

Lyli

Cours Java des vaisseaux

Planning des 3 jours

- Prise en main de Java
 - Variables
 - Conditions
 - Boucles
 - Fonctions
- Course des vaisseaux

Démarrage de la course aujourd'hui

Milieu d'après-midi

Nous ferons 2 courses par jour

Une en fin de matinée (11h30)

Une en fin de journée (16h30)

Fondamentaux Java

Définition d'un algorithme ?

Succession d'opérations déterministes

- Prend des données **en entrée**
- Produit un résultat **en sortie**

Quelques exemples d'algorithmes

- **Une recette de cuisine**
 - Prend des ingrédients en entrée
 - Produit un plat en sortie
- **Une partition de musique**
 - Prend des notes de musique en entrée
 - Produit une chanson en sortie

Découpage d'un algorithme

1. **Déclaration et initialisation** des données d'entrée
2. **Traitement** qui va appliquer la séquence d'opérations
3. **Résultat** qui est produit

Variables

**Une variable sert à mémoriser une
information pour la réutiliser / modifier plus
tard**

Variables

- On **réserve** une "case" mémoire en lui donnant un nom
- On précise le **type du contenu** (nombre entier, chaîne de caractères, etc)
- On **affecte** une valeur d'initialisation

Variable

```
// Déclaration d'une variable de type int (integer)  
// nommée age  
// Initialisée avec la valeur 14  
int age = 14;
```

```
System.out.println(age); // Affiche 14
```

Caractéristiques variables

- Le signe égal signifie **affecter à gauche**

```
int age = 14:
```

```
// On affecte la valeur 23 à la variable age
```

```
// Vous pouvez voir le signe = comme ceci <-
```

```
// age <- 23
```

```
age = 23;
```

Caractéristiques variables

- On ne peut déclarer une variable qu'une seule fois

```
int age = 14;  
int age = 10; // NOPE ! la variable age existe déjà
```

Caractéristiques variables





- Le type est définitif

```
int age = 14;  
age = "Hello"; // NOPE ! On ne peut mettre que des nombre entiers dans age
```


Liste des types de variables

Type	Définition	Exemple
<code>boolean</code>	Valeur booléen (Vrai / Faux)	<code>boolean aPerdu = true;</code>
<code>int</code>	Nombre entier	<code>int score = 345;</code>
<code>float</code>	Nombre à virgule	<code>float taille = 1.80f;</code>
<code>String</code>	Chaine de caractères / texte	<code>String name = "Bob";</code>

Opération sur les variables

- Additionner 
- Soustraire 
- Multiplier 
- Diviser 
- Les mêmes règles de priorités (multiplication avant addition)
- Vous pouvez utiliser des parenthèses pour préciser l'ordre des opérations

Opération sur les variables

```
float note1 = 10.5f;  
float note2 = 14.5f;  
float somme = note1 + note2;  
float moyenne = somme / 2;  
  
// Ou alors tout en une ligne  
float moyenne2 = (note1 + note2) / 2;
```

TP

- Déclarer une variable de chaque type (`boolean`, `int`, `float`, `String`)
- Initialiser chaque variable avec une valeur arbitraire
- Afficher chaque variable avec `System.out.println()`
- Modifier chaque variable numérique avec des opérateurs arithmétiques (+ - * /) et afficher le résultat

Conditions

Conditions

- Exécuter du code si une **condition est vraie**
- Comme un train qui prend une bifurcation de rails

```
// Pour rentrer dans le if, il faut que la condition soit vraie  
if (CONDITION) {  
    // Code exécuté si la condition est vraie  
}
```

Une condition vraie ?

Un test qui produit **true**

```
int age = 18;  
// Le résultat du test age == 18 produit true  
// on rentre donc dans ce if  
if (age == 18) {  
    System.out.println("Cette personne a 18 ans !");  
}
```

Liste des opérateurs

Test	Symbole	Exemple
A est égal à B	<code>==</code>	<code>if (A == B) { ... }</code>
A est différent de B	<code>!=</code>	<code>if (A != B) { ... }</code>
A est supérieur strictement à B	<code>></code>	<code>if (A > B) { ... }</code>
A est supérieur ou égal à B	<code>>=</code>	<code>if (A >= B) { ... }</code>
A est inférieur strictement à B	<code><</code>	<code>if (A < B) { ... }</code>
A est inférieur ou égal à B	<code><=</code>	<code>if (A <= B) { ... }</code>

Plus de conditions !

```
int age = 18;  
// Si l'age est supérieur ou égal à 18 ans  
if (age >= 18) {  
    // Exécuté SEULEMENT si la condition est vraie  
    System.out.println("Tu es majeur");  
} else {  
    // Exécuté SEULEMENT si la condition est fausse  
    // C'est-à-dire => if (age < 18)  
    System.out.println("Tu es mineur");  
}  
// Exécuté dans tous les cas, car en dehors du if / else  
System.out.println("Fin du programme")
```

Plus de conditions !

```
int score = 10;
int boost = 3;

if (boost == 1) {           // D'abord cette branche est évaluée, boost == 1 ?
    score = score + 15;
}
else if (boost == 2) {     // Ensuite cette branche, boost == 2 ?
    score = score + 50;
}
else if (boost == 3) {     // Ensuite cette branche, boost == 3 ?
    score = score + 100;
}
else {                     // Uniquement si aucune branche n'a été exécutée
    System.out.println("boost non valide");
}
```

Opérateurs logiques

On veut parfois exprimer **plusieurs conditions** dans un **if**

- Si A **ET** B sont vrais... => `if (A && B)`
- Si A **OU** B est vrai... => `if (A || B)`

Opérateur logique ET

```
int age = 18;  
boolean aLePermis = true;  
// Si les 2 conditions sont vraies, le if renvoie vrai  
// On rentre donc dans cette branche  
if (age >= 18 && aLePermis == true) {  
    System.out.println("Tu peux conduire la voiture");  
}
```

Table de vérité ET

A	B	Résultat
Vrai	Vrai	Vrai
Vrai	Faux	Faux
Faux	Vrai	Faux
Faux	Faux	Faux

TLDR; Vrai seulement si les 2 sont vrais

Opérateur logique OU

```
int age = 18;  
boolean aAutorisationParents = true;  
// Si au moins une des condition est vraie, le if renvoie vrai  
// On rentre donc dans cette branche  
if (age >= 18 || aAutorisationParents == true) {  
    System.out.println("Tu peux aller à ce festival");  
}
```

Table de vérité OU

A	B	Résultat
Vrai	Vrai	Vrai
Vrai	Faux	Vrai
Faux	Vrai	Vrai
Faux	Faux	Faux

TLDR; Faux seulement si les 2 sont faux

TP

- Initialisez une variable entière comme l'âge d'une personne et affichez un message si cette personne est majeure ou mineure
- Initialisez 2 variables entières et déterminez quelle est la valeur la plus petite des 2

TP

- Initialisez 2 variables entières et déterminez si le résultat du produit est positif **sans faire le calcul**
 - 2×2 donne un résultat positif
 - -3×2 donne un résultat négatif
 - -3×-3 donne un résultat positif
- Faîtes l'exercice qui affiche la mention d'un bachelier en fonction de sa note

TP

- Déterminez si la valeur d'une année est bissextile ou non. Une année est bissextile si elle satisfait une de ces 2 conditions (l'une OU l'autre) :
 - L'année est divisible par 4 ET NON divisible par 100
 - L'année est divisible par 400

TP

- Déterminez si une date est correcte
 - 3 variables entières qui représentent les jour, mois et année d'une date
 - Utilisez le code précédent pour déterminer si au mois de février il y a 28 ou 29 jours

Boucles

Boucles

- Parfois on veut faire la même action plusieurs fois de suite
 - Appliquer des dégâts à tous les personnages
 - Calculer une moyenne pour chaque étudiants
 - Afficher les emails de tous les participants
- **On va utiliser une boucle POUR**

Boucle POUR

- **Initialisation** : valeur de départ
- **Condition** : condition qui doit restée vraie pour rester dans la boucle
- **Itération** : instruction effectuée à chaque tour de boucle

```
for (initialisation; condition; itération) {  
    // Toutes instructions sont répétées ici tant que  
    // l'on reste dans la boucle for  
}
```

Boucle POUR

```
// for (initialisation; condition; itération)  
// J'initialise une variable i avec la valeur 0  
// On reste dans la boucle tant que i < 5  
// A chaque tour de boucle, on incrémente i  
for (int i = 0; i < 5; i = i + 1) {  
    System.out.println(i); // Affiche 0, 1, 2, 3, 4  
}
```


TP

- Affichez tous les nombres entre 0 et 100 inclus, par ordre croissant
- Affichez tous les nombres entre 100 et 0 inclus, par ordre décroissant
- Affichez la table de multiplication de la valeur d'une variable

TP

- Affichez tous les nombres multiples de 3 entre 0 et 100 par ordre croissant
 - En utilisant une valeur d'incrément spécifique pour votre boucle
 - En utilisant une valeur d'incrément de '1' et en utilisant un branchement et l'opérateur modulo

Fonctions


**Une fonction regroupe du code pour
l'utiliser comme une boîte noire**

Fonctions

1. On donne des **paramètres d'entrée**
2. La fonction fait son **traitement**
3. On récupère un **résultat**

Exemple de fonction

“ Je voudrais une fonction qui me donne la température en Fahrenheit ”

 **Fonctionnement boîte noire** => Je ne connais pas la formule, j'ai juste besoin de la conversion

Exemple de fonction

- Cette fonction s'appelle `toFahrenheit`
- Cette fonction prend 1 paramètre, `float tempCelcius`
- Cette fonction renvoie 1 valeur décimale, en `float`
- Ces 3 éléments constituent la **signature de la fonction**

```
float toFahrenheit(float tempCelcius)
```

Utilisation d'une fonction

```
// Rappel de la signature  
// float toFahrenheit(float tempCelcius)  
  
float temperature = 37.5f;  
  
// On définit une variable tempFahrenheit  
// On appelle la fonction toFahrenheit() en lui passant la valeur 37.5  
// La fonction renvoie un résultat qui est stocké  
// dans la variable tempFahrenheit  
float tempFahrenheit = toFahrenheit(temperature);  
  
System.out.println(tempFahrenheit); // Affiche 99.5
```


Définition d'une fonction

```
float toFahrenheit(float tempCelcius) {  
    // Cette variable n'existe qu'à l'intérieur de la fonction  
    // Dès que la fonction se termine, la variable disparaît  
    float fahrenheit = (celsius * 1.8) + 32;  
  
    // On renvoie un résultat de fonction (ce qu'elle produit)  
    // avec le mot clé "return"  
    return fahrenheit;  
}
```

TP

- Fonction `min` : 2 entiers en paramètres, 1 entier en sortie
 - Valeur minimum entre 2 nombre
 - Exemple : `min(2, 9)` retourne le résultat 2
- Fonction `abs` : 1 entier en paramètre, 1 entier en sortie
 - Valeur absolue d'un nombre
 - Exemple : `abs(-5)` retourne le résultat 5

TP

- Fonction `displayMultTable` : 1 entier en paramètre, affichage de la table
 - Affiche la table de multiplication du nombre passé en paramètre