Java : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

Java

Sun Microsystems

Une entreprise qui travaillait principalement sur

- Stations de travail **Unix** : ordinateurs puissants pour ingénieurs et chercheurs.
- Système d'exploitation **Solaris** (basé sur **Unix**).
- Le langage **C**, le compilateur **Sun C**, et les environnements de développement.
- La technologie **RISC** (processeurs **SPARC**).
- **NFS** (**N**etwork **F**ile **S**ystem) : système de fichiers réseau toujours utilisé aujourd'hui.

Java

Début des années 90

- Une équipe d'ingénieurs, mené par **James Gosling** chez **Sun Microsystems**
- Travaillait sur un projet appelé **Green Project**
- Fatigués des complexités de **C++**
- Visant à développer une plateforme embarquée pour des appareils électroniques (comme des décodeurs TV)
- avait un rêve : un langage **plus simple, plus sûr, orienté objet et portable**

Java

Une réunion pas comme les autres

- Réunis autour d'une machine à café
- Discussions animées, idées folles
- Un objectif : éliminer les pointeurs, la gestion manuelle de mémoire, et les pièges du **C++**

Java

Le projet **Oak**

- Nom initial : **Oak**, en hommage à un chêne devant le bureau de **James Gosling**
- Mais ce nom était déjà pris : utilisé par une autre société (une entreprise informatique basée en Californie).
- Besoin d'un nouveau nom, plus percutant

Java

Java est né

- Devant leur tasse de café : **Java**
- En référence au café de l'île indonésienne
- Simple, mémorable, universel

Java

Conclusion

- **Java** n'est pas né sur l'île de Java...
- ... mais dans un esprit de simplicité et de caféine
- Une légende qui continue d'inspirer les développeurs : un des langages les plus utilisés dans le monde (`https://www.tiobe.com/tiobe-index/`)

Java

Contexte à la fin des années 90

- **Java** devient le langage phare des applications d'entreprise.
- **Sun Microsystems** pousse **NetBeans** comme **IDE** officiel officiel pour **Java**.
- **IBM** utilise **VisualAge for Java**, devenu rigide et dépassé.

Java

2001 : **IBM** lance **Eclipse**

- **IBM** développe un **IDE** modulaire, extensible par plugins : **Eclipse**.
- Décision stratégique : le rendre open source dès 2001.
- Objectifs
  - Briser le monopole de **Sun** sur les outils **Java**.
  - Créer un écosystème ouvert et communautaire.
  - Séduire entreprises et développeurs.

Java

Le succès d'**Eclipse**

- Large adoption dans le monde professionnel.
- Création de la **Eclipse Foundation**.
- **Eclipse** devient l'**IDE Java** dominant pendant plus d'une décennie.

Java

La chute de **Sun** et le rachat par **Oracle**

- **Eclipse** éclipse (sans jeu de mots) **NetBeans** en popularité.
- Perte d'influence de **Sun** sur les outils de développement **Java**.
- **Sun Microsystems** a été rachetée par Oracle en 2010.
- **Oracle** a surtout été intéressée par **Java** (très utilisé dans les applications d'entreprise) et par **MySQL**, que **Sun** avait acquis en 2008.
- Après l'acquisition, de nombreux projets open source de **Sun** (comme **OpenSolaris** ou **OpenOffice**) ont perdu du soutien.
- Certains ingénieurs de **Sun** ont quitté **Oracle** et ont créé des forks :
  - **MariaDB** à partir de **MySQL**,
  - **LibreOffice** à partir d'**OpenOffice**.

Java

Java

- langage de programmation
  - orienté objet
  - fortement typé
  - compilé
  - sensible à la casse
- présenté officiellement en 1995 par **Sun Microsystems**
- syntaxe très proche du **C** (procédural) et **C++** (procédural, orienté objet)

Java

Java

- langage de programmation
  - orienté objet
  - fortement typé
  - compilé
  - sensible à la casse
- présenté officiellement en 1995 par **Sun Microsystems**
- syntaxe très proche du **C** (procédural) et **C++** (procédural, orienté objet)

Attention

Java $\neq$ JavaScript

Java, pourquoi ?

- Langage de haut niveau (pas de gestion de mémoire, pas d'allocation dynamique, pas de pointeur... comme en **C** et **C++**)
- Disposant d'une bonne documentation, des supports vidéos, plusieurs exemples sur internet
- Énorme communauté : un des langages les plus utilisés dans le monde (`https://www.tiobe.com/tiobe-index/`)
- Permettant de développer des programmes :
  - robustes
  - sécurisés et fiables
  - bien structurés et facilement maintenables
  - portables : **Windows**, **Mac OS**, **Linux** (Write once, run everywhere ou Écrire une fois, exécuter partout)
  - ...

Java

Trois plateformes d'exécution

- **J**ava **S**tandard **E**dition (**J2SE** ou **Java SE** ou **JSE**)
  - applications consoles
  - applications du bureau ou Desktop
- **J**ava **E**nterprise **E**dition (**J2EE** ou **Java EE** ou **JEE**)
  - applications web
  - services web
- **J**ava **M**icro **E**dition (**J2ME** ou **Java ME** ou **JME**)
  - applications mobiles
  - applications embarquées
  - jeux

Différentes versions de **Java**

- Java 1 (1995)
- Java 1.2 (1998, nommée **Playground**) : Swing
- Java 1.3 (2000, nommée **Kestrel**) : JNDI...
- Java 1.4 (2002, nommée **Merlin**) : JAXP...
- Java 5.0 ou 1.5 (2004, nommée **Tiger**) : généricité, annotation, énumération
- Java 6.0 ou 1.6 (2006, nommée **Mustang**) : JAX-WS
- Java 7 ou 1.7 (2011, nommée **Dolphin**) : `String` dans `switch`
- Java 8 ou 1.8 (Mars 2014, nommée **Spider**) : interface fonctionnelle, expression Lambda
- Java 9 (Septembre 2017, nommée **Umbrella**) : **JSON** et **HTTP/2**
- Java 10 (Mars 2018) : mot-clé `var`
- Java 11 (Septembre 2018) : simplifier l'exécution d'un programme en ligne de commande
- Java 12 (Mars 2019) : simplification de `switch` et `String` multi-lignes
- Java 13 (Septembre 2019) : `->` dans `switch`
- Java 14 (Mars 2020) : Text Blocks
- Java 15 (Septembre 2020) : `Record`
- Java 16 (Mars 2021) : Intégration du langage **C++**
- Java 17 (Septembre 2021) : Classes scellées
- Java 18 (Mars 2022) : **UTF-8** par défaut
- Java 19 (Septembre 2022) : Threads virtuels
- Java 20 (Mars 2023) : Imbrication des `Record`
- Java 21 (Septembre 2023) : Simplification du `main`
- Java 22 (Mars 2024) : `ListFormat`
- Java 23 (Septembre 2024) : **Markdown** dans **JavaDoc**
- Java 24 (Mars 2025) : `ListFormat`

rouge $\Rightarrow$ version non supportée, bleu $\Rightarrow$ **LTS** (**L**ong **T**erm **S**upport) version.

Versions **LTS** (**L**ong **T**erm **S**upport)

- Les versions **LTS** bénéficient d'un support à long terme, souvent pendant 8 à 11 ans.
- **Oracle** offre généralement 5 ans de support premier (mises à jour de sécurité et corrections de bugs), suivis de 3 ans de support étendu (souvent payant).
- Exemple : la **JDK 11** (sorti en 2018) a un support allant jusqu'en 2026 par **Oracle**.

Versions **LTS** (**L**ong **T**erm **S**upport)

- Les versions **LTS** bénéficient d'un support à long terme, souvent pendant 8 à 11 ans.
- **Oracle** offre généralement 5 ans de support premier (mises à jour de sécurité et corrections de bugs), suivis de 3 ans de support étendu (souvent payant).
- Exemple : la **JDK 11** (sorti en 2018) a un support allant jusqu'en 2026 par **Oracle**.

Versions non **LTS**

- Les versions non **LTS** ont un cycle de vie beaucoup plus court : 6 mois à 1 an.
- Ces versions reçoivent des mises à jour de sécurité et des corrections de bugs uniquement jusqu'à la sortie de la version suivante.
- Elles ne sont pas destinées à être utilisées en production sur le long terme. Elles servent surtout à introduire de nouvelles fonctionnalités ou expérimentations qui seront stabilisées dans les **LTS** suivantes.

Java

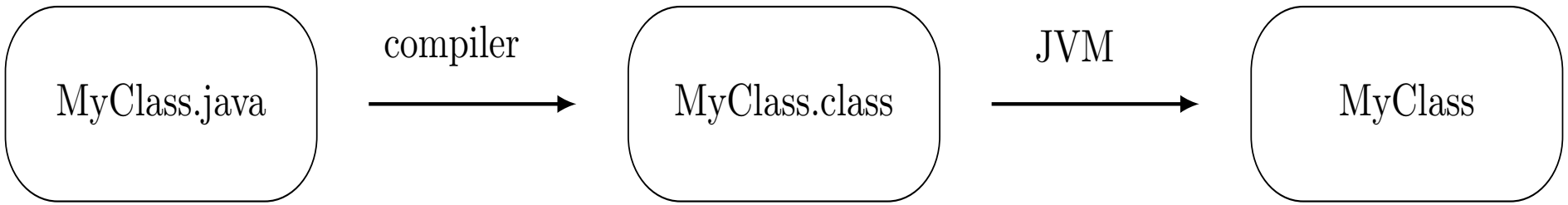
Remarque

Depuis **Java 17**, une version **LTS** sera proposée tous les deux ans.

Java

Comment ça fonctionne ?

- On écrit un programme dans un fichier `.java`
- Ensuite, le compilateur génère un fichier `.class` du même nom (contenant du bytecode)
- Puis, la machine virtuelle exécute le bytecode.



Java

De quoi on a besoin (le minimum) ?

- Un éditeur de texte (Bloc-notes, Notepad++, Sublime Text...)
- Un kit de développement (**JDK** : **J**ava **D**evelopment **K**it) contenant :
  - **J**ava **R**untime **E**nvironment (**JRE**, incluant la machine virtuelle de **Java** (**JVM**))
  - **JSE**, **JEE**, **JME**
  - Des commandes permettant la création, la compilation et l'exécution d'un programme **Java**
    - `javac` : pour compiler
    - `java` : pour exécuter
    - `javadoc` : pour générer une documentation
    - `jar` : pour archiver

Java

JDK

- Développé par **Oracle**.
- Usage commercial nécessite un abonnement.
- Versions optimisées et mises à jour plus stables.

OpenJDK

- Maintenu par une collaboration entre plusieurs entreprises (**Red Hat**, **Amazon**... et, des développeurs indépendants sous l'égide d'**Oracle**.
- Open-source.
- Versions non optimisées et mises à jour rapidement disponibles.

Java

JCP : **J**ava **C**ommunity **P**rocess

Organisation communautaire ouverte ayant comme rôle : définir et/ou améliorer les spécifications des technologies **Java**.

Java

JCP : **J**ava **C**ommunity **P**rocess

Organisation communautaire ouverte ayant comme rôle : définir et/ou améliorer les spécifications des technologies **Java**.

JSR : **J**ava **S**pecification **R**equest

Document utilisé par **JCP** pour décrire une proposition d'amélioration du langage **Java** et ses environnements.

Java

JCP : **J**ava **C**ommunity **P**rocess

Organisation communautaire ouverte ayant comme rôle : définir et/ou améliorer les spécifications des technologies **Java**.

JSR : **J**ava **S**pecification **R**equest

Document utilisé par **JCP** pour décrire une proposition d'amélioration du langage **Java** et ses environnements.

JEP : **J**DK **E**nhancement **P**rocess

Document utilisé pour décrire les nouveaux changement à **OpenJDK**.

Java

JDK 11 : téléchargement

```
https://www.oracle.com/java/technologies/javase/jdk11-archive-downloads.html
```

JDK 17 : téléchargement

```
https://www.oracle.com/java/technologies/javase/jdk17-archive-downloads.html
```

JDK 24 : téléchargement

```
https://www.oracle.com/java/technologies/downloads/
```

Java

Remarque

Pour lancer un programme en ligne de commande, il faut :

- aller dans `Panneau de configuration`, chercher `Système` et cliquer sur `Paramètres systèmes avancés`
- choisir `Variables d'environnement` puis dans la zone `Variables utilisateur` sélectionner `Path` et cliquer sur `Modifier`
- cliquer sur `Nouveau` puis saisir le chemin vers la **JDK** dans la zone de saisie qui a apparu
- valider

Java

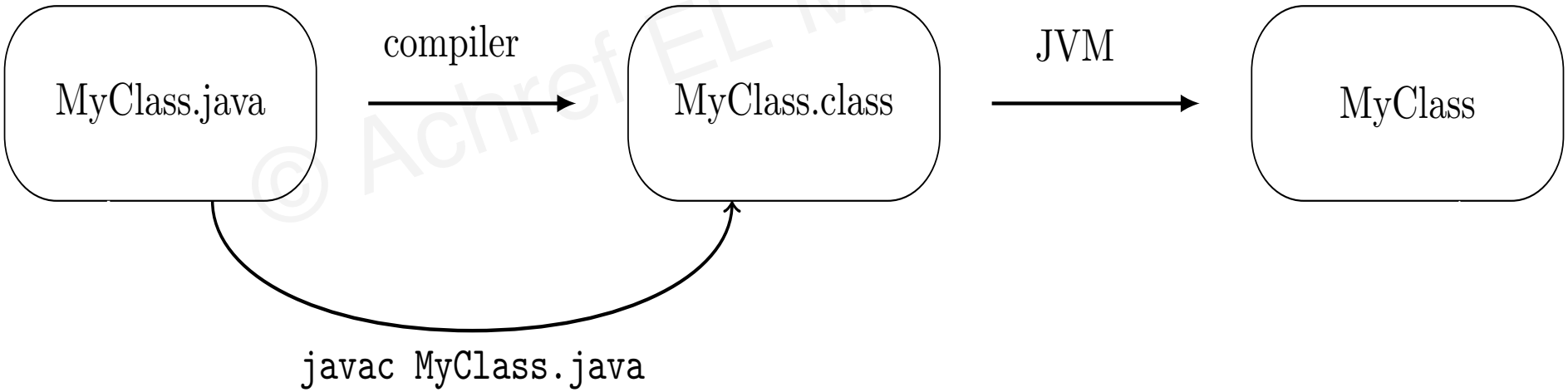
Créons une classe `MyClass` dans un fichier `MyClass.java`

```
public class MyClass {

  public static void main(String[] args) {
    System.out.print("Hello world from console");
  }
}
```

Java

Créons une classe `MyClass` dans un fichier `MyClass.java`

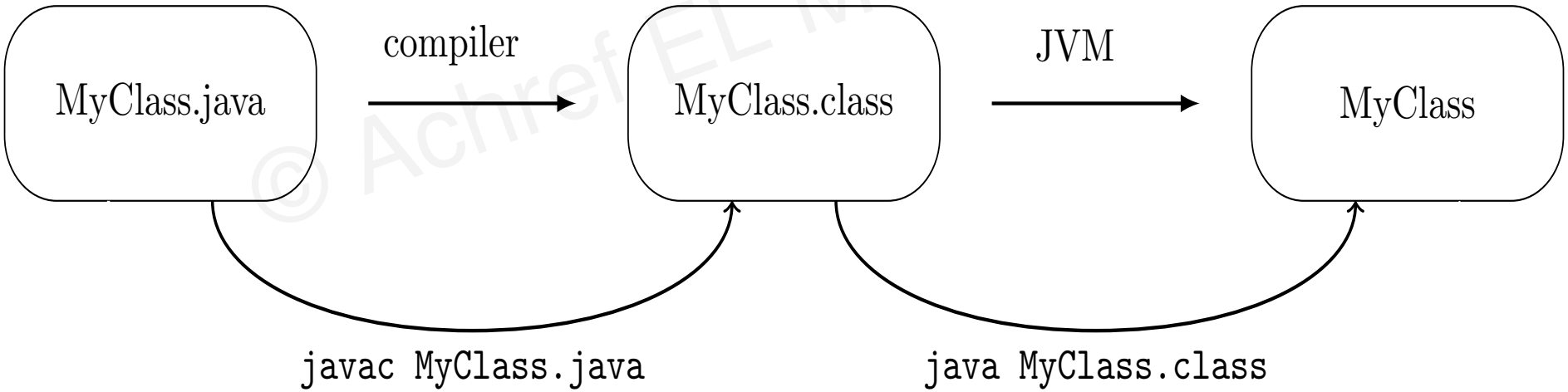
```
public class MyClass {

  public static void main(String[] args) {
    System.out.print("Hello world from console");
  }
}
```



Java

Créons une classe `MyClass` dans un fichier `MyClass.java`

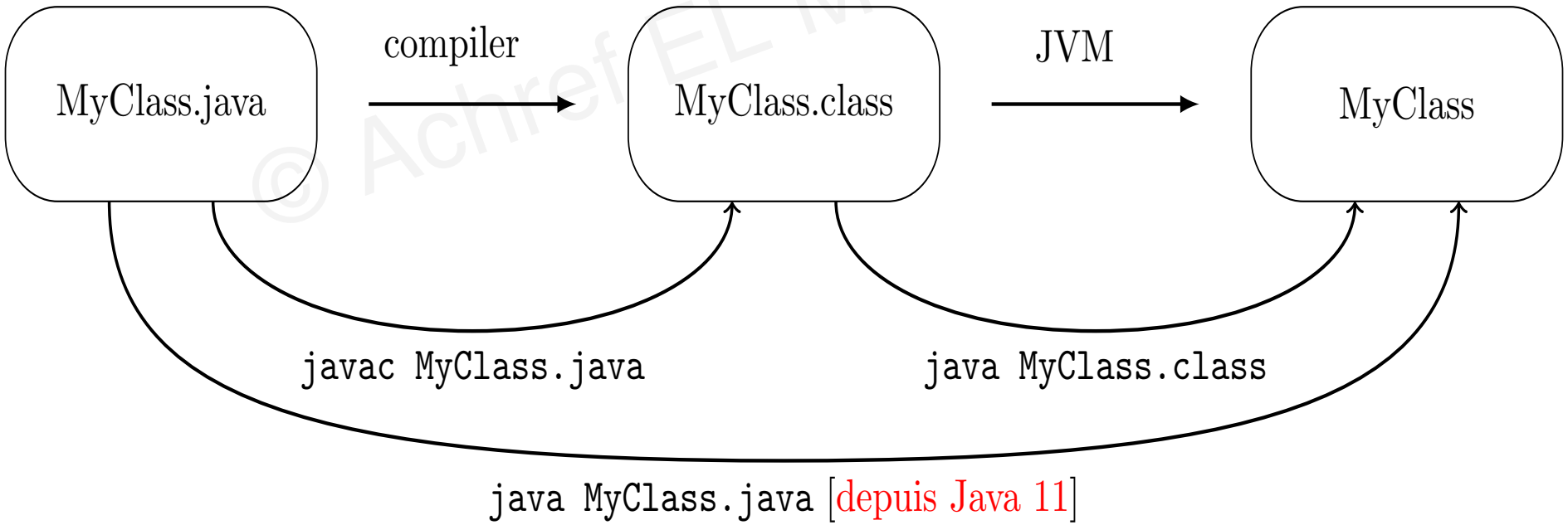
```
public class MyClass {

  public static void main(String[] args) {
    System.out.print("Hello world from console");
  }
}
```



Java

Créons une classe `MyClass` dans un fichier `MyClass.java`

```
public class MyClass {

  public static void main(String[] args) {
    System.out.print("Hello world from console");
  }
}
```



Java

Pour compiler

```
javac MyClass.java
```

Java

Pour compiler

```
javac MyClass.java
```

S'il existe plusieurs versions de JDK sur la machine

```
javac -target 8 -version 8 MyClass.java
```

Java

Pour compiler

```
javac MyClass.java
```

S'il existe plusieurs versions de JDK sur la machine

```
javac -target 8 -version 8 MyClass.java
```

Pour exécuter (ça affiche `Hello world from console`)

```
java MyClass
```

Java

On peut aussi utiliser un **IDE** (Environnement de développement intégré)

- pour éviter d'utiliser la console et les commandes
- car un **IDE** intègre un compilateur lancé même pendant l'écriture du code
- pour profiter de la coloration syntaxique, l'auto-complétion, l'indentation automatique...
- pour avoir une bonne structuration du projet

Java

Exemple d'**IDE** pour **Java**

- **Eclipse**
- Netbeans
- JDeveloper
- IntelliJ IDEA
- JBuilder
- JCreator...
- ...

Java

Eclipse, pourquoi ?

- open-source
- écrit en **Java**
- multi-langage : **Java**, **C++**, **PHP**, **Cobol**, **C#**, **JavaScript**...
- multi-OS : **Windows**, **Linux**, **Mac**...

Java

Eclipse : téléchargement

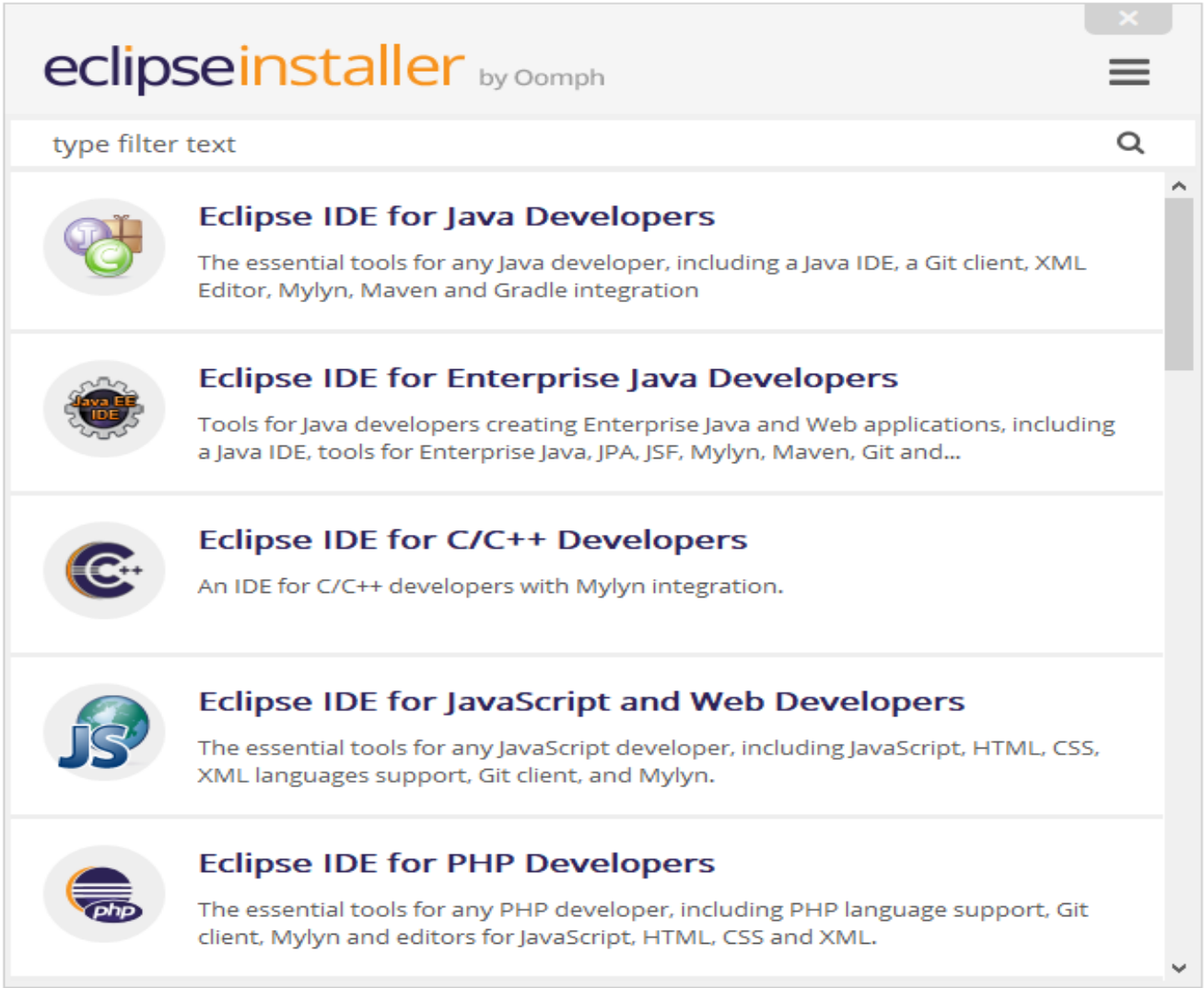
https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2025-03/R/eclipse-inst-jre-win64.exe

Java

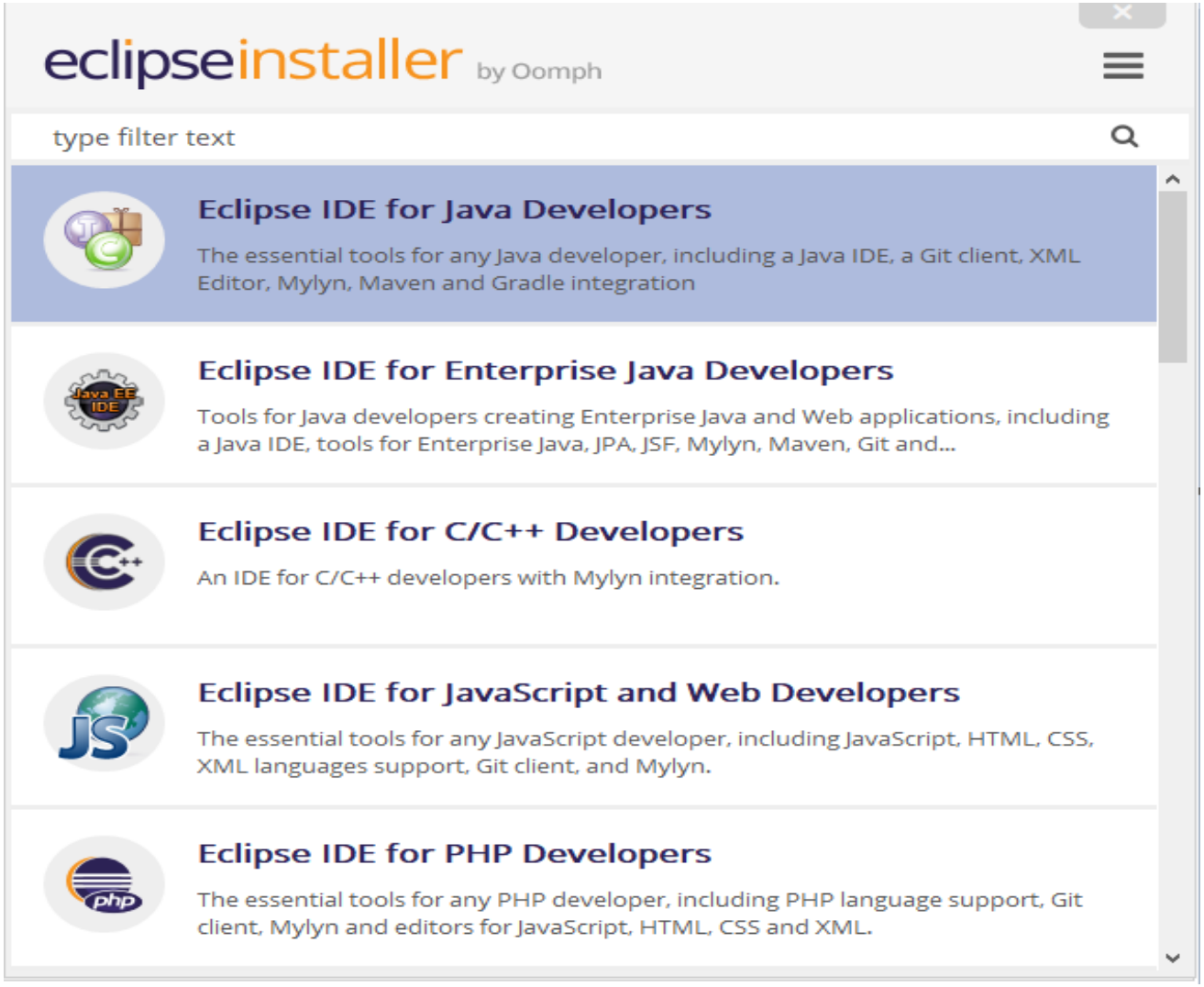
Editeur **Java** en ligne

`https://www.jdoodle.com/online-java-compiler/`

Java


eclipseinstaller by Oomph
type filter text
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration
Eclipse IDE for Enterprise Java Developers
Tools for Java developers creating Enterprise Java and Web applications, including a Java IDE, tools for Enterprise Java, JPA, JSF, Mylyn, Maven, Git and...
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn integration.
Eclipse IDE for JavaScript and Web Developers
The essential tools for any JavaScript developer, including JavaScript, HTML, CSS, XML languages support, Git client, and Mylyn.
Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

Java


eclipseinstaller by Oomph
type filter text
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration
Eclipse IDE for Enterprise Java Developers
Tools for Java developers creating Enterprise Java and Web applications, including a Java IDE, tools for Enterprise Java, JPA, JSF, Mylyn, Maven, Git and...
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn integration.
Eclipse IDE for JavaScript and Web Developers
The essential tools for any JavaScript developer, including JavaScript, HTML, CSS, XML languages support, Git client, and Mylyn.
Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

Quelques raccourcis **Eclipse**

- `Ctrl` + `Shift` + `:` : commenter/décommenter le code
- `Ctrl` + `Shift` + `f` : formater le code
- `Ctrl` + `Alt` + `↓` ou `Ctrl` + `Alt` + `↑` : dupliquer la ligne sélectionnée
- `Ctrl` + `Shift` + `o` : gérer les importer
- `Ctrl` + `Alt` + `l` : afficher la liste des raccourcis
- `Alt` + `Shift` + `R` : faire une sélection multiple
- `Shift` + `K` : aller à l'occurrence suivante
- `Ctrl` + `Shift` + `K` : aller à l'occurrence précédente
- `Ctrl` + `Alt` + `↓` : dupliquer une ligne

Java

Règles de nommage en **Java**

- Pour les classes et les fichiers : **Pascal case**
- Pour les variables, les objets et les méthodes : **Camel case**
- Pour les constantes : **Screaming snake case**
- Pour les noms de projets : **Kebab case**

Java

Règles de nommage en **Java**

- Pour les classes et les fichiers : **Pascal case**
- Pour les variables, les objets et les méthodes : **Camel case**
- Pour les constantes : **Screaming snake case**
- Pour les noms de projets : **Kebab case**

Pour plus de détails

`https://en.wikipedia.org/wiki/Naming_convention_(programming)`

Java

Instructions

- Chaque instruction se termine par `;`
- Il est possible d'écrire plusieurs instructions sur une même ligne (**mais** ce n'est pas une bonne pratique)

Java

Comment organiser un projet **Java** ?

- Une classe par fichier
- Organiser les classes par package selon la sémantique
- Une classe ne peut être définie dans plusieurs fichiers (pas de classe partielle en **Java**)
- Il est possible de créer deux classes avec le même nom dans deux packages différents

Java

Comment créer un projet sous **Eclipse** ?

- Aller dans `File > New > Java Project`
- Remplir le champ `Project name:` avec `cours-introduction` puis cliquer sur `Next`
- Décocher la case `Create module-info.java file` puis cliquer sur `Finish`

Java

Comment créer un projet sous **Eclipse** ?

- Aller dans `File > New > Java Project`
- Remplir le champ `Project name:` avec `cours-introduction` puis cliquer sur `Next`
- Décocher la case `Create module-info.java file` puis cliquer sur `Finish`

Que contient ce projet ?

- `JRE System Library` : l'ensemble de `.jar` indispensable pour le lancement du projet
- `src` : le répertoire qui contiendra les fichiers sources (les classes)

Java

Comment créer une classe ?

- Aller dans `File > New > Class`
- Dans `Package`, saisir `org.eclipse.classes`
- Dans `Class`, saisir `FirstClass`
- Cocher la case `public static void main (String[] args)`
- Cliquer sur `Finish`

Java

Comment créer une classe ?

- Aller dans `File > New > Class`
- Dans `Package`, saisir `org.eclipse.classes`
- Dans `Class`, saisir `FirstClass`
- Cocher la case `public static void main (String[] args)`
- Cliquer sur `Finish`

Remarque

Si on a un package, on peut le sélectionner au moment de la création de la classe

Java

Comment créer un package ?

- Aller dans `File > New > Package`
- Saisir le nom du package et valider

Java

Contenu de la classe `FirstClass`

```
package org.eclipse.classes;

public class FirstClass {

  public static void main(String[] args) {
    // TODO Auto-generated method stub

  }

}
```

Java

Explication

- En **Java**, et contrairement à plusieurs langages OO comme **C++**, **Python**, **PHP**..., on ne peut écrire une instruction en dehors d'une (méthode de) classe.
- En **Java**, un fichier contient une seule classe et une classe ne peut être déclarée dans plusieurs fichiers (contrairement à **C#**)
- La première ligne `package org.eclipse.classes` nous informe que la classe actuelle se situe dans `src/org/eclipse/classes` dans un répertoire `cours-introduction` situé dans le **workspace** d'**Eclipse**
- Dans un projet **Java**, il faut qu'au moins une classe contienne la méthode `public static void main(String[] args)` : point d'entrée du projet

Pour afficher `Hello World`, on modifie la classe `FirstClass`

```
package org.eclipse.classes;

public class FirstClass {

    public static void main(String[] args) {

        System.out.println("Hello World!");

    }
}
```


Java est un langage 100% (ou presque) orienté objet

- Pour afficher un message, il faut utiliser la classe `System`
- La classe `System` a deux objets pour l'entrée/sortie (`in`/`out`)
- L'objet `out` a plusieurs méthodes d'affichage comme `print()` et `println()`

Java

Comment exécuter le programme ? (voir le résultat)

- Soit en faisant clic droit sur `cours-introduction` dans `Package Explorer` et aller dans `Run As > Java Application`
- Soit en faisant clic droit sur la classe contenant `puclic static void main()` (ici `FirstClass`) dans le panneau central et aller dans `Run As > Java Application`
- Soit en cliquant sur le triangle vert dans la liste de raccourci

Java

Où voir le résultat ?

- Dans la console **Eclipse**
- Si la console n'est pas visible, aller dans `Window > Show View > Other...`, saisir `console` et la sélectionner puis valider.

Java

Où voir le résultat ?

- Dans la console **Eclipse**
- Si la console n'est pas visible, aller dans `Window > Show View > Other...`, saisir `console` et la sélectionner puis valider.

Où sont les `.class` générés ?

- Dans le `work-space`, aller voir dans le répertoire portant le nom du projet (ici `cours-introduction`)
- Dans `org/eclipse/classes`, un fichier `FirstClass.class` a été généré.

Java

Commentaires

Instructions ignorées par le compilateur

Java

Commentaires

Instructions ignorées par le compilateur

Trois types de commentaire en **Java**

Java

Commentaire sur une seule ligne

```
// commentaire
```

Java

Commentaire sur une seule ligne

```
// commentaire
```


Raccourci **Eclipse**

Pour commenter ou décommenter : `Ctrl` + `Shift` + `:` ou `Ctrl` + `Shift` + `c`

Java

Commentaire sur une plusieurs lignes (Raccourci VS :)

```
/* le commentaire
   la suite
   et encore la suite
*/
```

Java

Commentaire sur une plusieurs lignes (Raccourci VS :)

```
/* le commentaire
   la suite
   et encore la suite
*/
```


Raccourci **Eclipse**

Pour commenter ou dé-commenter : Ctrl + \

Java

Commentaire pour la documentation

```
/**
 * @author Achref El Mouelhi
 */
```

Java

Commentaire pour la documentation

```
/**
 * @author Achref El Mouelhi
 */
```


Raccourci **Eclipse**

Alt + Shift + j

Java

Documentation

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html