Algorithmie

Partie I

Sommaire

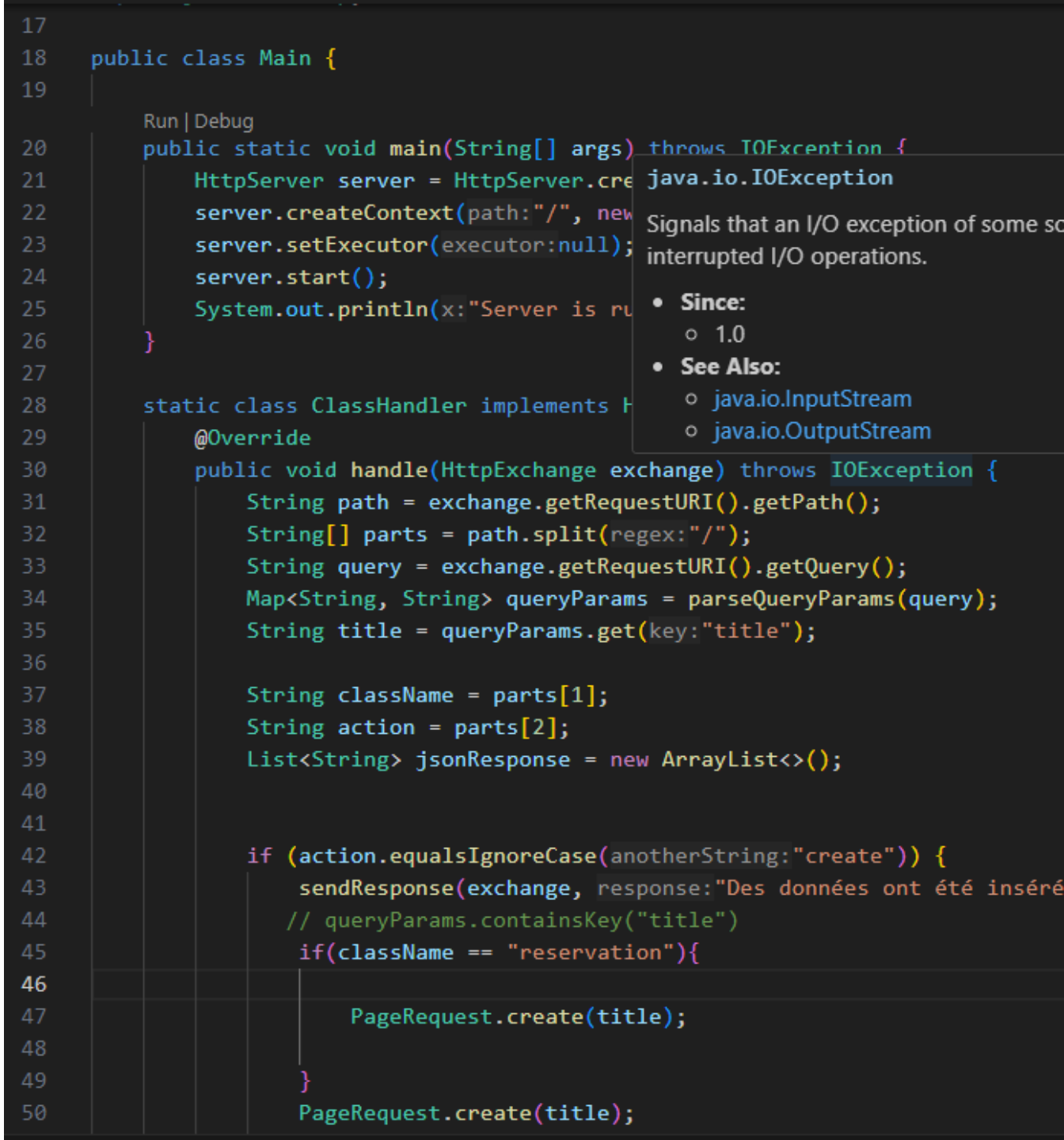
- Définition de l'Algorithmie
- Histoire de l'Algortihmie
- Langage de programmation
- Les variables
- Les types
- Opérateurs
- Condition
- Scanner
- Print

Définition

- Un algorithme est une suite d'instructions précises permettant de résoudre un problème ou d'accomplir une tâche, automatisée.

- Le but d'un algorithme d'effectuer une tâche par la machine avec le moins d'interventions de l'homme.

- L'homme se contente juste de donner des informations



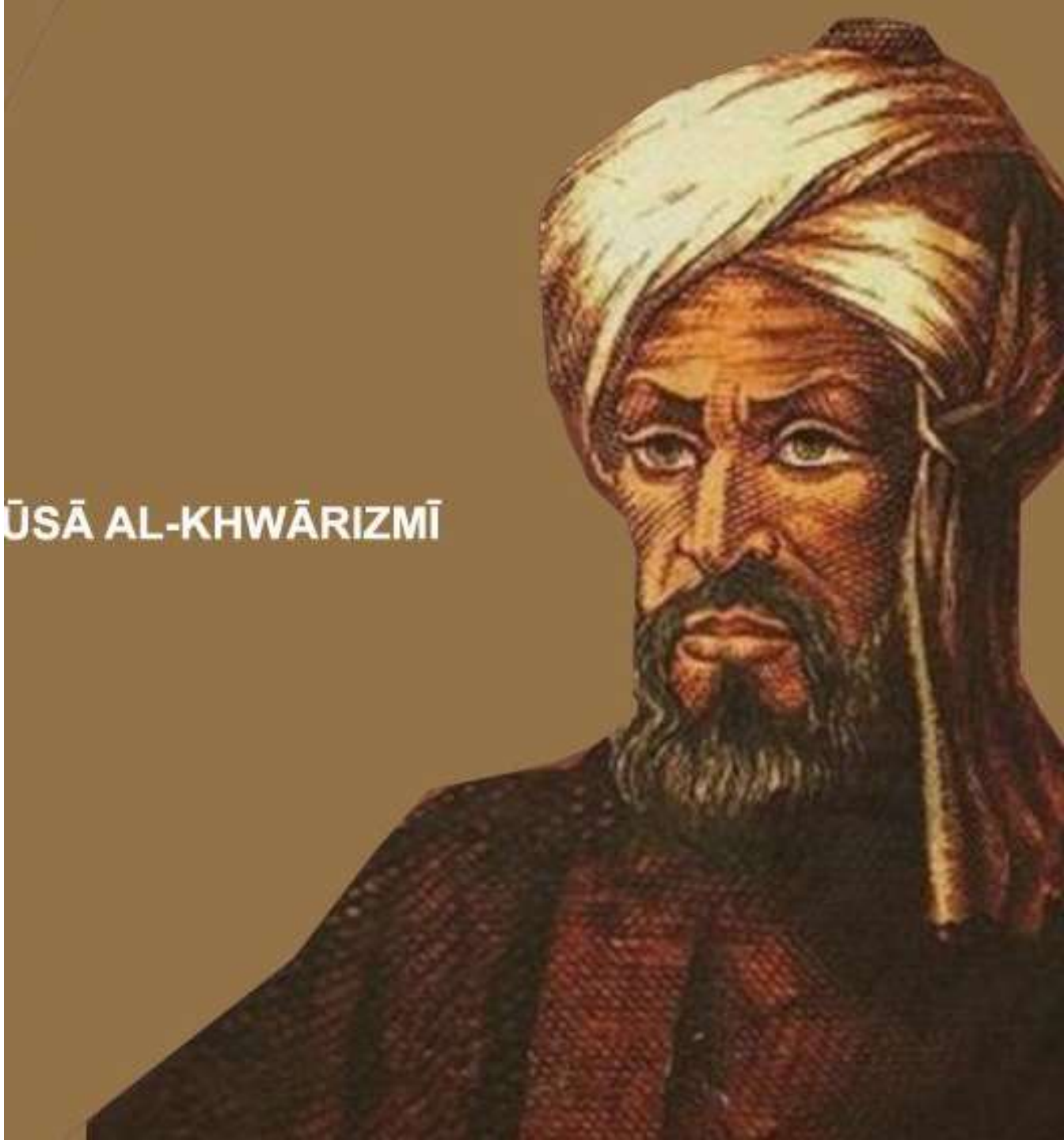

```
17
18  public class Main {
19
        Run | Debug
20      public static void main(String[] args) throws IOException {
21          HttpServer server = HttpServer.cre
22          server.createContext(path:"/", new
23          server.setExecutor(executor:null);
24          server.start();
25          System.out.println(x:"Server is ru
26      }
27
28      static class ClassHandler implements H
29          @Override
30          public void handle(HttpExchange exchange) throws IOException {
31              String path = exchange.getRequestURI().getPath();
32              String[] parts = path.split(regex:"/");
33              String query = exchange.getRequestURI().getQuery();
34              Map<String, String> queryParams = parseQueryParams(query);
35              String title = queryParams.get(key:"title");
36
37              String className = parts[1];
38              String action = parts[2];
39              List<String> jsonResponse = new ArrayList<>();
40
41
42              if (action.equalsIgnoreCase(anotherString:"create")) {
43                  sendResponse(exchange, response:"Des données ont été inséré
44                  // queryParams.containsKey("title")
45                  if(className == "reservation"){
46
47                      PageRequest.create(title);
48
49                  }
50                  PageRequest.create(title);
```

Histoire de l'Algorithmie

- **Al-Khwārizmī**, un savant perse du **IXe siècle ,** crée l'Algorithmie

- Al-Khwārizmī a créé des procédures logiques pour **résoudre des problèmes mathématiques.**

- Al-Khwārizmī **est sans le savoir le fondateur de l'informatique**



Histoire de l'Algorithmie

- **Alan Turing** a appliqué l'Algorithmie à la machine dans **les années 1940**

- Créant ainsi **l'informatique moderne.**

- **Les algorithmes** seront crucials pour concevoir **des logiciels** et **des systèmes d'exploitations.**

- **L'algorithmie permet entre-autre** de faire fonctionner de manière autonome un ordinateur, une application ou un site web dynamique.

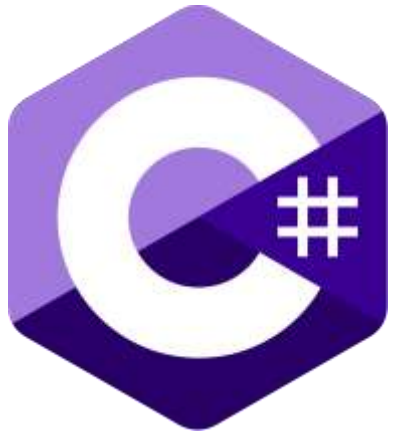
Langage de programmation

- **Un langage de programmation** est un langage utilisé pour écrire des instructions compréhensibles par un ordinateur afin de lui faire exécuter des tâches.

- Il sert à **traduire la logique d'un algorithme** en un code que la machine peut interpréter ou compiler pour fonctionner.

- Il existe **plus de 700 langages de programmation** recensés, mais tous ne sont pas utilisés aujourd'hui.

- **Les plus utilisées sont : Python, Javascript, Java, C# , C/ C++**



Langage Java

Pourquoi on va travailler sur JAVA ?

- ***Très utilisé dans le monde professionnel*** *: Java est un standard dans les grandes entreprises,*

- **Accessible pour les débutants** :Syntaxe est claire et bien structurée, ce qui le rend adapté à l'apprentissage.

- **Favorise la logique** : Java met davantage l'accent sur la **logique de programmation** et la **résolution de problèmes** que sur la connaissance approfondie du matériel ou du système.

- **Stable et mature** : Java évolue lentement comparé à d'autres langages, ce qui signifie moins de changements soudains à apprendre et une base solide de documentation.

Pseudo Code

Met avant de coder en Java, il est important de faire du Pseudo-Code

- Le pseudo code est un schéma de programmation , une première ébauche de votre algorithme

- Le pseudo code permet aussi de travailler votre logique de programamtion.

- Le but est d'être capable d'interprété votre propre code et de savoir en avance le résultat attendu.

- Faire du pseudo code vous permet aussi de limiter des erreurs dans votre code.

TYPE ENTIER VARIABLE x

TYPE ENTIER VARIABLE y

x = 1

y = 2

Si x < 1

Donc Affiche VRAI

Sinon Affiche FAux

Types

- Les **types de variables** permettent de définir **la nature des données** qu'une variable peut contenir (ex : un nombre, un texte, un vrai/faux…).

- Cela permet à l'ordinateur de savoir **comment traiter la donnée**, **quelles opérations sont autorisées**, et de **prévenir certaines erreurs**.

Type de variable	Description	Exemple de valeur
Entier (`int`)	Nombre entier, sans virgule	`5` , `-42` , `2025`
Décimal (`float`)	Nombre à virgule, précision moyenne	`3.14` , `-0.5`
Double (`double`)	Décimal avec plus de précision	`3.1415926535` , `2.71828`
Texte (`string`)	Suite de caractères	`"Bonjour"` , `"42"`
Booléen (`bool`)	Valeur logique : vrai ou faux	`true` , `false`
Caractère (`char`)	Un seul caractère	`'A'` , `'z'` , `'3'`
Liste (`array`)	Ensemble de valeurs du même type	`[1, 2, 3]` , `["a", "b"]`
Date (`date`)	Représente une date ou une date et une heure	`2025-04-21` , `21/04/2025 15:30`

Opérateurs

- Les **opérateurs** en algorithmie servent à **effectuer des opérations** sur des variables ou des valeurs.

- Ils sont indispensables pour manipuler les données, faire des calculs, comparer des valeurs, prendre des décisions ou encore modifier des contenus.

- **Deuxième partie du tableau page suivante =>**

Catégorie	Opérateur	Signification / Utilité	Exemple
Affectation	`←` ou `=`	Affecte une valeur à une variable	`x ← 5` ou `x = 5`
Arithmétiques	`+`	Addition	`a + b`
	`-`	Soustraction	`a - b`
	`*`	Multiplication	`a * b`
	`/`	Division (entière ou réelle selon le langage)	`a / b`
	`%`	Modulo (reste de la division entière)	`7 % 3` donne `1`
Logiques	`ET` ou `AND`	Vrai si les deux conditions sont vraies	`A ET B`
	`OU` ou `OR`	Vrai si au moins une condition est vraie	`A OU B`
	`NON` ou `NOT`	Inverse la valeur logique	`NON A`

Opérateurs

Comparaison	`=`	Égal à	`a = b`
	`≠` ou `!=`	Différent de	`a ≠ b`
	`<`	Inférieur à	`a < b`
	`>`	Supérieur à	`a > b`
	`≤` ou `<=`	Inférieur ou égal à	`a ≤ b`
	`≥` ou `>=`	Supérieur ou égal à	`a ≥ b`
Concaténation	`+` ou `&`	Assemble deux chaînes de caractères	`"Hello" + "World"`
Incrément/Décrément	`++` / `--`	Ajoute ou retire 1 à une variable (selon les langages)	`i++`, `i--` (pas en pseudo-code)

En Java on utilise `&&`, `||`, `!`. Pour le ET, OU et NON

Note importantes

- **equals( ) : Méthode pour vérifier si deux chaînes de caractères sont identiques. Utilisez equals en Java pour tester si deux chaînes de caractères sont identiques exemple :**

```
if(!password.equals(confirmpassword)){ // Vérifier si les mots de passes ne sont pas id

    System.out.println("inscription annulée "); /* Afficher votre instruction */


}else{
```

Scanner

Un **Scanner**, c'est un **objet** (dans certains langages comme Java) ou une **fonction** qui permet de **récupérer ce que l'utilisateur saisit**, pour ensuite l'utiliser dans le programme.

```java
import java.util.Scanner;

public class Exemple {
    public static void main(String[] args) {
        Scanner clavier = new Scanner(System.in); // création du scanner
        System.out.print("Entrez votre nom : ");
        String nom = clavier.nextLine(); // lecture d'une ligne
        System.out.println("Bonjour " + nom);
    }
}
```

Méthode Java	Ce qu'elle lit
`nextLine()`	Une **ligne complète** de texte
`next()`	Un **mot**
`nextInt()`	Un **entier** (ex : 42)
`nextDouble()`	Un **nombre à virgule**
`nextBoolean()`	Une **valeur booléenne** (true/false)

Condition

En **algorithmie**, les **conditions** permettent de **prendre des décisions**.

Elles indiquent **quoi faire selon qu'une situation est vraie ou fausse**.

C'est ce qui rend un algorithme **intelligent et adaptatif**, plutôt que linéaire.

Il faut utiliser le "If"

If(Instruction) {

(execution si VRAI)

}else if (2e instruction){

(execution si 2e VRAI)

}else{

(execution si FAUX)

}

```
if (className.equalsIgnoreCase(ano
    PageRequest.findAll().forEach(
        page.getId(),
        page.getTitle(),
        page.getDescription(),
        page.getContent(),
        page.isVisibility(),
        page.getDatePublished(),
        page.getTag()
    ));
} else if (className.equalsIgnoreCa
    MenuRequest.findAll().forEach(
        menu.getId(),
        menu.isTopmenu(),
        menu.isVisibility(),
        menu.getSubmenus()
    ));
} else {
    sendResponse(exchange, "{\"erro
    return;
}
```

Condition

Vous avez deux types de conditions :

- **If else :** Si la condition est vrai alors on lance le résultat sinon autres choses.

- **Switch :** Change le résultat en focntion de la valeur

```
if (className.equalsIgnoreCase(ano
    PageRequest.findAll().forEach(
        page.getId(),
        page.getTitle(),
        page.getDescription(),
        page.getContent(),
        page.isVisibility(),
        page.getDatePublished(),
        page.getTag()
    ));
} else if (className.equalsIgnoreC
    MenuRequest.findAll().forEach(
        menu.getId(),
        menu.isTopmenu(),
        menu.isVisibility(),
        menu.getSubmenus()
    ));
} else {
    sendResponse(exchange, "{\"err
    return;
}
```

Boucle

En **algorithmie**, les boucles permet de créer une itération c'est à dire une action qui se répète d'un point A à un point B, ou simple s'arrête lorsque la condition devient vrai

```
For ( I=0; I<=10; I++){

    Affiche ( I );

}
```

```
if (className.equalsIgnoreCase(ano
    PageRequest.findAll().forEach(
        page.getId(),
        page.getTitle(),
        page.getDescription(),
        page.getContent(),
        page.isVisibility(),
        page.getDatePublished(),
        page.getTag()
    ));
} else if (className.equalsIgnoreC
    MenuRequest.findAll().forEach(
        menu.getId(),
        menu.isTopmenu(),
        menu.isVisibility(),
        menu.getSubmenus()
    ));
} else {
    sendResponse(exchange, "{\"err
    return;
}
```

Boucle

Vous avez 3 types de boucles :

For : Boucle classique idéale pour les compteurs

While : Tant que la condition est fausse alors ça continue

Do While : Lance une première itération vrai puis continue si elle est fausse

Foreach : Idéale pour parcourir un tableau

```
if (className.equalsIgnoreCase(ano
    PageRequest.findAll().forEach(
        page.getId(),
        page.getTitle(),
        page.getDescription(),
        page.getContent(),
        page.isVisibility(),
        page.getDatePublished(),
        page.getTag()
    ));
} else if (className.equalsIgnoreC
    MenuRequest.findAll().forEach(
        menu.getId(),
        menu.isTopmenu(),
        menu.isVisibility(),
        menu.getSubmenus()
    ));
} else {
    sendResponse(exchange, "{\"err
    return;
}
```

Print

Le mot-clé ou la fonction **`print`** (ou **`afficher`** en pseudo-code) sert à **afficher des informations à l'écran**. C'est **l'un des outils les plus utilisés en programmation**, surtout pour :

```java
System.out.println("Bienvenue !");
```

Un peu de pratique

- Me declarer une variable type entire, lui donner une Valeur puis l’afficher

- Avec Scanner & Print, je me faire afficher votre message d'entrée.

- Avec scanner et If , me faire un algorithme qui permet de faire la somme de deux valeurs entiers et me dire si la valeur est supérieur à 10.

- **Pseudo code et code Java demandé**