

Du Processing au Java

Valeur d'accueil et de reconversion en informatique (VARI1)

- 1 Les premiers programmes : mots clés et compilation
- 2 Fonctions avancées, lecture clavier

Le 1er programme : Processing VS Java

Sous Processing :

```
println("Salut_les_amis!");
```

Sous Java :

```
class PremierProg{  
    public static void main(String[] args){  
        System.out.println("Salut_les_amis!");  
    }  
}
```

Le 1er programme : Processing VS Java

Sous Processing :

```
println("Salut_les_amis!");
```

Sous Java :

```
class PremierProg {  
    public static void main(String [] args) {  
        System.out.println("Salut_les_amis!");  
    }  
}
```

Il faut toujours déclarer **class X**, où **X** est le nom du fichier sans extension

Les méthodes sont déclarées comme sous Processing mais il faut ajouter `public static` au début.


Le 1er programme : Processing VS Java

Sous Processing :

```
println("Salut_les_amis!");
```

Sous Java :

```
class PremierProg{  
    public static void main(String[] args){  
        System.out.println("Salut_les_amis!");  
    }  
}
```



Remplacer le `println(...)` de Processing par `System.out.println(...)`, il n'y a pas d'autre choix!

Le 1er programme : Processing VS Java

Sous Processing :

```
println (" Salut_les_amis! ");
```

Sous Java :

```
class PremierProg {  
    public static void main (String [] args) {  
        System.out.println (" Salut_les_amis! ");  
    }  
}
```

Il faut deux étapes pour lancer : compilation et exécution

compilation `javac PremierProg.java`

→ un exécutable bytecode pour la machine virtuelle java

exécution `java PremierProg`

→ lance la machine virtuelle Java

Le 1er programme : Processing VS Java

Sous Processing :

```
println (" Salut_les_amis! ");
```

Sous Java :

```
class PremierProg {  
    public static void main (String [] args) {  
        System.out.println (" Salut_les_amis! ");  
    }  
}
```



Conclusion

Beaucoup de mots clés (langage verbeux),
mais le programme n'est pas si complexe,
pas d'interactions entre les mots clés

Une 2ème fonction : calculer l'IMC = $\frac{\text{kg}}{\text{mètres}^2}$

```
class CalculerImc{
    static float calcImc(float kg, float m){
        return kg/(m*m);
    }
    public static void main(String [] args){
        float poidsKg = 90;
        float tailleM = 1.8; //ou 180 cm
        float imc = calcImc(poidsKg, tailleM);
        System.out.println(imc);
    }
}
```


Une 3ème fonction : la puissance

```
class CalculerPuissance{
    //renvoyer x^n
    static double puissance(double x, int n){
        double p = x;
        for(int i=0;i<n;i++) //est-ce que
            p = p * x;      //c'est bien
        return p;          //correct?:
    }
    public static void main(String [] args){
        double puissance5 = puissance(2,5);
        System.out.println(puissance5);
    }
}
```

! Remarquez qu'on met `static` devant chaque fonction !

- 1 Les premiers programmes : mots clés et compilation
- 2 Fonctions avancées, lecture clavier

Demander à l'utilisateur de saisir x

Il faut utiliser un objet de classe `Scanner`

Ce code permet de lire un entier :

```
java.util.Scanner s ;  
s = new java.util.Scanner(System.in) ;  
int x = s.nextInt() ;
```

On peut lire :

`un double` `s.nextDouble()`

`un mot` `s.next()`

`une ligne` `s.nextLine()`

Plusieurs saisies, un seul objet Scanner

```
class TestScanner{
    public static void main(String [] args){
        java.util.Scanner s ;
        s = new java.util.Scanner(System.in) ;
        int          x = s.nextInt() ;
        double       y = s.nextDouble() ;
        String toto = s.next() ;
    }
}
```

Note : `System.out` et `System.in` font référence à l'entrée et la sortie par rapport au terminal

Valeurs minimales et maximales d'un tableau

- 1 Écrire une fonction `Java` pour déterminer la valeur minimale d'un tableau
- 2 Faire appel à la fonction ci-dessus pour calculer la valeur maximale :
 - inverser tous les nombres
 - calculer min
 - inverser tous les nombres

Valeurs minimales et maximales d'un tableau

- 1 Écrire une fonction `Java` pour déterminer la valeur minimale d'un tableau
- 2 Faire appel à la fonction ci-dessus pour calculer la valeur maximale :
 - inverser tous les nombres
 - calculer min
 - inverser tous les nombres

Valeurs minimales et maximales d'un tableau

- 1 Écrire une fonction `Java` pour déterminer la valeur minimale d'un tableau
- 2 Faire appel à la fonction ci-dessus pour calculer la valeur maximale :
 - inverser tous les nombres
 - calculer min
 - inverser tous les nombres

Les variables globales : déclaration `static`

Et si on voulait calculer la valeur minimale **et** maximale d'un tableau :

- il ne suffit pas un seul `return`

⇒ on met le résultat dans des variables globales `min` et `max`

- il suffit de les déclarer `static` au début du code

```
class TabMinEtMax{
    static int min;
    static int max;
    static void calcMinMax(int [] t){
        ...
        min = ...
        max = ...
    }
}
```


Les variables globales : déclaration `static`

Et si on voulait calculer la valeur minimale et maximale d'un tableau :

- il ne suffit pas un seul `return`

⇒ on met le résultat dans des variables globales `min` et `max`

- il suffit de les déclarer `static` au début du code

```
class TabMinEtMax{
    static int min;
    static int max;
    static void calcMinMax(int [] t){
        ...
        min = ...
        max = ...
    }
}
```

La racine carré

- 1 Utiliser `Math.sqrt(x)`, c. à. d. la fonction statique `sqrt` de la classe `Math`
- 2 Écrire notre propre fonction `racine(double x)`
 - Utiliser la suite convergente suivante :

$$r_{n+1} = \frac{r_n + \frac{x}{r_n}}{2}$$

- 3 Calculer $\sqrt[4]{81}$, implémenter une fonction pour la racine d'ordre 4 !

La racine carré

- 1 Utiliser `Math.sqrt(x)`, c. à. d. la fonction statique `sqrt` de la classe `Math`
- 2 Écrire notre propre fonction `racine(double x)`
 - Utiliser la suite convergente suivante :

$$r_{n+1} = \frac{r_n + \frac{x}{r_n}}{2}$$

- 3 Calculer $\sqrt[4]{81}$, implémenter une fonction pour la racine d'ordre 4 !

La racine carré

- 1 Utiliser `Math.sqrt(x)`, c. à. d. la fonction statique `sqrt` de la classe `Math`
- 2 Écrire notre propre fonction `racine(double x)`
 - Utiliser la suite convergente suivante :

$$r_{n+1} = \frac{r_n + \frac{x}{r_n}}{2}$$

- 3 Calculer $\sqrt[4]{81}$, implémenter une fonction pour la racine d'ordre 4 !