

# Programmation Orientée Objets Langage Java

---

DR. DhiaEddineSaied

# Plan du cours

---

Chapitre 1: Introduction à la programmation en Java

Chapitre 2: Syntaxe et Éléments de base de langage Java

# Chapitre 1:


---

## INTRODUCTION À LA PROGRAMMATION EN JAVA



# Historique de Java (4/2)

---

- 1990: Le développement de Java a commencé par une équipe de Sun dirigée par James Gosling. Le projet, qui portait au début le nom **Oak** (chêne en anglais), avait pour but de développer un nouveau langage de programmation indépendant du système, orienté objet et léger
- 1992: Première présentation interne des possibilités de **Oak**. Un appareil appelé Star Seven permet de visualiser une animation montrant Duke, la mascotte actuelle de Java 
- 1993: Sun redirige ce langage vers Internet et diffuse HotJava, un navigateur internet entièrement écrit en Java
- 1995 : Sun a changé le nom du langage de **Oak** à **Java** (café en argot américain) car il était déjà utilisé par un autre langage de programmation
- Mai 1995: Lancement officiel de Java 1.0

# Historique de Java (5/2)

---

- Depuis sa première diffusion en 1995, le langage et les plateformes Java ont subi de nombreuses améliorations, on peut citer :

- 1996 : Lancement du JDK 1.0.1
- 1997 : Lancement du JDK 1.1
- 1998 : Lancement de J2SE 1.2
- 1999 : Lancement de J2EE 1.2
- 2000 : Lancement de J2SE 1.3
- 2001 : Lancement de J2EE 1.3
- 2002 : Lancement de J2SE 1.4
- 2003 : Lancement de J2EE 1.4
- 2004 : Lancement de J2SE 5.0
- 2006 : Lancement de JavaSE 6
- 2009 : Lancement de Java EE 6
- 2010 : Sun Microsystem est racheté par Oracle.  
Désormais, Java est maintenu par la société Oracle.
- 2011 : Lancement de JavaSE 7
- 2013 : Lancement de Java EE 7
- 2014 : Lancement de JavaSE 8
- 2017 : Lancement de Java SE 9, Java EE 8
- 2018 : Lancement de Java SE 10, Java SE 11
- 2019 : Java SE 12, Java SE 13, Jakarta EE 8
- 2020 : Java SE 14, Java SE 15, Jakarta EE 9
- 2021 : Java SE 16, Java SE 17, Jakarta EE 9.1
- 2022 : Java SE 18, Jakarta EE 10



**Évolution très rapide et succès du langage**

# Atouts de Java (6/3)

---

## ○ Java est Familier :

- Syntaxe proche de celle de C/C++
  - Même types de données (int, float, double, etc.) et Même formes de déclarations
  - Même structure de contrôle : if, for, while, etc.

## ○ Java est Simple :

- Apprentissage facile
- Pointeurs et goto éliminés

## ○ Java est Ouvert et Distribué :

- Possède une importante bibliothèque (**java.net**) pour
  - l'accès à des objets distants sur Internet via des URL (Universal Resource Locators)
  - la programmation client/serveur via des sockets TCP et UDP
  - l'exécution des méthodes distantes (RMI : Remote Method Invocation)
  - la gestion de serveurs Web via les Servlets
- Connexion intégrée aux bases de données (JDBC)

# Atouts de Java (7/3)

---

- **Java est Orienté Objet :**

- Java ne permet d'utiliser que des objets
- Objet est une entité regroupant un ensemble de données et de méthodes de traitement

- **Java est Fiable :**

- Gestion automatique de la mémoire
  - Possède une fonctionnalité appelée le Ramasse-miettes (ou Garbage Collector) qui se charge de libérer la mémoire lorsqu'il détecte qu'elle n'est plus utilisée
- Gestion des exceptions
  - Ouverture d'un fichier inexistant, division par zéro, création d'un point de communication réseau (socket) vers une @IP inexistante, . . . Le programmeur est **forcé de gérer** diverses exceptions
- Sources d'erreurs limitées
  - Typage fort, Pas d'héritage multiple, Pas de manipulations de pointeurs, etc.

# Atouts de Java (8/3)

---

## ○ Java est Interprété :

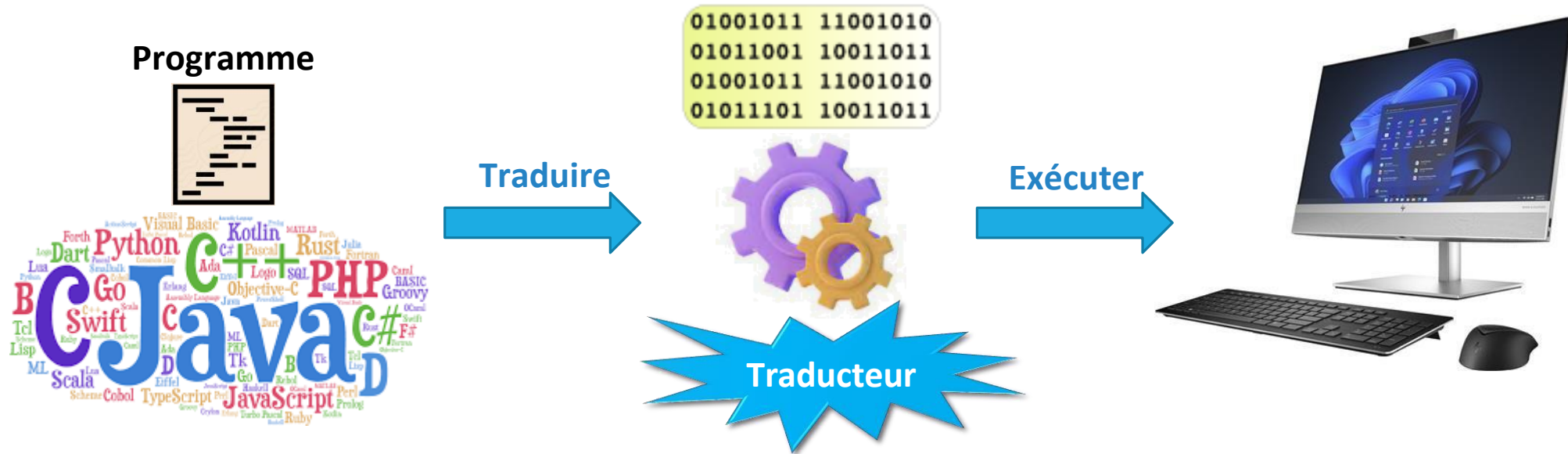
- Un code écrit en Java n'est pas exécuté tel quel
  - Le compilateur Java transforme le code Java en un fichier intermédiaire appelé **Bytecode**
  - L'interpréteur Java exécute le **Bytecode** directement sur n'importe quelle plateforme à l'aide d'une machine virtuelle **JVM** (**J**ava **V**irtual **M**achine)

## ○ Java est portable et indépendant de l'architecture :

- Le **Bytecode** généré par le compilateur est indépendant de la plate-forme
- Les bibliothèques sont intégrées de manière standard au langage
- Un programme écrit en Java sur une plateforme peut être exécuté sans aucune modification sur un autre système, à condition bien sûr qu'une machine virtuelle **JVM** soit disponible sur ce dernier



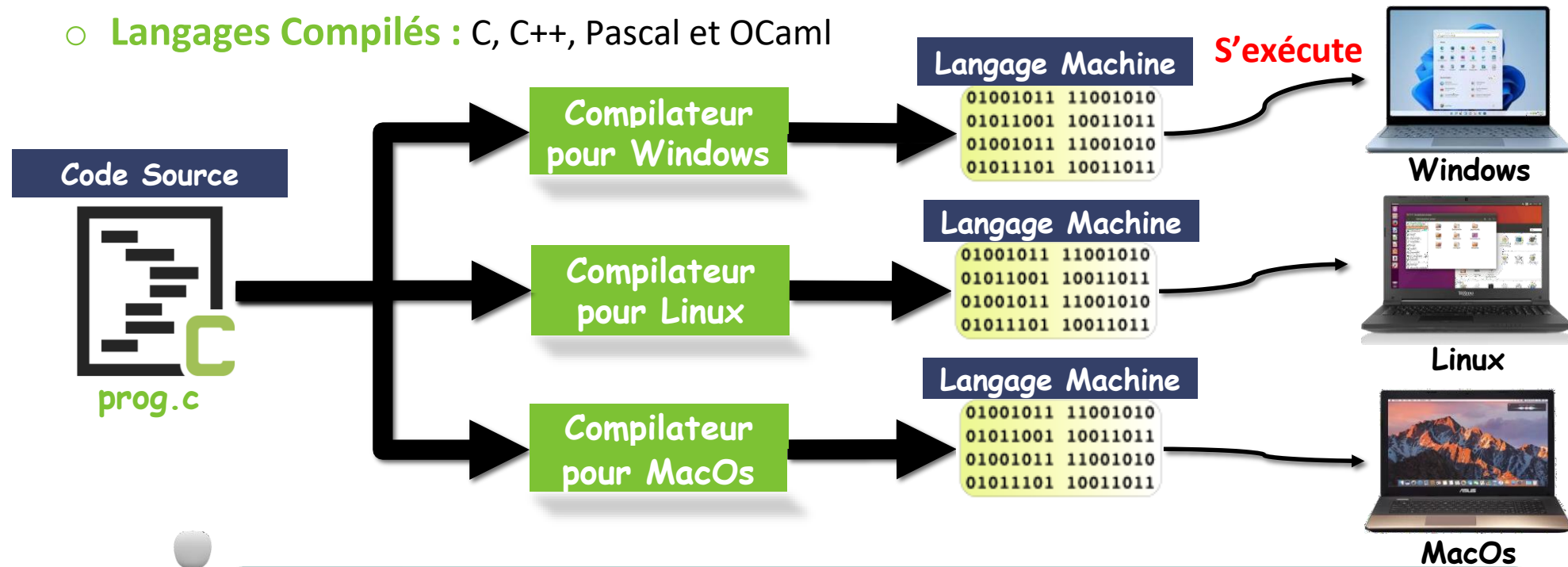
# Déploiement d'un programme (9/4)



- Il existe deux façons pour convertir le **code source** en **langage machine**
  - **Compiler** le code : c'est à dire le traduire en binaire
  - **Interpréter** le code : c'est à dire le lire en temps réel et exécuter les instructions

# Déploiement d'un programme

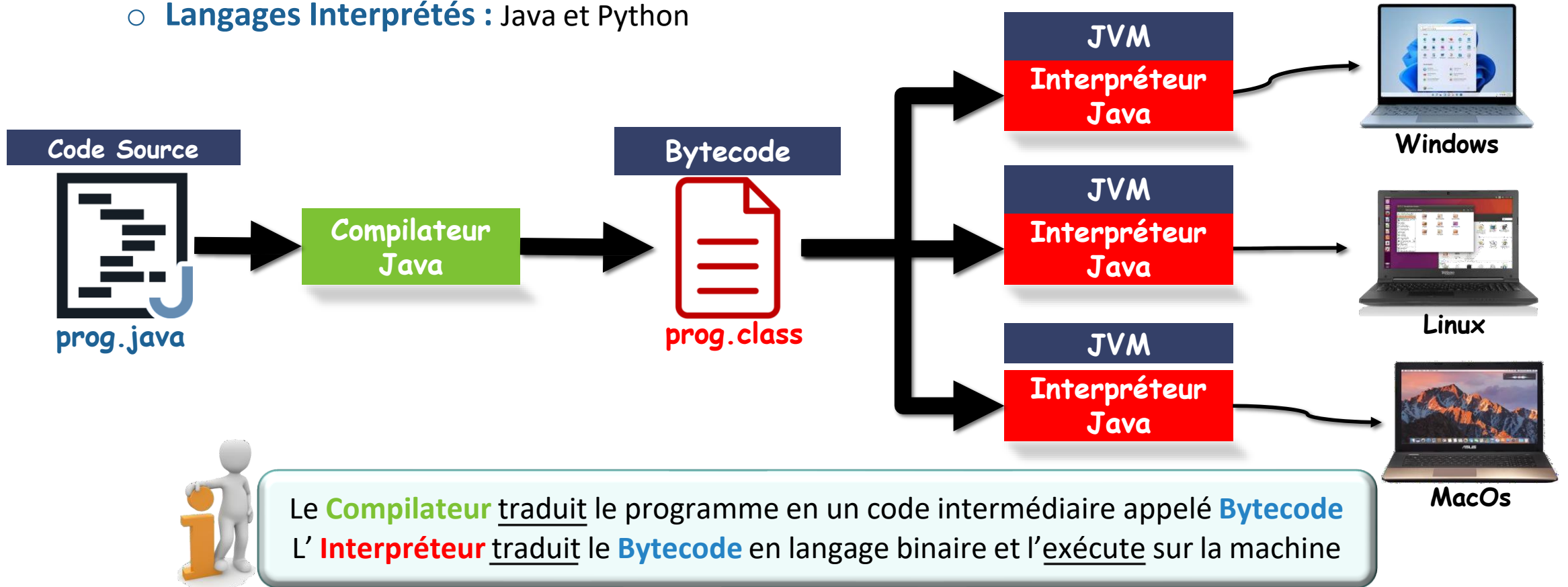
- **Langages Compilés** : C, C++, Pascal et OCaml



Le **Compilateur** génère du code natif directement exécutable, **mais** spécifique à chaque environnement

# Déploiement d'un programme

- **Langages Interprétés :** Java et Python



# Déploiement d'un programme

---

## LANGAGE COMPILÉ (EXP. C++)

+ Rapidité d'exécution

### **Mais**

- Demande beaucoup plus de lignes de code
- Plus dure à apprendre
- Code compilé dépend de la plateforme

## LANGAGE INTERPRÉTÉ (EXP. JAVA)

+ Portabilité (Indépendant de la plateforme)

+ Code est plus léger

+ Plus simple à écrire

### **Mais**

- Plus lent que les langages compilés

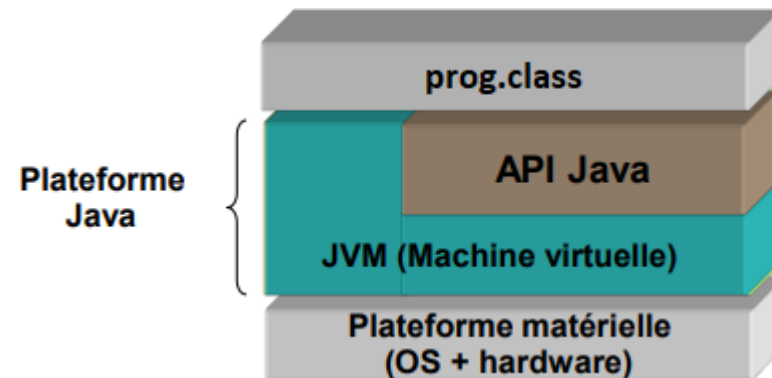


Écrire une fois,  
exécuter partout

# Plateforme Java (1/3)

---

- La **plateforme Java** est entièrement logicielle et s'exécute au dessus des plateformes matérielles
- La **plateforme Java** est constituée de :
  - Des interfaces de programmation d'application (API Java)
  - Machine virtuelle (JVM)



# Plateforme Java – API Java (2/3)

---

- **Java Application Programming Interface (Java API)**

- Vaste collection de composants logiciels (classes et interfaces)
- Organisée en paquetages (ou packages)
- Offre de nombreuses fonctions de manière standard (indépendamment de la plateforme matérielle)
- Les principaux packages :
  - java.util : structures de données classiques
  - java.io : entrées / sorties
  - java.lang : chaînes de caractères, interaction avec l'OS, threads
  - java.applet : les applets sur le web
  - java.awt : interfaces graphiques, images et dessins
  - javax.swing : création d'interfaces graphiques
  - java.net : sockets, URL
  - java.rmi : Remote Method Invocation
  - java.sql : fournit le package JDBC

# Plateforme Java – JVM (3/3)

---

- **Java Virtual Machine (JVM)**

- Les machines ne peuvent pas comprendre un **Bytecode**. Il s'agit d'un type de code non exécutable qui devient compréhensible par la machine après qu'un **interpréteur Java** l'a traduit en code machine au fur et à mesure que le programme est exécuté



C'est le rôle de la **JVM**

- Ainsi, tout système disposant déjà d'une **JVM** peut facilement exécuter un tel code quel que soit le système d'exploitation

# Étapes de développement d'une application en Java (16/14)

---

- Il existe deux manières pour écrire un programme en Java
  - **En écrivant le code dans un simple éditeur de texte**
    - Compilation et exécution du code en ligne de commande (DOS)
  - **En utilisant un environnement de développement (IDE)**
    - Eclipse (<http://www.eclipse.org>)
    - Netbeans (<http://www.netbeans.com>)
    - Borland JBuilder (<http://www.borland.com/jbuilder>)
    - IBM WebSphere Studio (<http://www.ibm.com/software/awdtools>)
    - Sun ONE Studio (<http://wwws.sun.com/software/sundev>)
    - Microsoft .Net Studio (<http://msdn.microsoft.com/vstudio>)



# Étapes de développement d'une application en Java (17/14)

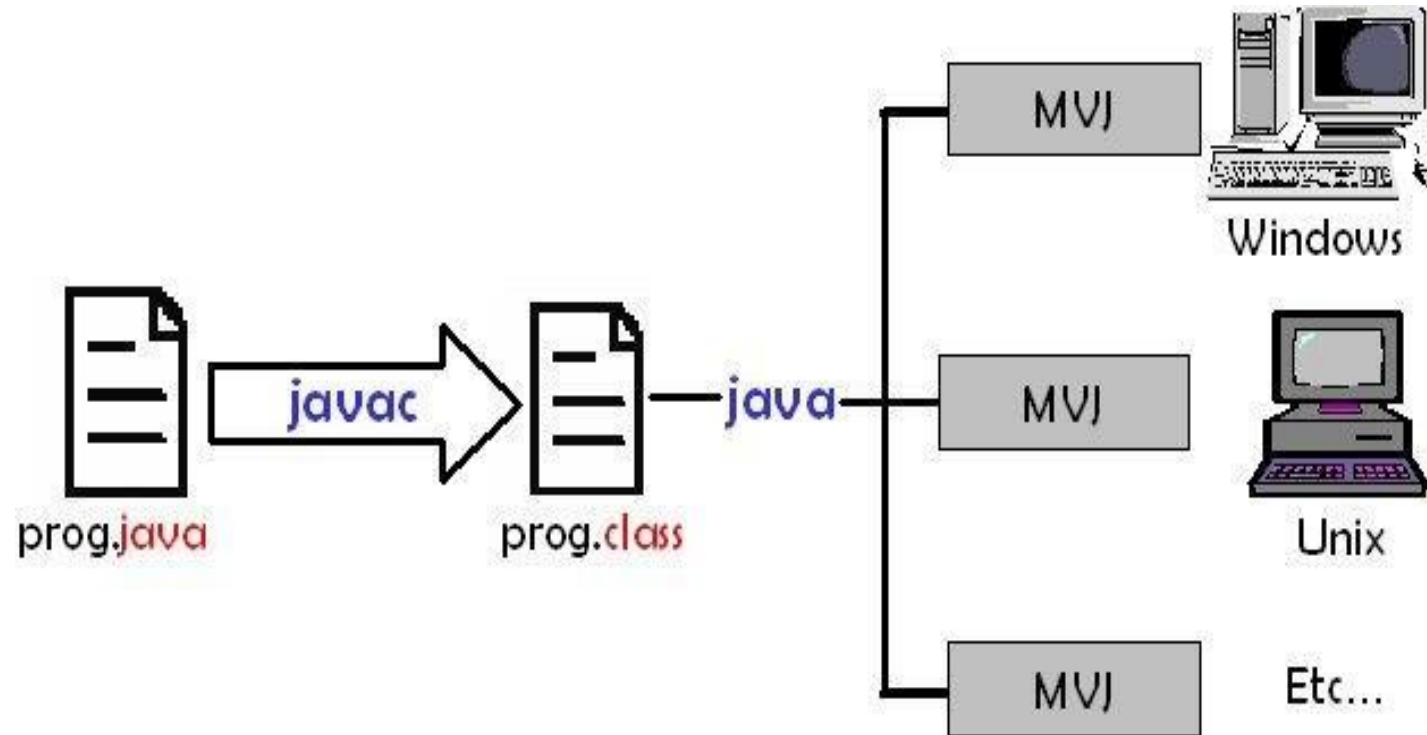
---

## ○ Écriture du code en utilisant un éditeur de texte

- Téléchargez la dernière version du **JDK** (Java Development Kit)
- Le **JDK** est l'ensemble des outils nécessaire pour développer et exécuter une application Java.
- Outils fournis par le **JDK** :
  - Le compilateur **javac**: Compile les fichiers sources **.java** en fichiers Bytecode **.class**
  - L'interpréteur **java**: Prend en paramètre le nom de la classe, cherche le ou les fichiers **.class** qui lui correspondent et appelle la méthode **main** de la classe
  - Le documenteur **javadoc**: Générateur de documentation d'API
  - L'utilitaire **Jar**: Permet de grouper et compresser des fichiers utilisés par un programme Java
  - L'interpréteur d'applet **appletviewer**: Un programme permettant de tester les applets Java, prévues pour être intégrées dans des pages HTML
  - Le générateur d'interface avec C, **javah** : Un programme permettant de lier des programmes Java avec des méthodes natives, écrites dans un autre langage et dépendants des systèmes
  - Etc.

# Étapes de développement d'une application en Java (18/14)

- Écriture du code en utilisant un éditeur de texte



# Étapes de développement d'une application en Java (19/14)

## ○ Écriture du code en utilisant un éditeur de texte

### 1. Créer un fichier texte : HelloWorld.java

```
public class HelloWorld
{
    public static void main (String[] args)
    {
        System.out.println("Hello the World");
    }
}
```

Tout code java doit être défini à l'intérieur d'une classe

Le point d'entrée pour l'exécution est la méthode main qui porte la signature ci-contre

Écrire à l'écran "Hello the World"

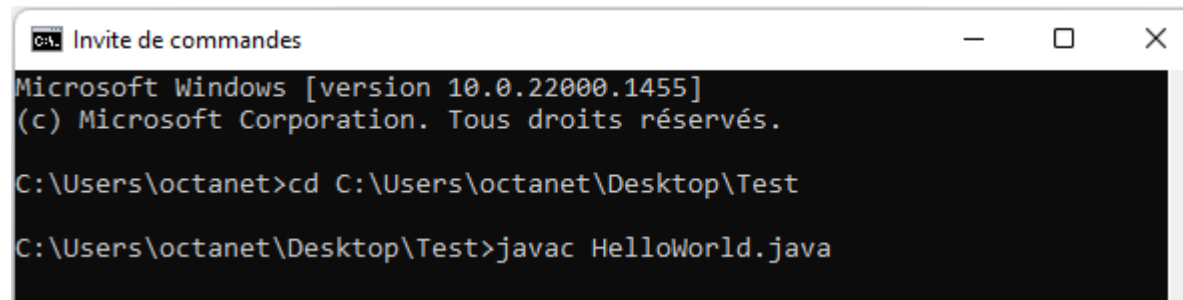
La description de la classe est effectuée à l'intérieur d'un bloc {}

# Étapes de développement d'une application en Java (20/14)

---

- **Écriture du code en utilisant un éditeur de texte**

2. Compiler le programme : `javac HelloWorld.java`

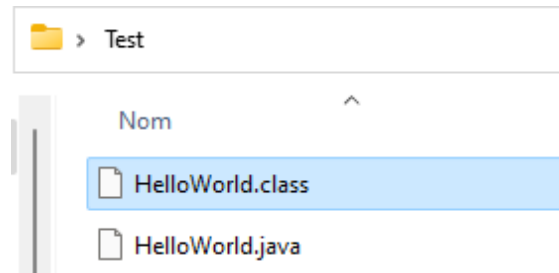


```
Invite de commandes
Microsoft Windows [version 10.0.22000.1455]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\octanet>cd C:\Users\octanet\Desktop\Test

C:\Users\octanet\Desktop\Test>javac HelloWorld.java
```

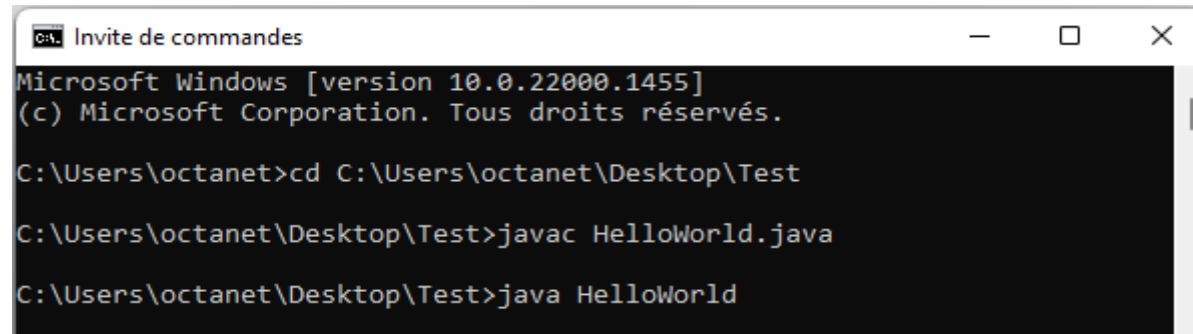
3. Le compilateur génère le Bytecode dans le fichier : **HelloWorld.class**



# Étapes de développement d'une application en Java (21/14)

- **Écriture du code en utilisant un éditeur de texte**

4. Exécuter l'application : `java HelloWorld`



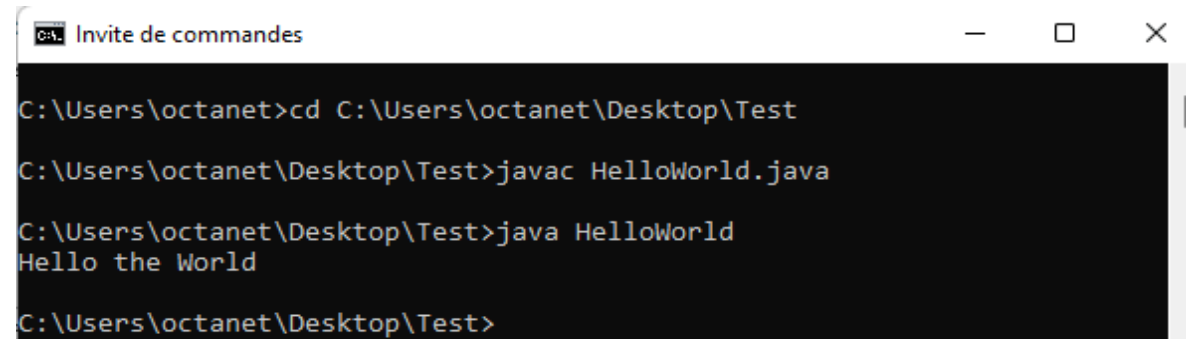
```
Invite de commandes
Microsoft Windows [version 10.0.22000.1455]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\octanet>cd C:\Users\octanet\Desktop\Test

C:\Users\octanet\Desktop\Test>javac HelloWorld.java

C:\Users\octanet\Desktop\Test>java HelloWorld
```

5. **Hello the World** s'affiche sur l'écran



```
Invite de commandes

C:\Users\octanet>cd C:\Users\octanet\Desktop\Test

C:\Users\octanet\Desktop\Test>javac HelloWorld.java

C:\Users\octanet\Desktop\Test>java HelloWorld
Hello the World

C:\Users\octanet\Desktop\Test>
```

# Étapes de développement d'une application en Java (22/14)

---

## ○ Écriture du code en utilisant un environnement de développement (IDE)



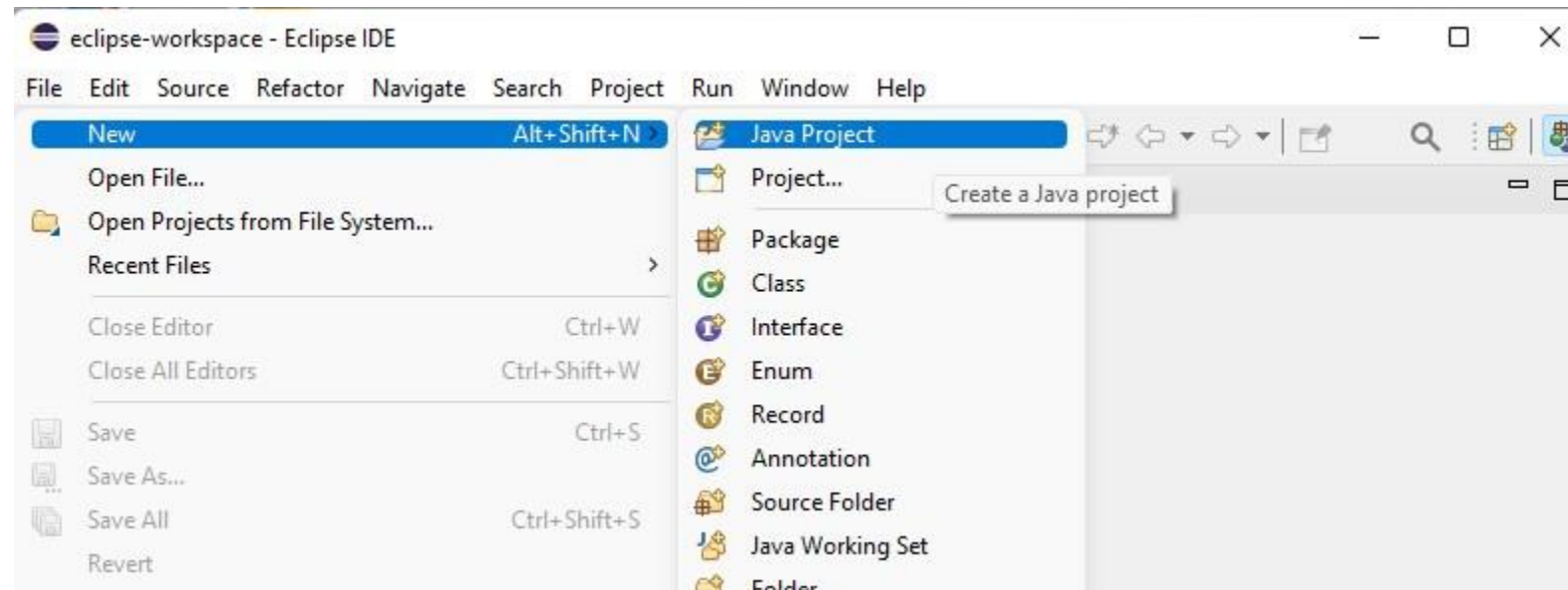
- **Eclipse** est un Environnement de Développement Intégré (IDE)
- Spécialement conçu pour le développement en Java
- Créé à l'origine par IBM, puis cédé à la communauté Open Source
- Caractéristiques principales
  - Notion de « projet » (1 programme → 1 projet)
  - Colore le code en fonction de la signification des mots utilisés
  - Compile le code en temps réel → Identifie les erreurs en cours de frappe
  - Peut générer des bouts de code automatiquement
  - Permet de gérer le lancement des applications

# Étapes de développement d'une application en Java (23/14)

- **Écriture du code en utilisant un environnement de développement (IDE)**

1. Créer un projet:

a. **File → New → Java Project**



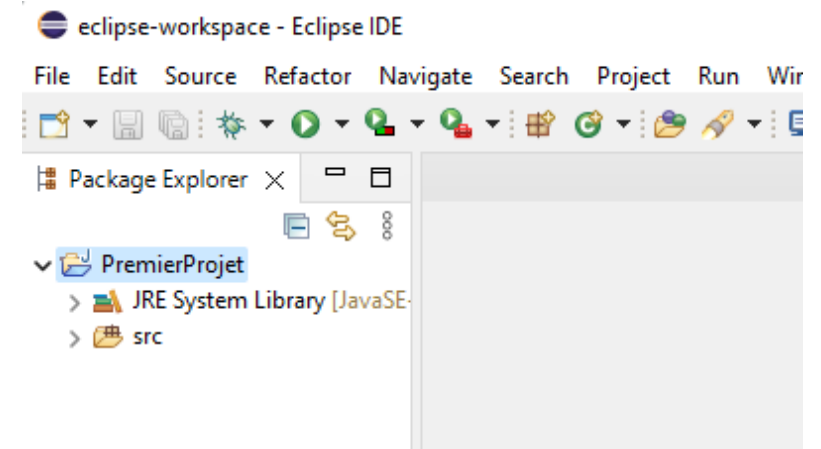
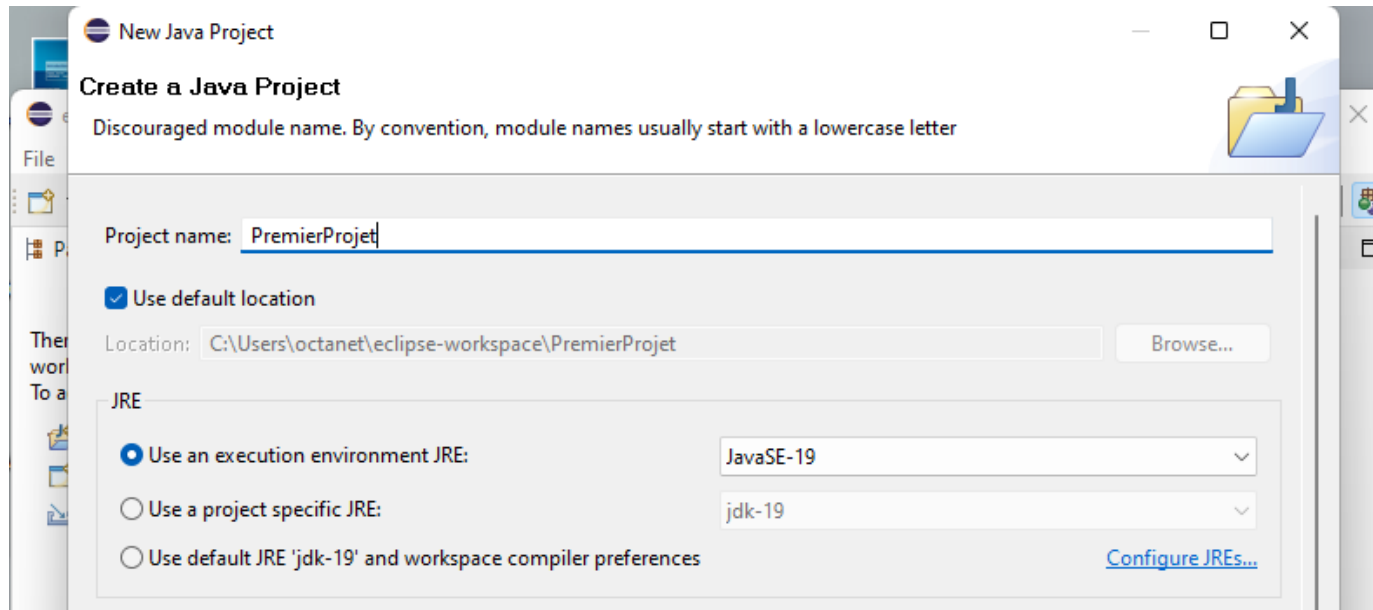
# Étapes de développement d'une application en Java (24/14)

## ○ Écriture du code en utilisant un environnement de développement (IDE)

1. Créer un projet:

a. **File → New → Java Project**

b. **Project name** → Exp. « PremierProjet »





# Étapes de développement d'une application en Java (25/14)

---

- **Écriture du code en utilisant un environnement de développement (IDE)**



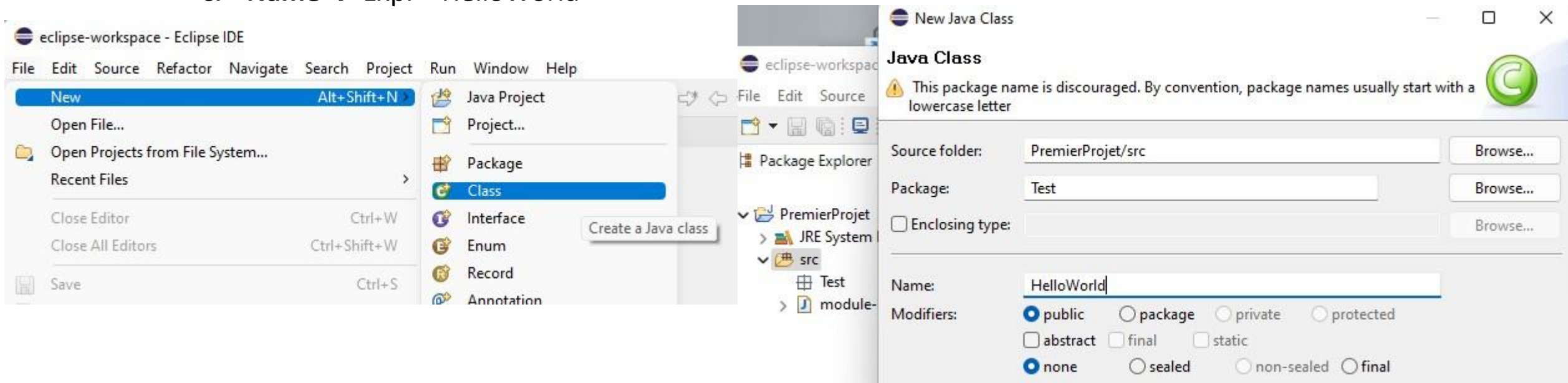
- Notre projet a été créé, alors nous pouvons commencer le développement de l'application
  - Une application Java est composée de « Classes »
  - Chaque classe correspond à un fichier
  - Le fichier source doit prendre le nom de la classe et son extension est **.java**
  - Java est dit « case-sensitive » → **Distingue majuscules et minuscules!!!**

# Étapes de développement d'une application en Java (26/14)

## ○ Écriture du code en utilisant un environnement de développement (IDE)

### 2. Créer une Classe:

- a. **File** → **New** → **Class**
- b. **Package** → Exp. « Test »
- c. **Name** → Exp. « HelloWorld »



# Étapes de développement d'une application en Java (27/14)

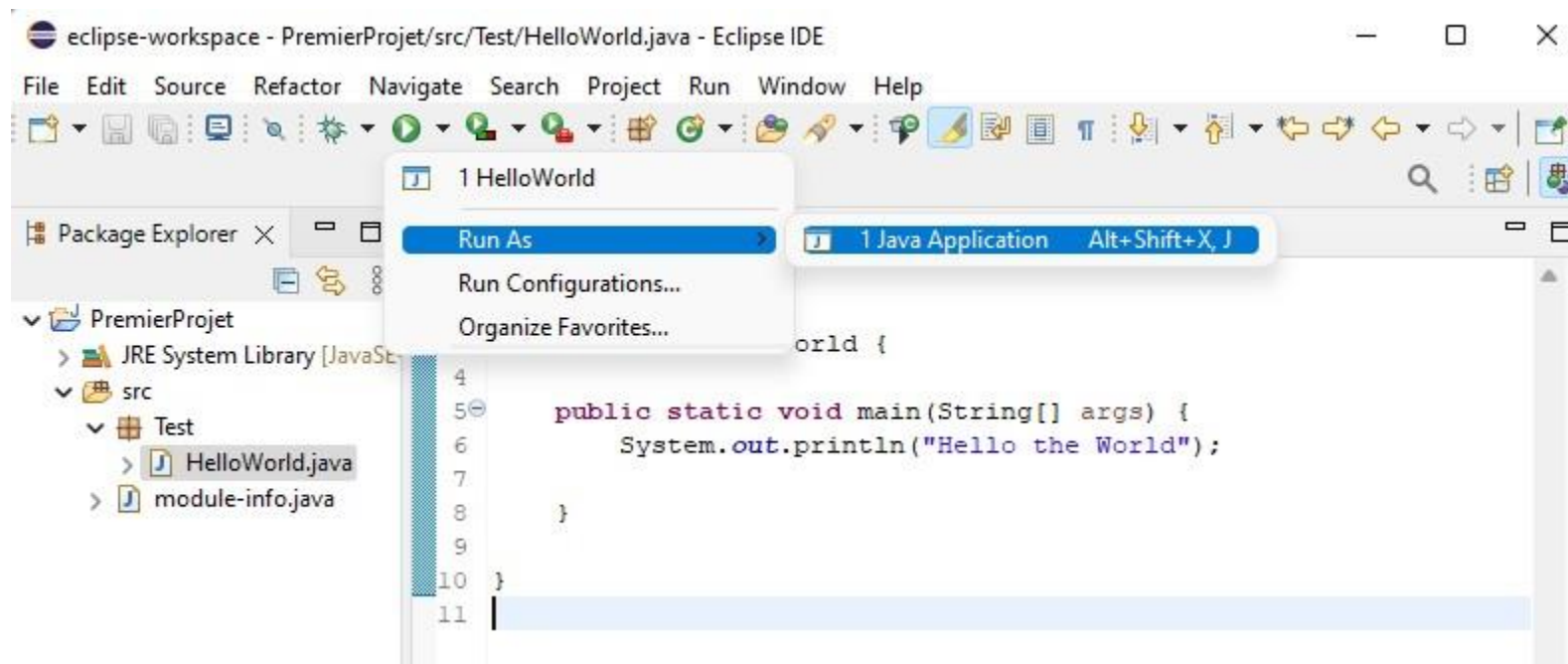
- **Écriture du code en utilisant un environnement de développement (IDE)**
- 3. Insérer l'instruction « `System.out.println(« Hello the World");` » dans le corps de la méthode `main`.

A screenshot of the Eclipse IDE interface. The title bar reads "eclipse-workspace - PremierProjet/src/Test/HelloWorld.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows the project structure: PremierProjet > JRE System Library [JavaSE] > src > Test > HelloWorld.java. The main editor window displays the code for HelloWorld.java:

```
1 package Test;
2
3 public class HelloWorld {
4
5     public static void main(String[] args) {
6         System.out.println("Hello the World");
7     }
8 }
9
10
11
```

# Étapes de développement d'une application en Java (28/14)

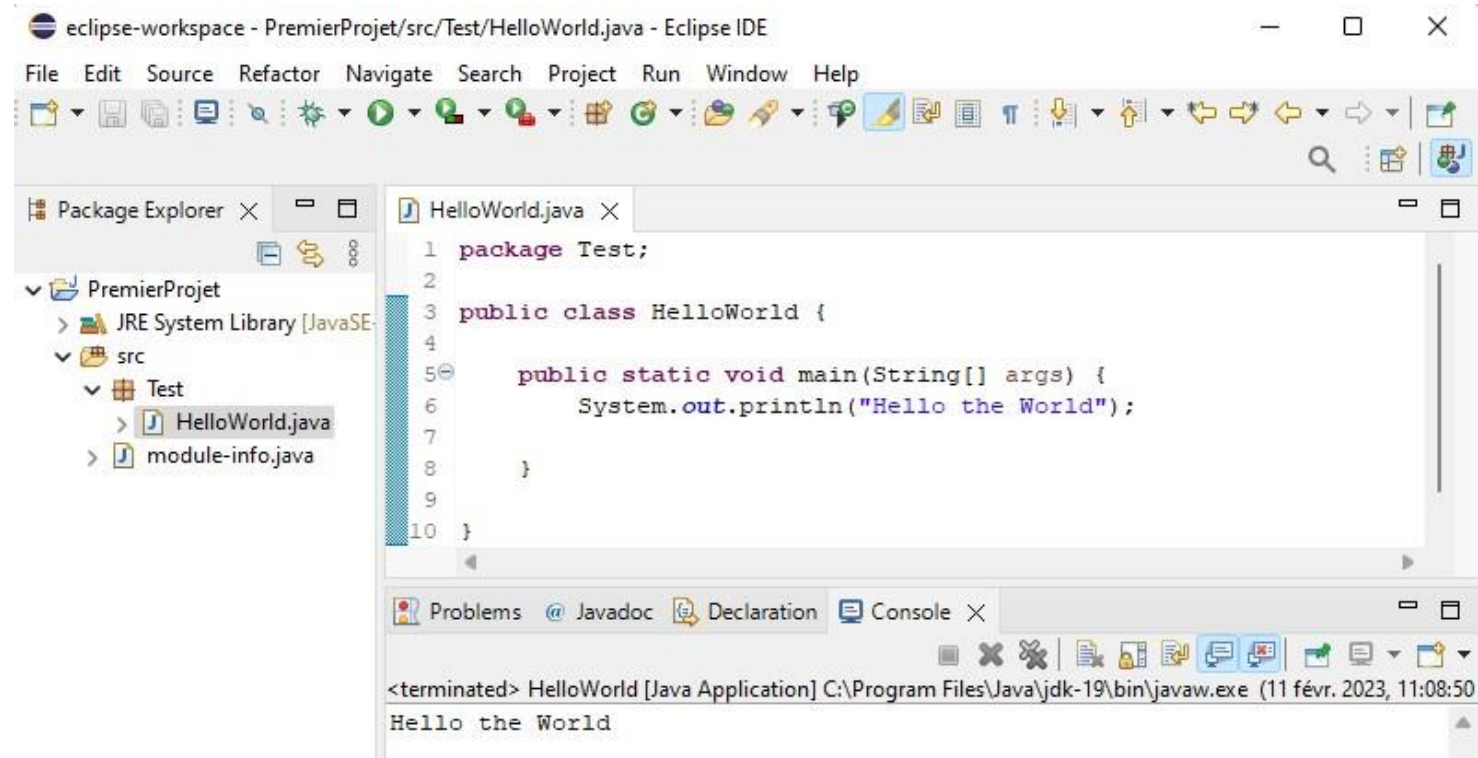
- **Écriture du code en utilisant un environnement de développement (IDE)**
4. Compiler et Exécuter le code source
    - a. **Run As → Java Application**



# Étapes de développement d'une application en Java (29/14)

- **Écriture du code en utilisant un environnement de développement (IDE)**

## 5. Affichage du résultat

A screenshot of the Eclipse IDE interface. The title bar reads "eclipse-workspace - PremierProjet/src/Test/HelloWorld.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows the project structure: PremierProjet > JRE System Library [JavaSE] > src > Test > HelloWorld.java. The main editor window displays the code for HelloWorld.java:

```
1 package Test;
2
3 public class HelloWorld {
4
5     public static void main(String[] args) {
6         System.out.println("Hello the World");
7     }
8 }
9
10
```

The Console window at the bottom shows the output: "<terminated> HelloWorld [Java Application] C:\Program Files\Java\jdk-19\bin\javaw.exe (11 févr. 2023, 11:08:50 Hello the World".