



Java

Retour sur la syntaxe

Objectifs

- Retour sur les syntaxes abordées
- Lecture et écriture dans un fichier



- Affichage sur la sortie standard

```
System.out.println("Hello world");  
System.out.print("Bonjour monde");
```

retourne à la ligne
ne retourne pas à la ligne

- Affichage sur la sortie d'erreur

```
System.err.println("Hello world");  
System.err.print("Bonjour monde");
```

système d'erreur (sort en rouge sur blanc)

- Concaténer des chaînes

```
String prenom = "Etienne";  
System.out.println("Bonjour " + prenom);
```



- Les variables et constantes

```
public class Screenshots {  
  
    // Déclaration d'une constante  
    public static final int ageMajorite = 18;  
  
    public static void main(String[] args) {  
        // Déclaration d'une variable  
        int age = 30;  
        System.out.println("L'age de majorité est : " + ageMajorite + ". Vous avez + " + age + " ans.");  
    }  
}
```

La valeur ne va jamais changer
final (ne changera pas de valeur)
agemajorite=21 => erreur



- Les tableaux

```
// Déclarer un tableau
int[] monTableau = new int[100];
float[] unAutreTableau = new float[] {5.2f, 3.4f};  sans les f, il aurait mis des doubles

// Affecter une valeur
monTableau[0] = 42;

// Accéder à une valeur
System.out.println(unAutreTableau[1]);
```



- Les tableaux

```
// Déclarer un tableau
int[] monTableau = new int[100];
float[] unAutreTableau = new float[] {5.2f, 3.4f};

// Affecter une valeur
monTableau[0] = 42;

// Accéder à une valeur
System.out.println(unAutreTableau[1]);
```

Souvenez-vous :

- les indices d'un tableau commencent à 0
- le plus grand indice est « taille du tableau – 1 »



- Saisie utilisateur

```
// Déclaration d'un Scanner
Scanner sc = new Scanner(System.in);

// Récupération de diverses valeurs
int age = sc.nextInt();
float taille = sc.nextFloat();
String nom = sc.nextLine();

// Ne pas oublier de fermer le Scanner à la fin
sc.close();
```

- N'oubliez pas !

```
import java.util.Scanner;
```



- Les conditions

```
// Les instructions "else if" et "else" sont facultatives
// A vous de voir en fonction de vos besoins !
if (age < 18) {
    System.out.println("Vous êtes mineur");
} else if (age < 50) {
    System.out.println("Allez au boulot !");
} else if (age < 60) {
    System.out.println("Bientôt la retraite !");
} else {
    System.out.println("Toujours vivant ?");
}
```



- Les conditions

```
switch(materiau) {  
  case "carton" : {  
    System.out.println("Poubelle jaune");  
    break; // A ne pas oublier !  
  }  
  case "verre" : {  
    System.out.println("Bac à verre");  
    break; // Toujours à ne pas oublier  
  }  
  default : {  
    System.out.println("Poubelle ordinaire");  
    break; // Ici c'est moins grave, mais on le met quand meme  
  }  
}
```

Le default est forcément en dernier



- Les boucles

```
String[] prenom = new String[] {"Etienne", "François", "Antoine", "Maxime"};
// boucle "for" avec variable itérative
for (int i = 0; i < prenom.length; i++) {
    System.out.println("Bonjour " + prenom[i]);
}
// boucle "foreach", si je n'ai pas besoin de l'indice
for (String current : prenom) {
    System.out.println("Coucou " + current);
}
```

current n'est pas obligatoire: on aurait pu mettre i



- Les boucles

```
int puissanceDeDeux = 2;  
while (puissanceDeDeux < 1000) {  
    System.out.println(puissanceDeDeux);  
    puissanceDeDeux *= 2;  
}
```



- Les boucles

```
Scanner sc = new Scanner(System.in);  
int choix;
```

```
do {  
    System.out.println("Choisissez votre classe !");  
    System.out.println("1. Guerrier");  
    System.out.println("2. Mage");  
    System.out.println("3. Archer");  
    choix = sc.nextInt();  
} while (choix < 1 && choix > 3);  
sc.close();
```

do {} while ; Très utilisé pour les menus utilisateurs

erreur dans la condition



- Les fonctions

```
public class Screenshots {  
  
    public static void main(String[] args) {  
  
        int valeur1 = 3;  
        int valeur2 = 12;  
        int somme = fonctionAvecParametreEtRetour(valeur1, valeur2); (1)  
        fonctionAvecParametreSansRetour(valeur1, valeur2); (2)  
        String nom = fonctionSansParametreAvecRetour(); (3)  
        fonctionSansParametreSansRetour(); (4)  
    }  
  
    public static void fonctionSansParametreSansRetour() { (4)  
        System.out.println("Pensez à signer l'émargement");  
    }  
  
    public static String fonctionSansParametreAvecRetour() { (3)  
        return "Etienne";  
    }  
  
    public static void fonctionAvecParametreSansRetour(int param1, int param2) { (2)  
        System.out.println(param1 + " + " + param2 + " = " + (param1 + param2));  
    }  
  
    public static int fonctionAvecParametreEtRetour(int param1, int param2) { (1)  
        return param1 + param2;  
    }  
}
```

pas de limite sur une fonction
(bonne pratique: ne pas excéder 7

- Les I/O (Input Output) depuis un fichier. Ecrire dans un fichier :

```
public static void main(String[] args) {  
    // Les valeurs à écrire dans le fichier  
    String[] valeurs = new String[] {"Daniel", "Clémence", "Michel", "Dark Vador"};  
  
    try {  
        // On crée ou on écrase le fichier "valeur.txt"  
        FileWriter f = new FileWriter("valeurs.txt");  
        for (String current : valeurs) {  
            // On écrit les valeurs une à une  
            f.write(current + "\n");  
        }  
        System.out.println("Fin de l'enregistrement");  
        // On ferme l'accès au fichier  
        f.close();  
    } catch (IOException e) {  
        System.err.println("Une erreur est survenue");  
        e.printStackTrace();  
    }  
}
```

Attention: S'il existait , on l'écrase
de plus on mettrait des doubles \ pour
échapper le 1er

Try... catch : génère des erreurs (pb de droit,...)

Eclipse la met an auto: met un message d'erreur
et dit quel appel de fonction a fait planter



- Les I/O (Input Output) depuis un fichier. Ecrire dans un fichier :

```
public static void main(String[] args) {  
    // Les valeurs à écrire dans le fichier  
    String[] valeurs = new String[] {"Daniel", "Clémence", "Michel", "Dark Vador"};  
    // On "essaie" de créer le fichier, mais l'opération peut échouer  
    try {  
        // On crée ou on écrase le fichier "valeurs.txt"  
        FileWriter f = new FileWriter("valeurs.txt");  
        for (String current : valeurs) {  
            // On écrit les valeurs une à une  
            f.write(current + "\n");  
        }  
        System.out.println("Fin de l'enregistrement");  
        // On ferme l'accès au fichier  
        f.close();  
    } catch (IOException e) {  
        // En cas d'échec, on affiche "une erreur est survenue"  
        System.err.println("Une erreur est survenue");  
        e.printStackTrace();  
    }  
}
```



- Les I/O (Input Output) depuis un fichier. Ecrire dans un fichier :

```
public static void main(String[] args) {  
    // Les valeurs à écrire dans le fichier  
    String[] valeurs = new String[] {"Daniel", "Clémence", "Michel", "Dark Vador"};  
    // On "essaie" de créer le fichier, mais l'opération peut échouer  
    try {  
        // On crée ou on écrase le fichier "valeurs.txt"  
        FileWriter f = new FileWriter("valeurs.txt");  
        for (String current : valeurs) {  
            // On écrit les valeurs une à une  
            f.write(current + "\n");  
        }  
        System.out.println("Fin de l'enregistrement");  
        // On ferme l'accès au fichier  
        f.close();  
    } catch (IOException e) {  
        // En cas d'échec, on affiche "une erreur est survenue"  
        System.err.println("Une erreur est survenue");  
        e.printStackTrace();  
    }  
}
```

On appelle « Exception »
les cas de figure nous
permettant de gérer les
cas d'erreur dans nos
programmes



- Les I/O (Input Output) depuis un fichier. Lire un fichier :

```
public static void main(String[] args) {  
    try {  
        FileInputStream fis = new FileInputStream("valeurs.txt");  
        Scanner sc = new Scanner(fis);  
        while (sc.hasNextLine()) {  
            System.out.println(sc.nextLine());    Va lire fichier.txt ligne par ligne  
        }  
        System.out.println("Fin de la lecture");  
        sc.close();  
        fis.close();  
    } catch (FileNotFoundException e) {    fichier n'existe pas  
        e.printStackTrace();  
    } catch (IOException e) {    par exemple :fichier corrompu  
        e.printStackTrace();  
    }  
}
```



Atelier 1

Un petit Scrabble





Java

Retour sur la syntaxe