

Les fonctions

Une **fonction** est un bout de code qui a un **nom** et qui peut être appelée depuis le reste du programme.

Exemples de fonctions prédéfinies: Math.pow(a,b),
Math.abs(a), Math.random(), Math.sqrt(a), ...

Intérêts des fonctions

- Meilleure lisibilité et concision (évite d'écrire plusieurs fois le même code)
- Niveau d'abstraction plus élevé (un nom remplace une partie complexe du programme)
- Gain de place mémoire
- Modification plus aisée
- Partage de fonctions (bibliothèque de fonction)
- Récursivité

Déclaration d'une fonction

On déclare une fonction à l'extérieur du **main** mais à l'intérieur de **class**. Je conseille de les déclarer juste après le programme principal.

```
public class programme
{
    public static void main(String args[])
    {
        // Programme principal
    }

    // Déclaration des fonctions

}
```

Deux types de fonction

- Fonction qui renvoie une valeur (Math.sqrt(),
Math.random(), périmètre(), moyenne(),...)
- Fonction qui ne renvoie pas une valeur (affichage d'un tableau, ...)

Déclaration d'une fonction qui renvoie une valeur

```
public static type Nomfonction(paramètres formels)
{ //instructions de la fonction
    return valeurrenvoyée; }
```

- * **type** est le type de la valeur renvoyée
- * **Nomfonction** est le nom de la fonction
- * **Paramètres formels** sont les données nécessaires pour la déclaration de la fonction (ils restent donc sans valeur)

Appel de la fonction

```
variable=Nomfonction(paramètres effectifs);
//renvoie la valeur de la fonction
```

- * **Les paramètres effectifs** sont les paramètres sur lesquels la fonction est exécutée
- * **Lors de l'exécution les paramètres formels sont remplacés par les paramètres effectifs**

Les fonctions

Fonction qui renvoie le maximum de deux entiers

```
public static int maximum(int a, int b)
    {int maxi; // Variable locale
     if(a>b)
        maxi=a;
     else
        maxi=b;
     return maxi; // Renvoie le maximum de a et b
    }
```

Les variables locales sont connues seulement dans la fonction

Appel de la fonction qui renvoie le maximum de deux entiers

```
int x=1,y=2,max;
max=maximum(x,y); // Appel de la fonction sur x et y
System.out.println(max); // affichage de 2
```



Les fonctions

Fonction qui renvoie la moyenne de 2 réels

```
public static double moyenne(double a, double b)
{double moy; // Variable locale
moy=(a+b)/2;
return moy; // Renvoie la moyenne de a et b
}
```

Appel de la fonction qui renvoie la moyenne

```
double x=1.5,y=2.5,moy;
moy=moyenne(x,y); // Appel de la fonction sur x et y
System.out.println(moy); // affichage la moyenne
```

Il y a ici deux variables moy !!!!!!!!

Les fonctions

Déclaration d'une fonction qui ne renvoie pas de valeur

```
public static void Nomfonction(paramètres formels)
{
    //instructions de la fonction
    //il n'y a pas le return
}
```

Appel de la fonction

```
Nomfonction(parametres effectifs);
// C'est une instruction qui ne renvoie pas de valeur
// Elle se contente d'exécuter les instructions de la fonction
```

Les fonctions

Fonction qui affiche le maximum de deux entiers (sans le renvoyer)

```
public static void maximum(int a, int b)
{int maxi; // Variable locale
 if(a>b)
    maxi=a;
 else
    maxi=b;
 System.out.println("Le max de "+a+" et "+b+" est : "+maxi);
// Affiche le maximum de a et b
}
```

Appel de la fonction qui affiche le maximum de deux entiers

```
int x,y;
x=1;
y=2;
maximum(x,y);      // Appel de la fonction sur x et y
```



Les fonctions

Peut-on modifier un paramètre de fonction?

-**Les paramètres de types int, char, double, boolean, String ne peuvent pas être modifiés au cours de l'exécution d'une fonction.**

On dit que les paramètres sont **passés par valeur**.

-**Les paramètres de types tableaux 1D ou 2D peuvent être modifiés.**

On dit que les paramètres sont **passés par références (ou variable)**.

Les fonctions

Tentative de modification d'un paramètre passé par valeur

```
public static int ajoutun(int a)
{ a=a+1;
  return a;
}
```

Appel de la fonction

```
int x;
x=1;
y=ajoutun(x);      // Appel de la fonction sur x

// A la fin, y=2 et x=1
```

Les fonctions

Tentative de modification d'un paramètre passé par valeur

```
public static int ajoutun(int a)
{ a=a+1;
  return a;
}
```

Appel de la fonction

```
int x;
x=1;
x=ajoutun(x);      // Appel de la fonction sur x

// A la fin, x=2 Ici la modification est réalisée
// (mais c'est en trichant!!!)
```

Les fonctions

Modification d'un paramètre passé par référence

```
public static void modif(int tab[])
{ tab[0]=1;
  tab[1]=2;
  tab[2]=3;
}
```

Appel de la fonction

```
int T[];
T=new int[3];
T[0]=0;T[1]=1;T[2]=2;
modif(T);      // Appel de la fonction sur T

// A la fin, T[0]=1 et T[1]=2 et T[2]=3
```

Les fonctions - La récursivité

Une fonction est **récursive** si elle s'appelle elle même.

Il y a toujours un cas de base sans appel récursif, et le cas général où intervient la récursivité.

Fonction récursive pour calculer $n! = (n-1)! * n$

```
public static int facto(int n)
{
    if (n==0)
        return 1;      // Cas de base
    else
        return n*facto(n-1); // Cas général de la récursivité
}
```

Appel de facto

```
int n,resu;
n=10;
resu=facto(n);      // Appel de la fonction récursive facto
```



Les fonctions - La récursivité

Partant d'un couple de lapins, combien de couples obtiendrons-nous après un nombre donné de mois sachant que chaque couple produit chaque mois un nouveau couple, lequel ne devient productif qu'après deux mois.

Fonction récursive pour calculer les nombres de Fibonacci

$$f_0 = 1, f_1 = 1, f_n = f_{n-1} + f_{n-2}$$

```
public static int fibo(int n)
{
    if(n==0 || n==1)
        return 1;          // Cas de base
    else
        return fibo(n-1)+fibo(n-2); // Cas général
}
```

Appel de fibo

```
int n,resu;
n=10;
resu=fibo(n);      // Appel de la fonction récursive fibo
```

Les fonctions - La récursivité

Fonction récursive pour calculer les coefficients binomiaux

$$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$$

```
public static int coeff(int n, int k)
{
    if (k==0 || k==n)
        return 1;          // Cas de base
    else
        return coeff(n-1, k-1)+coeff(n-1, k); // Cas général
}
```

Appel de coeff

```
int n, k, resu;
n=10; k=3;
resu=fibo(n, k);      // Appel de la fonction récursive coeff
```