,x2,y2)`) :





2. Comment dessiner la figure suivante avec deux fonctions mutuellement récursives ? Et avec une seule fonction ?



▶ **Exercice 2**

```
import fr.jussieu.script.Deug;

class Pyramide{

    static String repete( int n, String s){
        if (n == 0) return "";
        else return (s + repete(n-1, s));
    }

    static void pyramide (int n, String s){
        if (n==0) return;
        else {
            pyramide(n-1,s);
            Deug.println(repete(n,s));
        }
    }

    public static void main (String[] args) {
        pyramide(5,"bla");
    }
}
```

▶ **Exercice 3**

```
import fr.jussieu.script.Deug;

class Puissance{

    static int puissance (int n, int p) {
        if (p == 0) return 1;
        else
            return ( n* puissance(n,p-1));
    }

    public static void main (String[] args) {
        int n = Deug.readInt();
        int p = Deug.readInt();
        Deug.println(puissance(n,p));
    }
}
```

▶ **Exercice 4**

```
import fr.jussieu.script.Deug;

class Fib{

    static int fib (int n) {
        if (n == 0)
            return 0;
        else {
        if (n==1) return 1;
        else return (fib(n-2)+fib(n-1));
```

```
    }
    }

    public static void main (String[] args) {
        int p = Deug.readInt();
        Deug.println(fib(p));
    }
}
```

▶ **Exercice 5**

```
import fr.jussieu.script.Deug;

class Ack{

    static int ack (int n, int p) {
        if (n == 0)
            return p+1;
        else {
        if (p==0) return ack(n-1,1);
        else return (ack(n-1,ack(n,p-1)));
    }
    }

    public static void main (String[] args) {
        int n = Deug.readInt();
        int p = Deug.readInt();
        Deug.println(ack(n,p));
    }
}
```

▶ **Exercice 7**

```
import fr.jussieu.script.Deug;

class CarreRecursif {
    /* x, y: centre, n: demi cote*/
    static void CarreDroit(int n, int x, int y) {
    if (n==1)
        return;
    Deug.drawLine(x+n,y+n, x-n, y+n);
    Deug.drawLine(x+n,y+n, x+n, y-n);
    Deug.drawLine(x+n,y-n, x-n, y-n);
    Deug.drawLine(x-n,y+n, x-n, y-n);
    CarrePenche (n, x, y);
    }

    /* x, y: centre, n: demi diagonale*/
    static void CarrePenche(int n, int x, int y) {
    if (n==1)
        return;
    Deug.drawLine(x+n,y, x, y+n);
    Deug.drawLine(x+n,y, x, y-n);
    Deug.drawLine(x-n,y, x, y-n);
```

```
    Deug.drawLine(x-n,y, x, y+n);
    CarreDroit (n/2, x, y);
    }
    static int puissance(int n, int p) {
    if (p == 0) return 1;
    return n * puissance (n, p-1);
    }

    public static void main (String[] args) {
    Deug.println("donner l'ordre de la courbe");
    int p =Deug.readInt();
    int n= puissance(2,p);
        Deug.startDrawings(2*n,2*n);

    CarreDroit(n,n,n);
    Deug.println("c'est fini appuyer sur une <Enter> pour fermer la fenetre");
    Deug.readChar();
    Deug.stopDrawings();

    }
}
```