

Aujourd'hui:
Syntaxe de base du langage Java,
la sémantique et le concept de la compilation.

- Prise en main de l'outil de développement Eclipse.
- Votre premier programme en Java.

Prise en main de l'outil de développement Eclipse (1/11)

Installation des outils nécessaires: télécharger l'environnement Java sur le site d'Oracle, - **JRE ou JDK** -












Java Platform, Standard Edition		
Java SE 7u4 This release includes the first Oracle JDK release for Mac OS X. JavaFX 2.1 is now bundled with the JDK on Windows and Mac. We also include bug fixes and performance improvements. Learn more  "What Java Do I Need?" You must have a copy of the JRE (Java Runtime Environment) on your system to run Java applications and applets. To develop Java applications and applets, you need the JDK (Java Development Kit), which includes the JRE.	JDK DOWNLOAD  JDK 7 Docs <ul style="list-style-type: none">▪ Installation Instructions▪ ReadMe▪ ReleaseNotes▪ Oracle License▪ Java SE Products▪ Third Party Licenses▪ Certified System Configurations	JRE DOWNLOAD  JRE 7 Docs <ul style="list-style-type: none">▪ Installation Instructions▪ ReadMe▪ ReleaseNotes▪ Oracle License▪ Java SE Products▪ Third Party Licenses▪ Certified System Configurations

Prise en main de l'outil de développement Eclipse (2/11)

Java SE Runtime Environment 7u5

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☒ Accept License Agreement ☐ Decline License Agreement

Product / File Description	File Size	Download
Linux x86	20.46 MB	 jre-7u5-linux-i586.rpm
Linux x86	32.78 MB	 jre-7u5-linux-i586.tar.gz
Linux x64	20.74 MB	 jre-7u5-linux-x64.rpm
Linux x64	31.35 MB	 jre-7u5-linux-x64.tar.gz
Solaris x86	35.8 MB	 jre-7u5-solaris-i586.tar.gz
Solaris SPARC	40.16 MB	 jre-7u5-solaris-sparc.tar.gz
Solaris SPARC 64-bit	12.38 MB	 jre-7u5-solaris-sparcv9.tar.gz
Solaris x64	9.38 MB	 jre-7u5-solaris-x64.tar.gz
Windows x86 Online	0.85 MB	 jre-7u5-windows-i586-iftw.exe
Windows x86 Offline	20.08 MB	 jre-7u5-windows-i586.exe
Windows x64	20.86 MB	 jre-7u5-windows-x64.exe

Prise en main de l'outil de développement Eclipse (3/11)

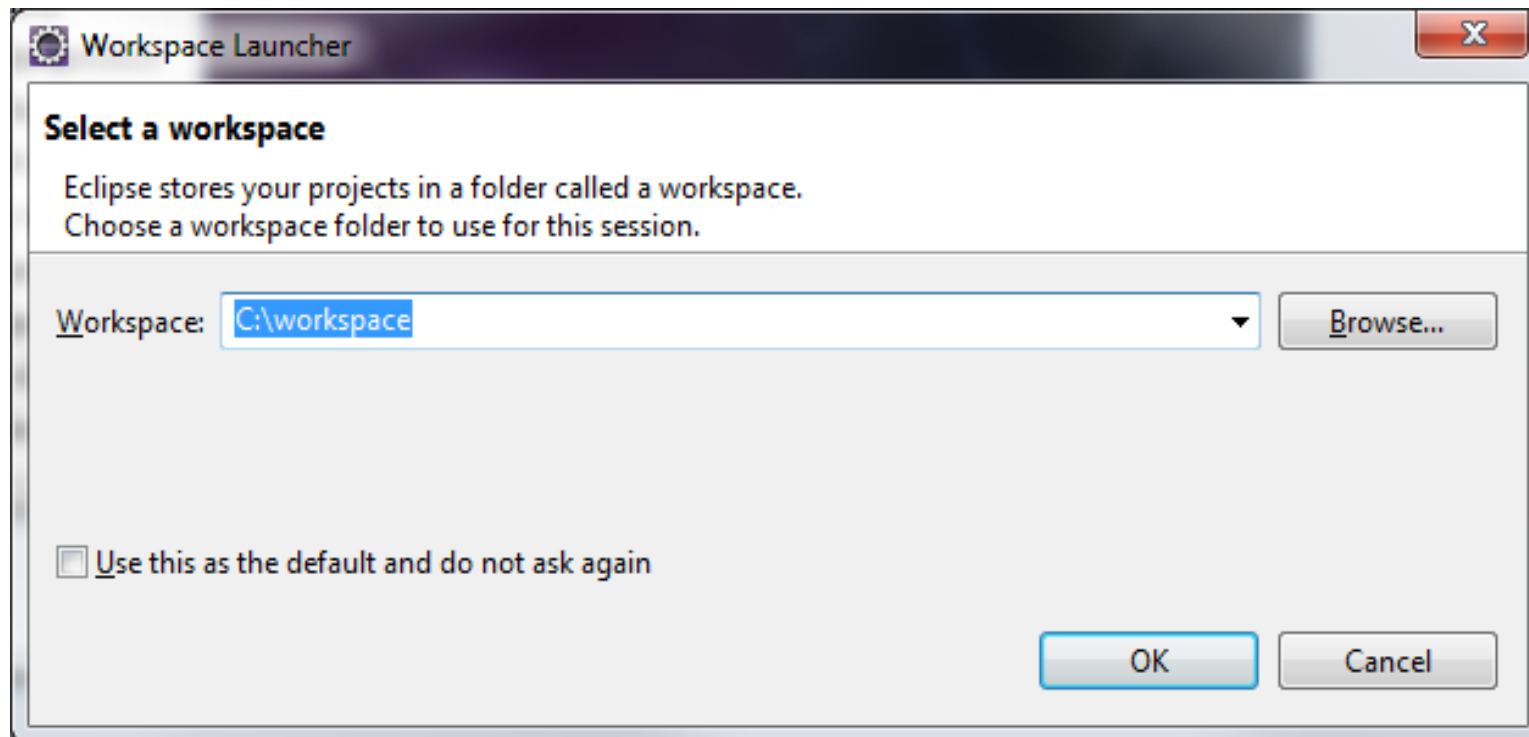
Pourquoi le IDE Eclipse:

- La spécificité d'Eclipse IDE vient du fait que son architecture est totalement développée autour de la notion de plugin: toutes ses fonctionnalités sont développées en tant que plugins.
- Pour faire court, si vous voulez ajouter des fonctionnalités à Eclipse, vous devez :
 - télécharger le plugin correspondant ;
 - copier les fichiers spécifiés dans les répertoires spécifiés ;
 - démarrer Eclipse, et ça y est !

N.B: Eclipse IDE est lui-même principalement écrit en Java.

Prise en main de l'outil de développement Eclipse (4/11)

Au démarrage d'Eclipse: choix du dossier d'enregistrement de projets



N.B: rien ne
vous

empêche de spécifier un autre dossier que celui proposé par défaut.

Prise en main de l'outil de développement Eclipse (5/11)

Présentation rapide de l'interface d'Eclipse

File: C'est ici que nous pourrons créer de nouveaux projets Java, les enregistrer et les exporter le cas échéant.

Les raccourcis à retenir sont :

- **ALT + SHIFT + N** : nouveau projet ;
- **CTRL + S** : enregistrer le fichier où l'on est positionné ;
- **CTRL + SHIFT + S** : tout sauvegarder ;
- **CTRL + W** : fermer le fichier où l'on est positionné ;
- **CTRL + SHIFT + W** : fermer tous les fichiers ouverts.

Prise en main de l'outil: de développement Eclipse (6/11)

La barre d'outils



1 nouveau général : cliquer sur ce bouton revient à faire Fichier > Nouveau ;

2 enregistrer : revient à faire **CTRL + S** ;

3 imprimer .

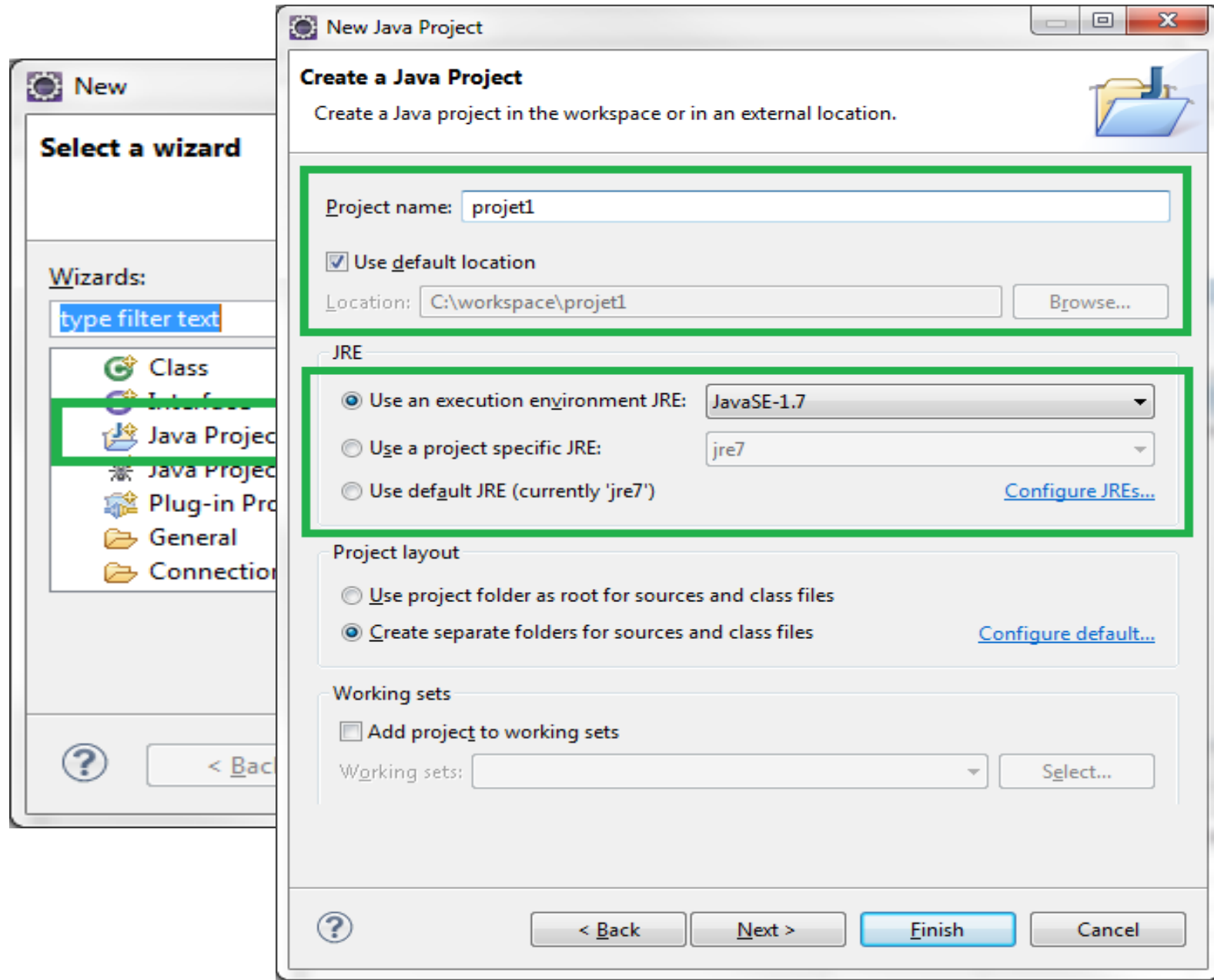
4 exécuter la classe ou le projet spécifié : nous verrons ceci plus en détail ;

5 créer un nouveau projet : revient à faire Fichier > Nouveau > Java Project;

6 créer une nouvelle classe : créer un nouveau fichier. Cela revient à faire Fichier > Nouveau > Classe.

Prise en main de l'outil: de développement Eclipse (7/11)

Création d'un nouveau projet Java



Prise en main de l'outil: de développement Eclipse (8/11)

Création de projet Java - Étape 2 -

New Java Class

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

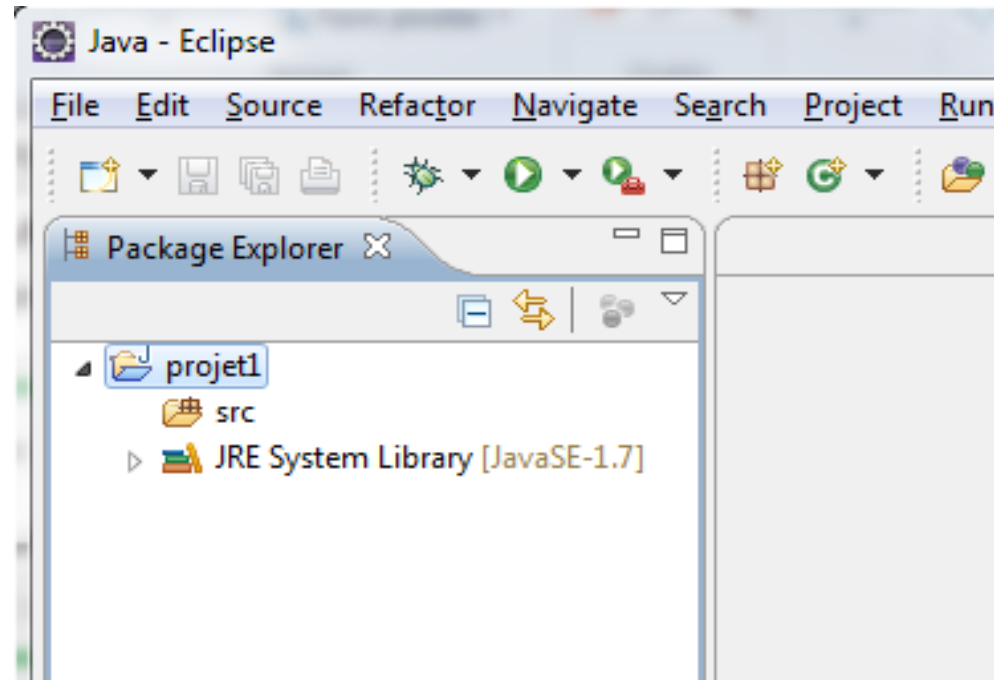
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

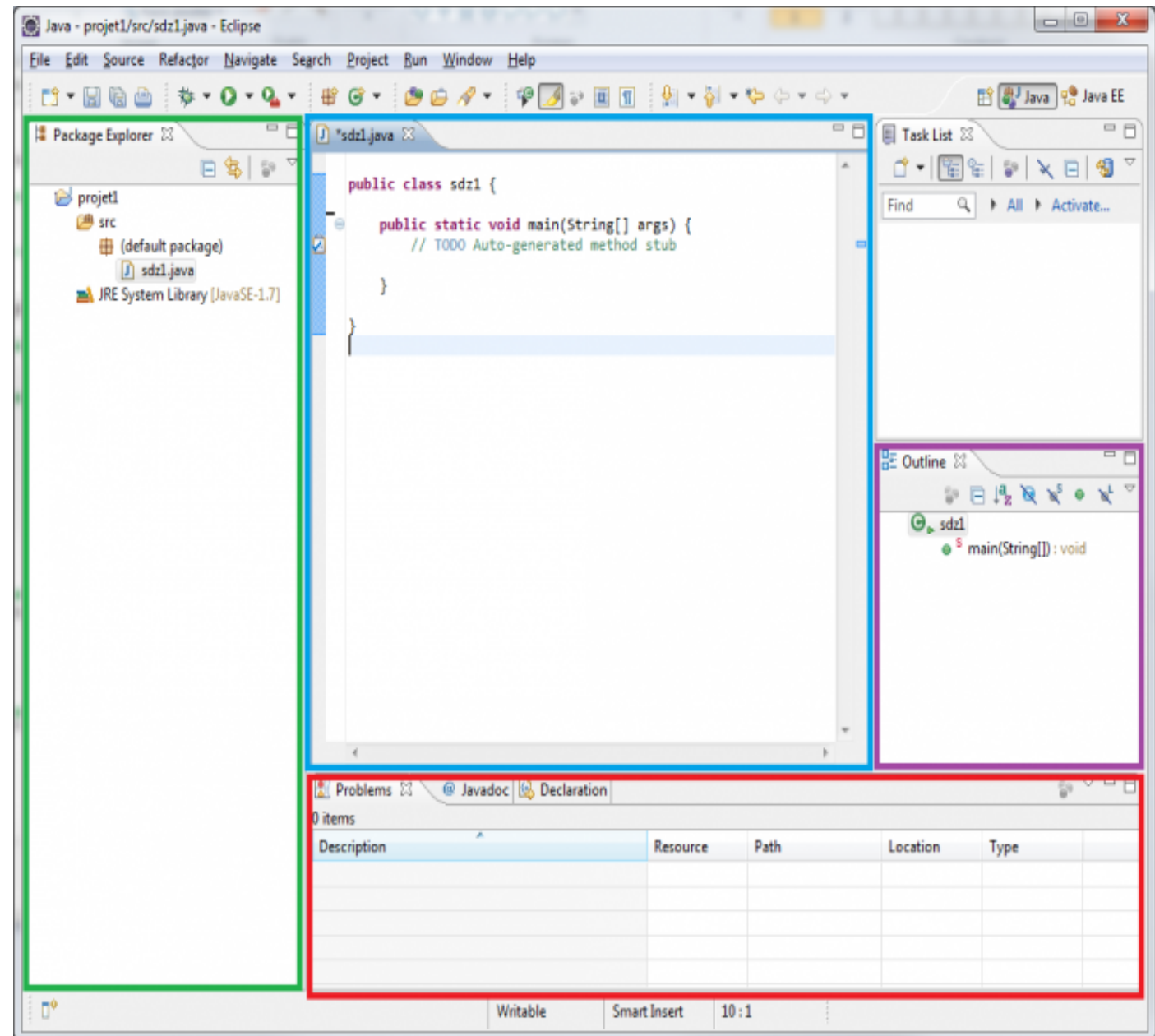
Prise en main de l'outil de développement Eclipse (9/11)

La fenêtre d'un nouveau projet: Explorateur de projet.



Prise en main de l'outil de développement Eclipse (10/11)

Mon premier programme
composé d'une classe.



Prise en main de l'outil Eclipse (11/11)

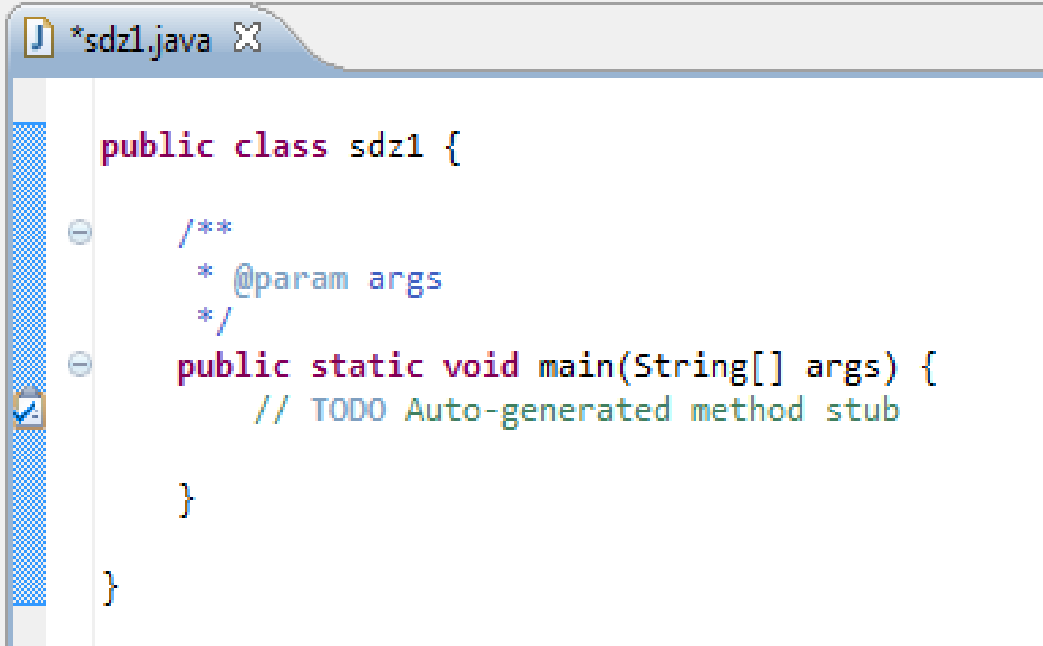
Mon premier programme
composé d'une classe.

Tous les programmes Java sont composés d'au moins une classe. Elle doit contenir une méthode appelée `main` : ce sera le point de démarrage de notre programme.

Votre premier programme en Java (1/7)

Ce que il faut retenir pour le moment:

- La classe est définie par un mot clé (`class`), qu'elle a un nom (ici, « `sdz1` »)
- son contenu est délimité par des accolades (`{ }`). Nous écrirons nos codes sources entre les accolades de la méthode `main`. La syntaxe de cette méthode est toujours la même :



```
public class sdz1 {  
    /**  
     * @param args  
     */  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
    }  
}
```

Votre premier programme en Java (2/7)

Commenter son programme:

Deux syntaxes sont disponibles pour commenter son texte :

- 1- Les commentaires unilignes : introduits par les symboles « // », ils mettent tout ce qui les suit en commentaire, du moment que le texte se trouve sur la même ligne ;
- 2- Les commentaires multilignes : ils sont introduits par les symboles « /* » et se terminent par les symboles « */ ».

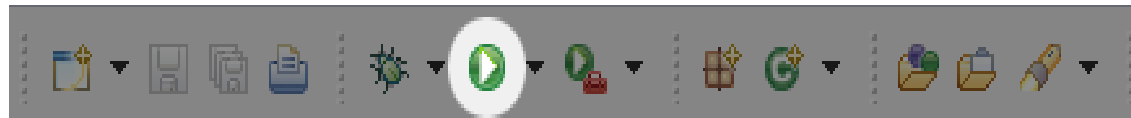
```
public static void main(String[] args) {  
    //Un commentaire  
    //Un autre  
  
    /*  
    Un commentaire  
    Un autre  
    Encore un autre  
    */  
}
```

Votre premier programme en Java (3/7)

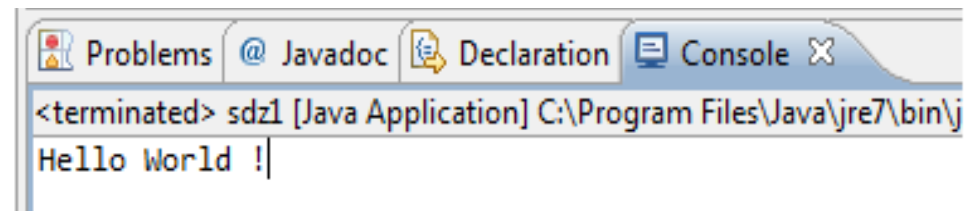
Compilation et exécution de « Hello World »:

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

Pour lancer le programme:



La console d'Eclipse :



Votre premier programme en Java (4/7)

Reprenons le code

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

La méthode `print()` va écrire « Hello World ! » en utilisant l'objet `out` de la classe `System` » .

`System` : ceci correspond à l'appel d'une classe qui se nomme « System » . C'est une classe utilitaire qui permet surtout d'utiliser l'entrée et la sortie standard, c'est-à-dire la saisie clavier et l'affichage à l'écran.

`out` : objet de la classe `System` qui gère la sortie standard.

`print` : méthode qui écrit dans la console le texte passé en paramètre (entre les parenthèses).

Application: Afficher le message: « Bonjour je m 'appelle XXXXXX ».

Votre premier programme en Java (5/7)

Retour à la ligne

- soit utiliser un caractère d'échappement, ici `\n`;
- soit utiliser la méthode `println()` à la place de la méthode `print()`.

Exemple d'application

```
public static void main(String[] args) {  
    System.out.print("Hello le monde \n");  
    System.out.println("je m'appelle");  
    System.out.print("\n XXXXXXXXXXXX");  
    //Un commentaire  
    /*  
    Un commentaire  
    Un autre  
    */  
}
```

Votre premier programme en Java (6/7)

Pour la documentation du code

Project ->Generate Javadoc

- Public cochée (par défaut): seuls les éléments en visibilité **public** seront exportés.
- La visibilité **Private** permet de générer a documentation de tous es éléments.

Votre premier programme en Java (7/7)

Et en cas d'erreur?

- supposons que nous avons oublié le ';' dans notre logne de code:

```
System.out.print("Hello le monde")
```

- analyse des messages d'erreurs: fenêtres Console et Problems