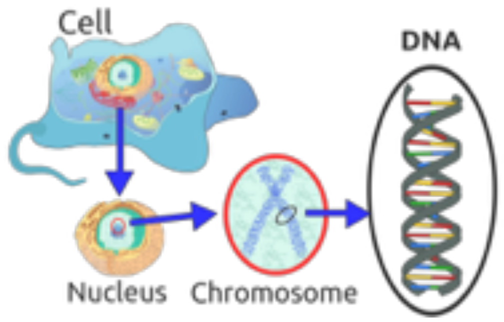
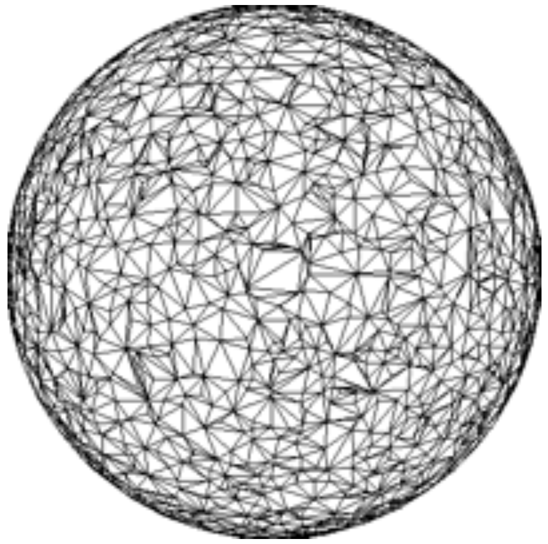
CSE 201
Java Programming I

Smart Coding School
website: http://www.smartcodingschool.com

Computer Science



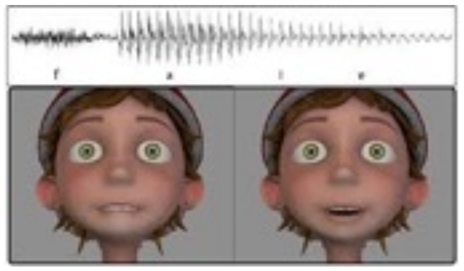
Computational Biology



Computational Geometry

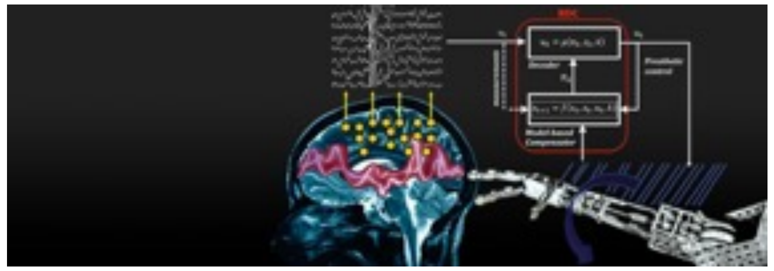
Computational Finance

Computational Geography



Computational Animation

Game Design



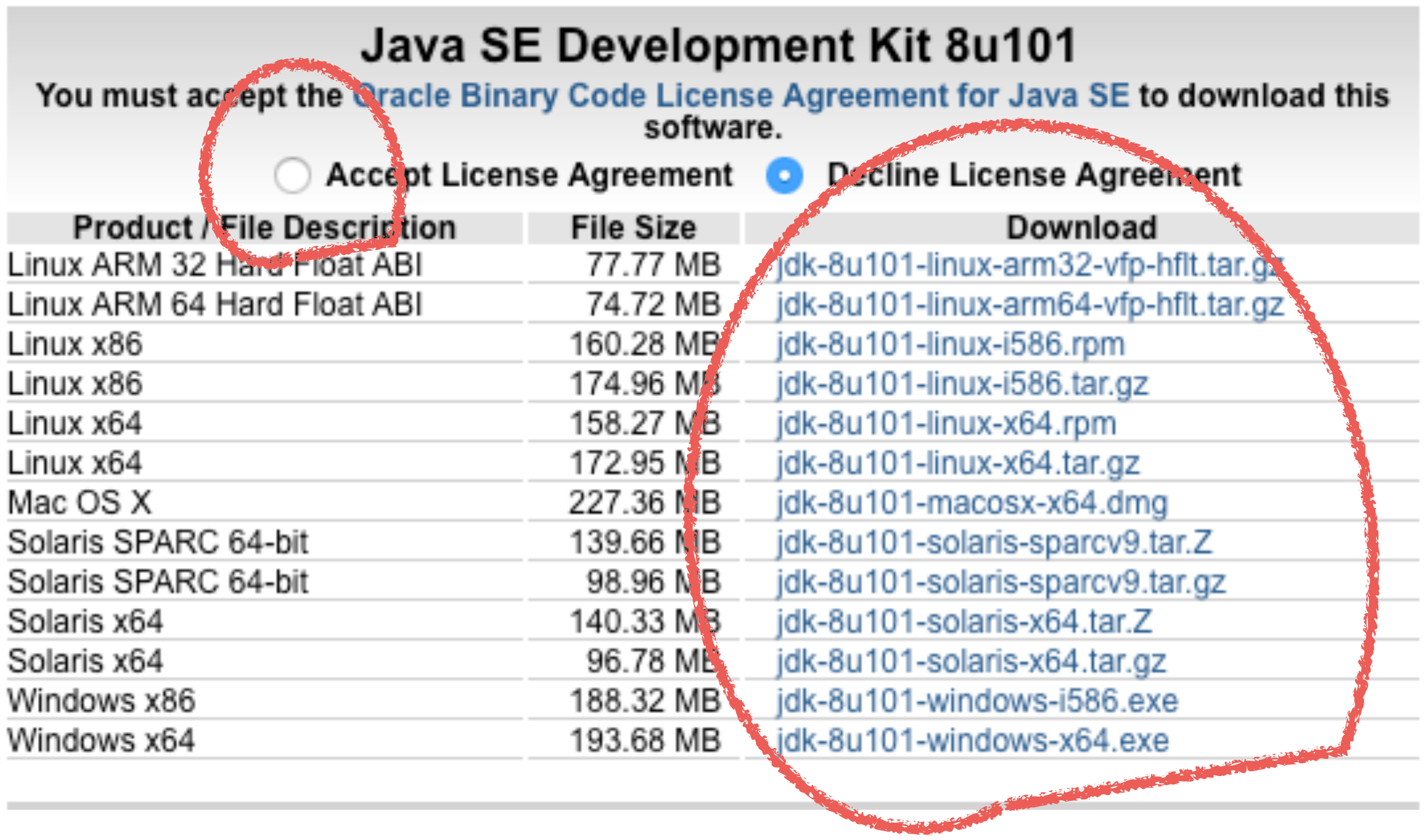
Computational Medicine

Java Programming Language

1. Object-oriented programming language
2. Concurrency
3. Cross-platform: Write once Run anywhere

Java 8 SDK

Download Link: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html



Java SE Development Kit 8u101

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

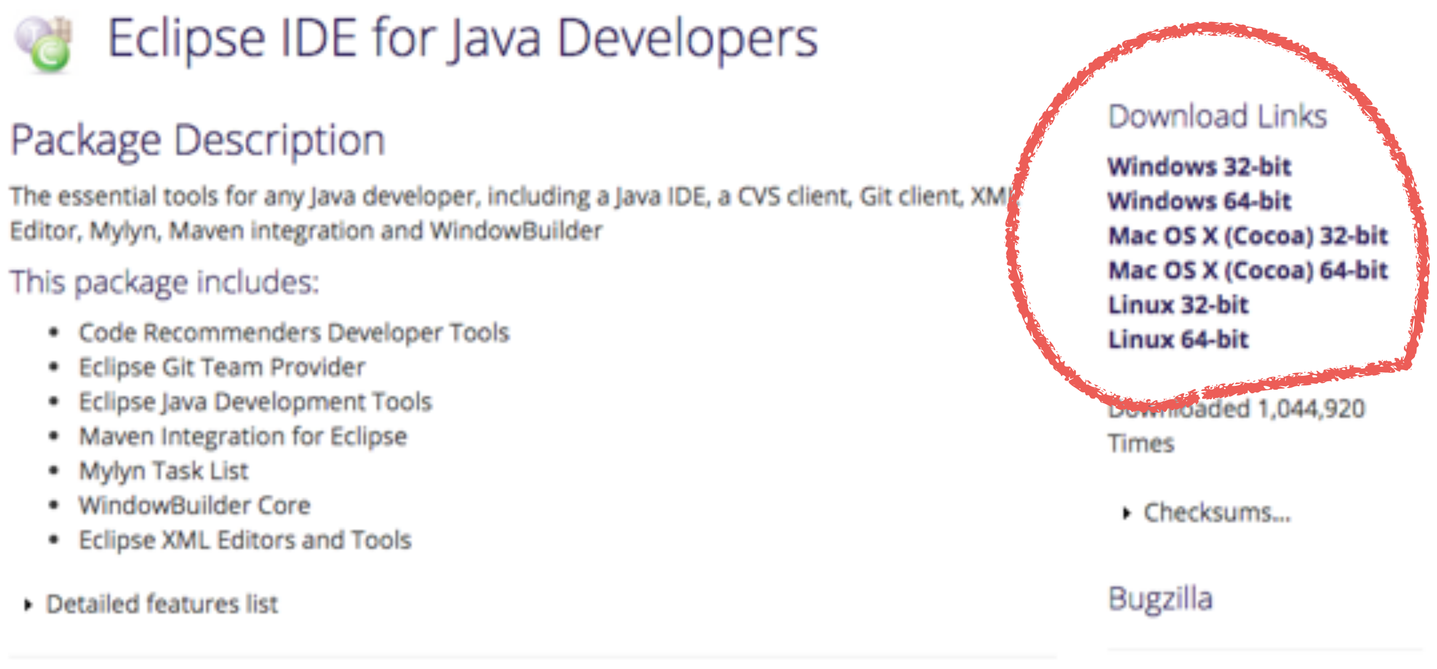
Accept License Agreement [✓ Decline License Agreement]

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.77 MB	jdk-8u101-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.72 MB	jdk-8u101-linux-arm64-vfp-hflt.tar.gz
Linux x86	160.28 MB	jdk-8u101-linux-i586.rpm
Linux x86	174.96 MB	jdk-8u101-linux-i586.tar.gz
Linux x64	158.27 MB	jdk-8u101-linux-x64.rpm
Linux x64	172.95 MB	jdk-8u101-linux-x64.tar.gz
Mac OS X	227.36 MB	jdk-8u101-macosx-x64.dmg
Solaris SPARC 64-bit	139.66 MB	jdk-8u101-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	98.96 MB	jdk-8u101-solaris-sparcv9.tar.gz
Solaris x64	140.33 MB	jdk-8u101-solaris-x64.tar.Z
Solaris x64	96.78 MB	jdk-8u101-solaris-x64.tar.gz
Windows x86	188.32 MB	jdk-8u101-windows-i586.exe
Windows x64	193.68 MB	jdk-8u101-windows-x64.exe

Accept License Agreement first then download

Java IDE
(integrated development environment)

Download Link: (Mars)
http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/mars2


Eclipse IDE for Java Developers

Package Description

The essential tools for any Java developer, including a Java IDE, a CVS client, Git client, XML Editor, Mylyn, Maven integration and WindowBuilder

This package includes:

- Code Recommenders Developer Tools
- Eclipse Git Team Provider
- Eclipse Java Development Tools
- Maven Integration for Eclipse
- Mylyn Task List
- WindowBuilder Core
- Eclipse XML Editors and Tools

▸ Detailed features list

Download Links

Windows 32-bit
Windows 64-bit
Mac OS X (Cocoa) 32-bit
Mac OS X (Cocoa) 64-bit
Linux 32-bit
Linux 64-bit

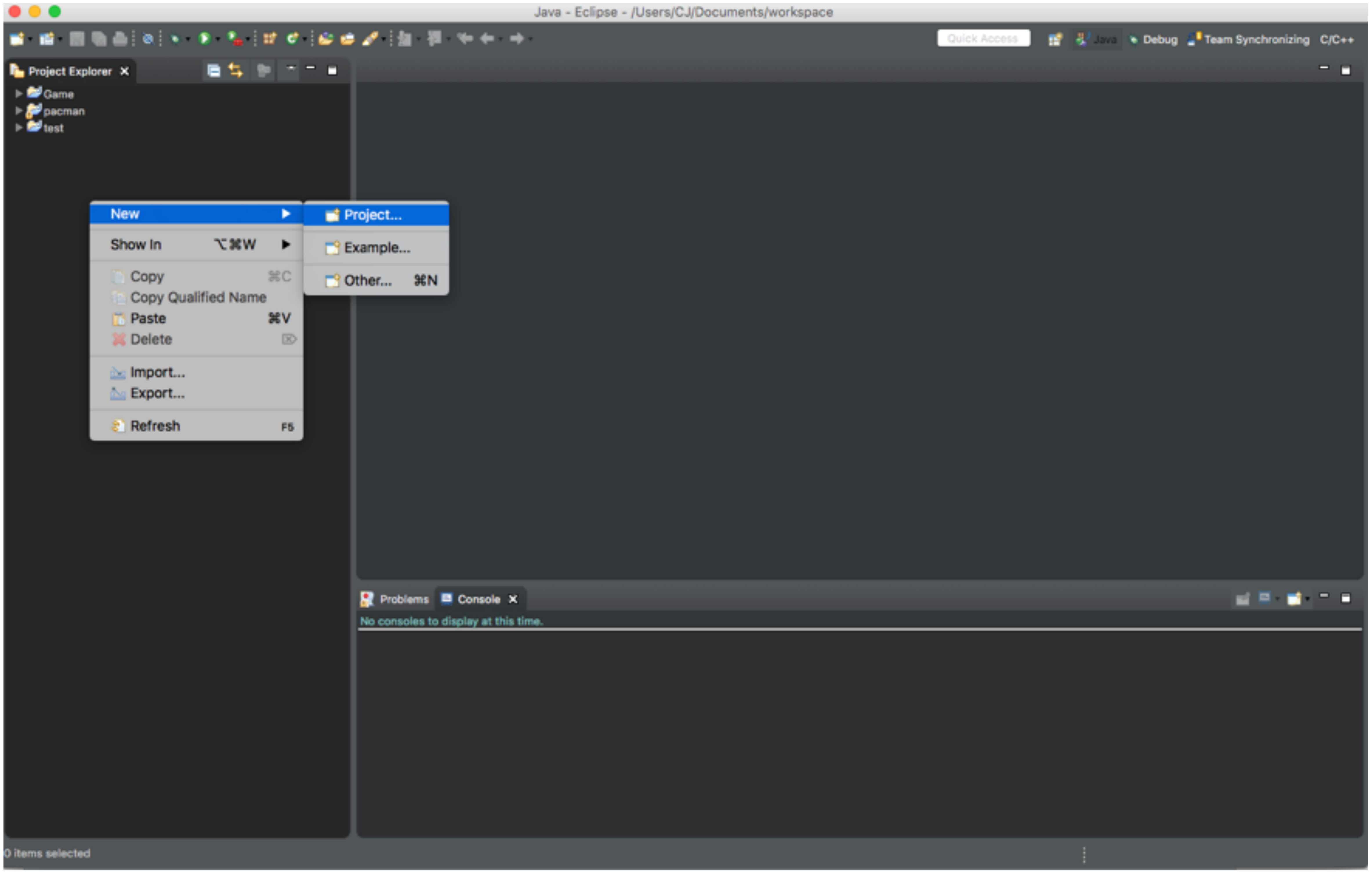
Downloaded 1,044,920 Times

▸ Checksums...

Bugzilla

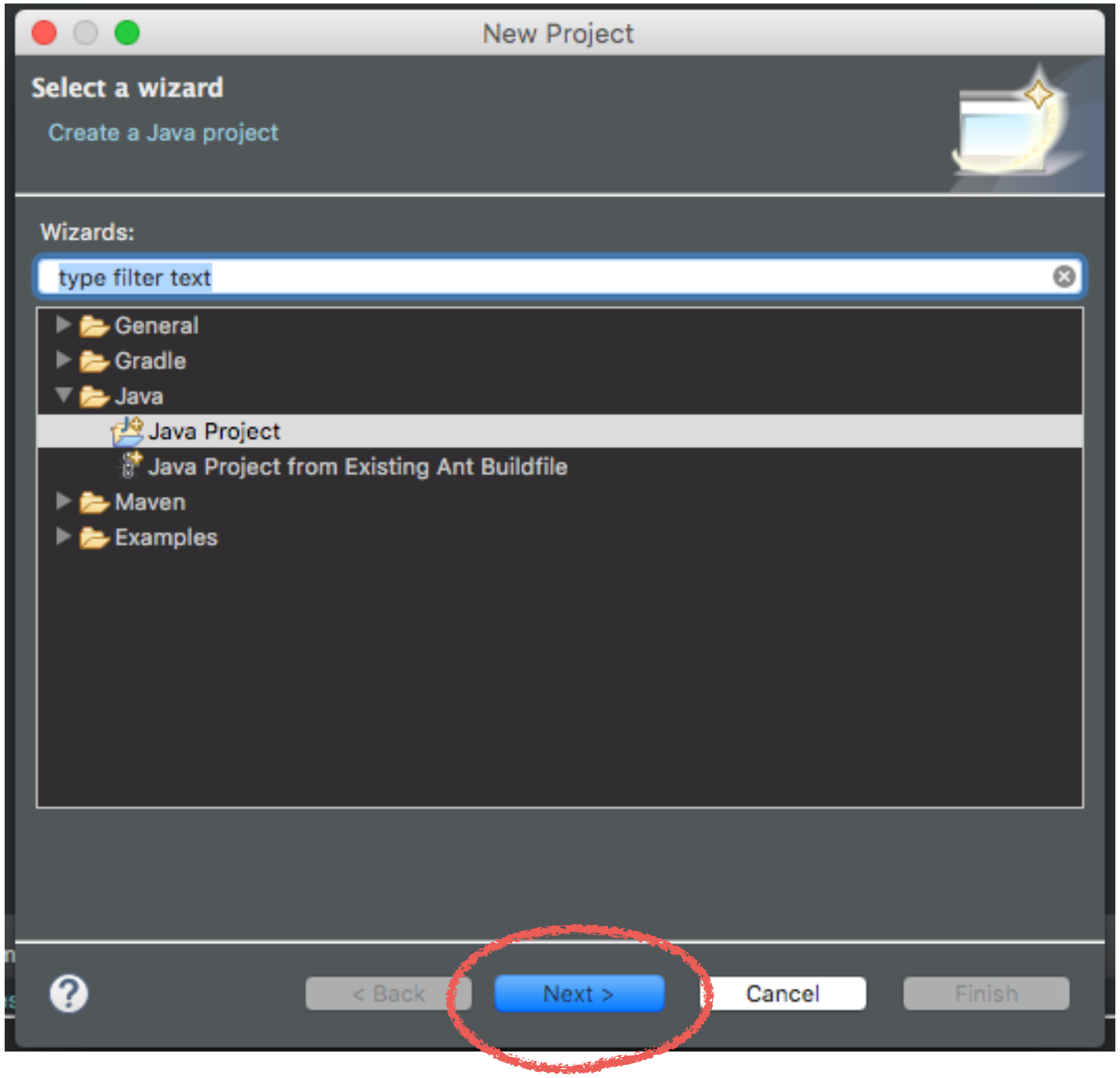
Project setup

Right click the project explorer, select new -> project



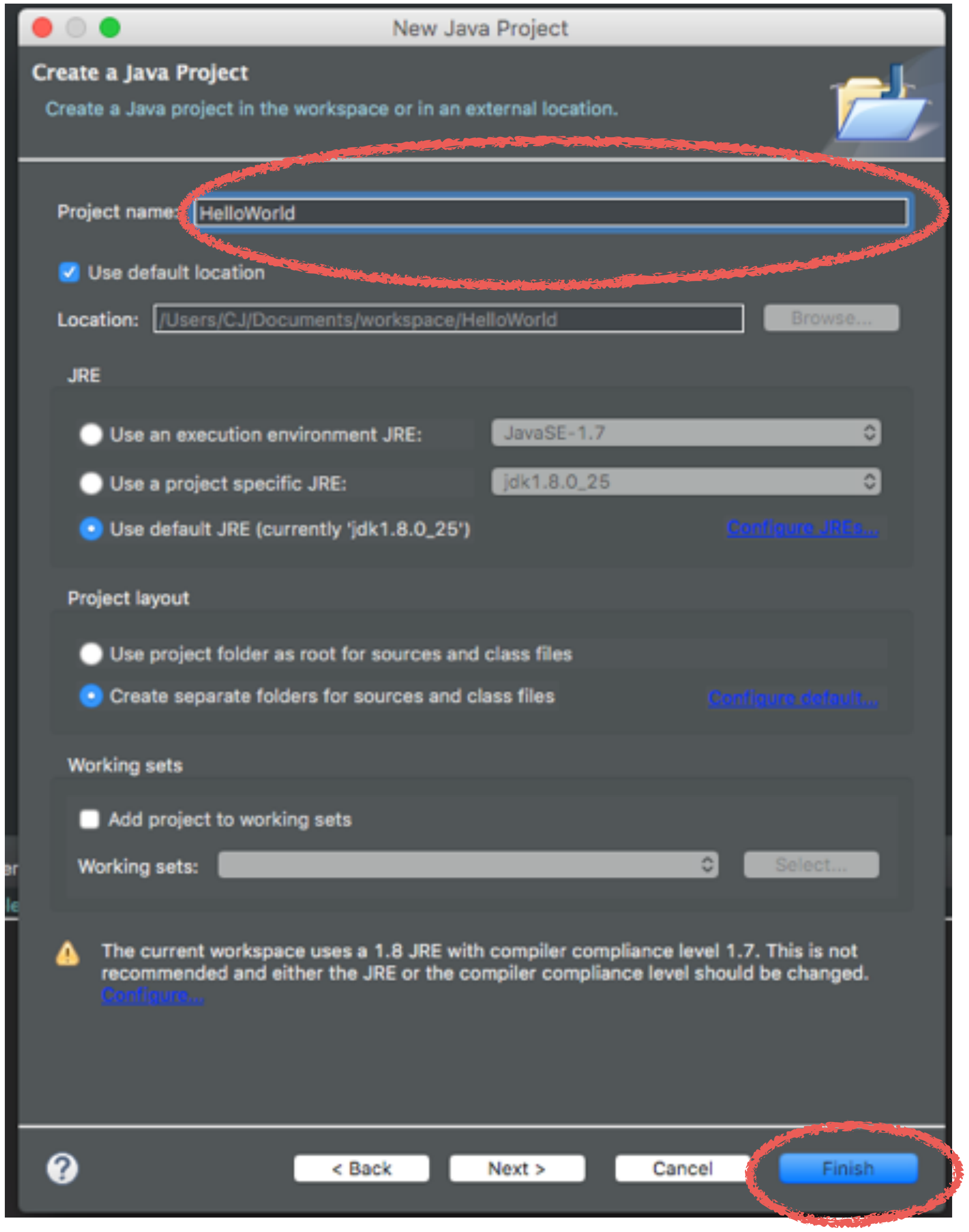
Project setup

click Next>



Project setup

Enter the project name and click Finish



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: HelloWorld

[✓] Use default location

Location: /Users/CJ/Documents/workspace/HelloWorld Browse...

JRE

- ○ Use an execution environment JRE: JavaSE-1.7
- ○ Use a project specific JRE: jdk1.8.0_25
- ● Use default JRE (currently 'jdk1.8.0_25') Configure JREs...

Project layout

- ○ Use project folder as root for sources and class files
- ● Create separate folders for sources and class files Configure default...

Working sets

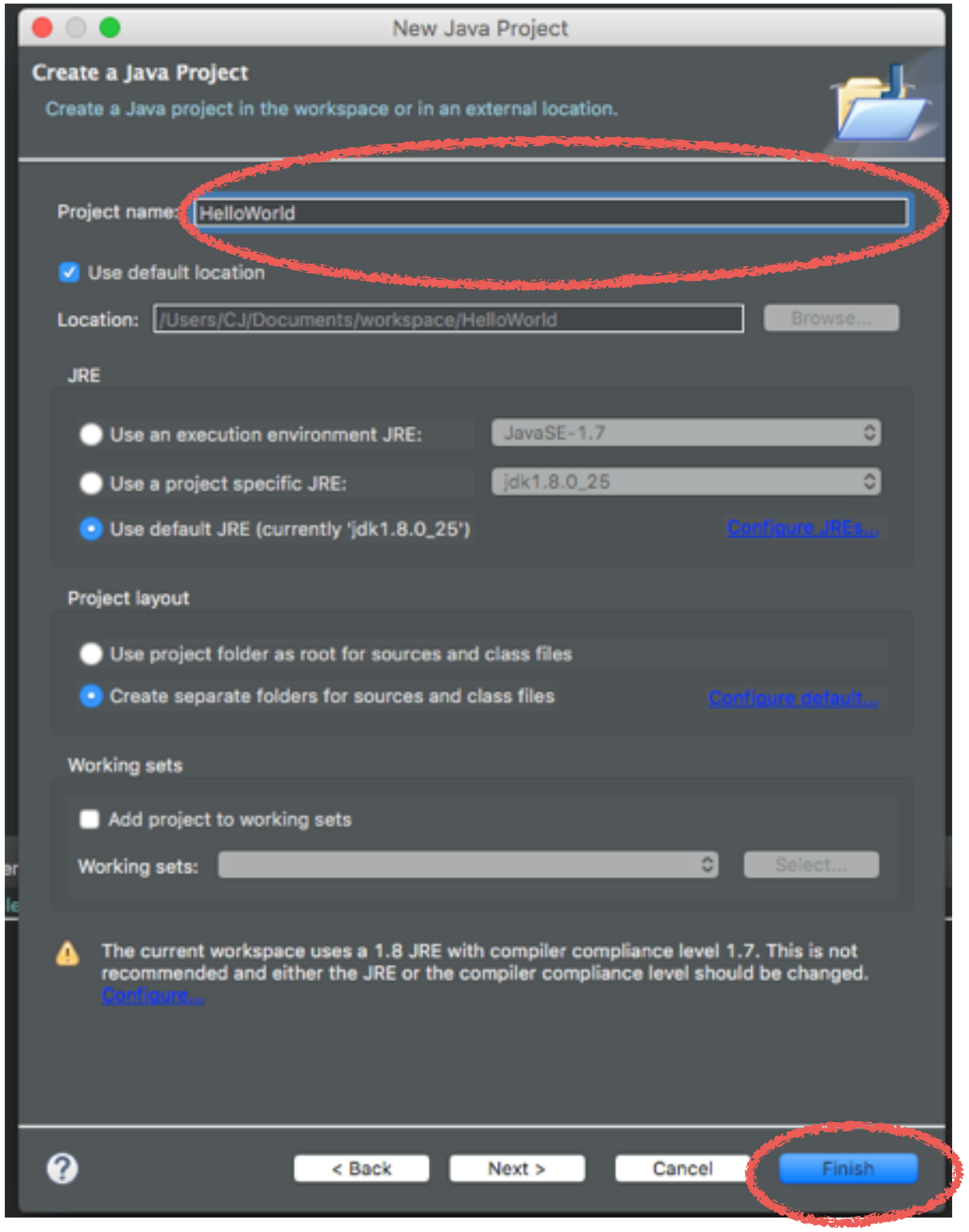
[ ] Add project to working sets

Working sets: Select...

The current workspace uses a 1.8 JRE with compiler compliance level 1.7. This is not recommended and either the JRE or the compiler compliance level should be changed. Configure...

Project setup

Enter the project name and click Finish


New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: HelloWorld

[✓] Use default location

Location: /Users/CJ/Documents/workspace/HelloWorld

JRE

- Use an execution environment JRE: JavaSE-1.7
- Use a project specific JRE: jdk1.8.0_25
- [✓ Use default JRE (currently 'jdk1.8.0_25')]

Project layout

- Use project folder as root for sources and class files
- [✓ Create separate folders for sources and class files]

Working sets

- [ ] Add project to working sets

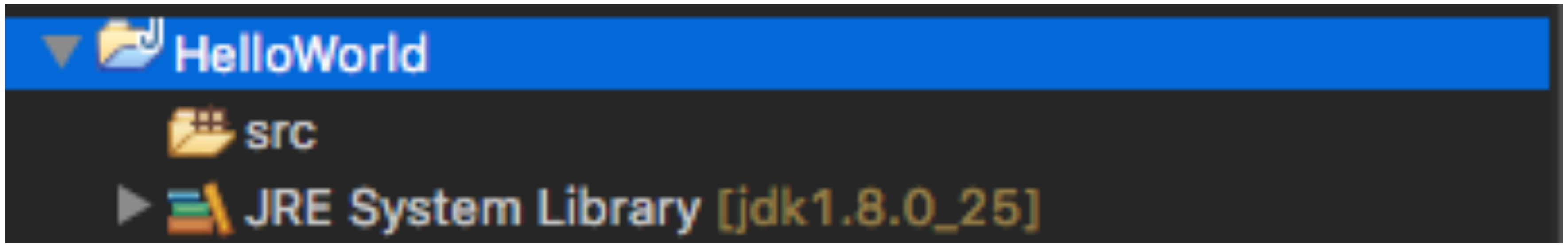
Working sets:

The current workspace uses a 1.8 JRE with compiler compliance level 1.7. This is not recommended and either the JRE or the compiler compliance level should be changed. Configure...

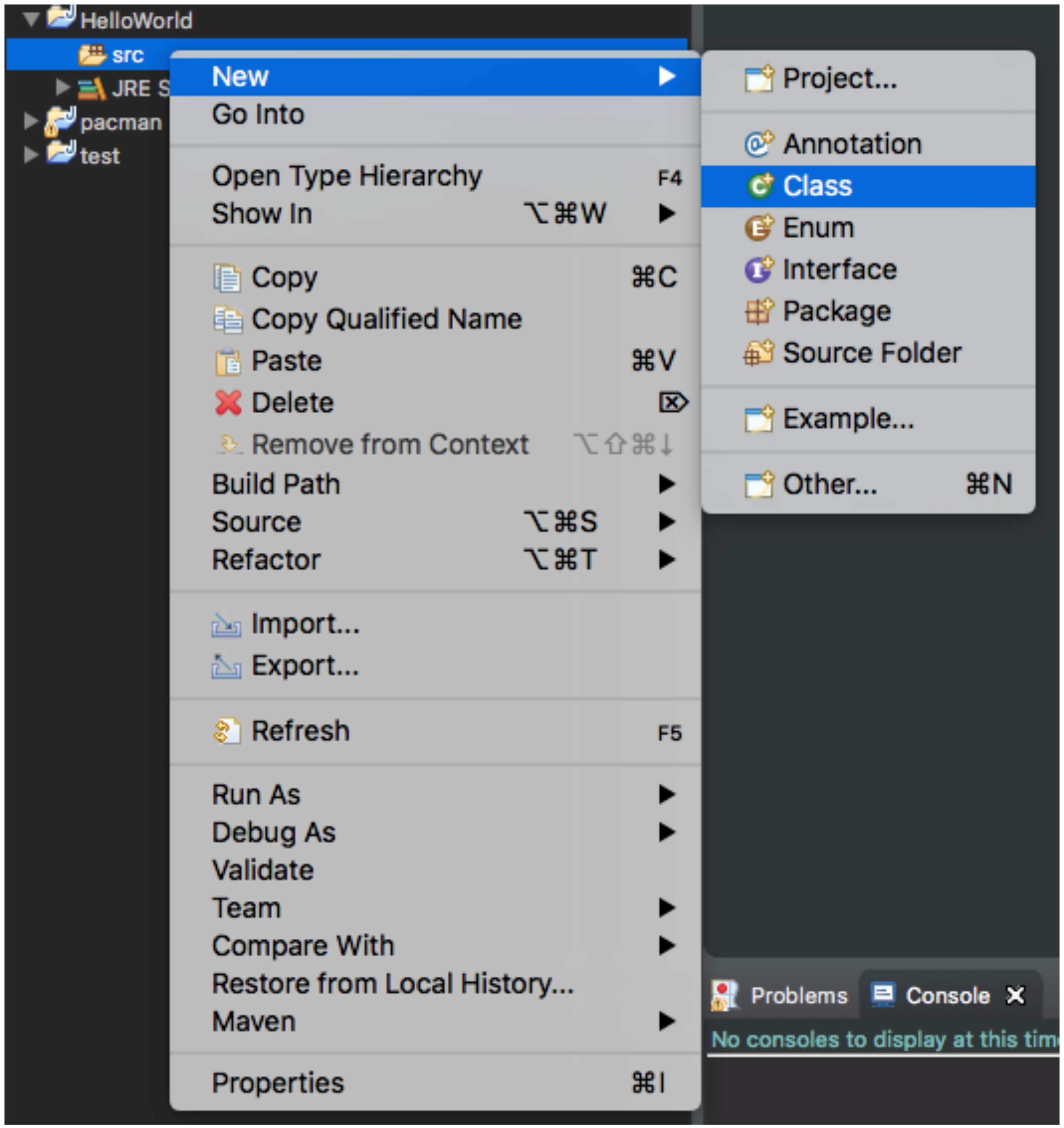
Project setup

Find your project
in the project explorer on the left side

Click the arrow sign to expand the project folder



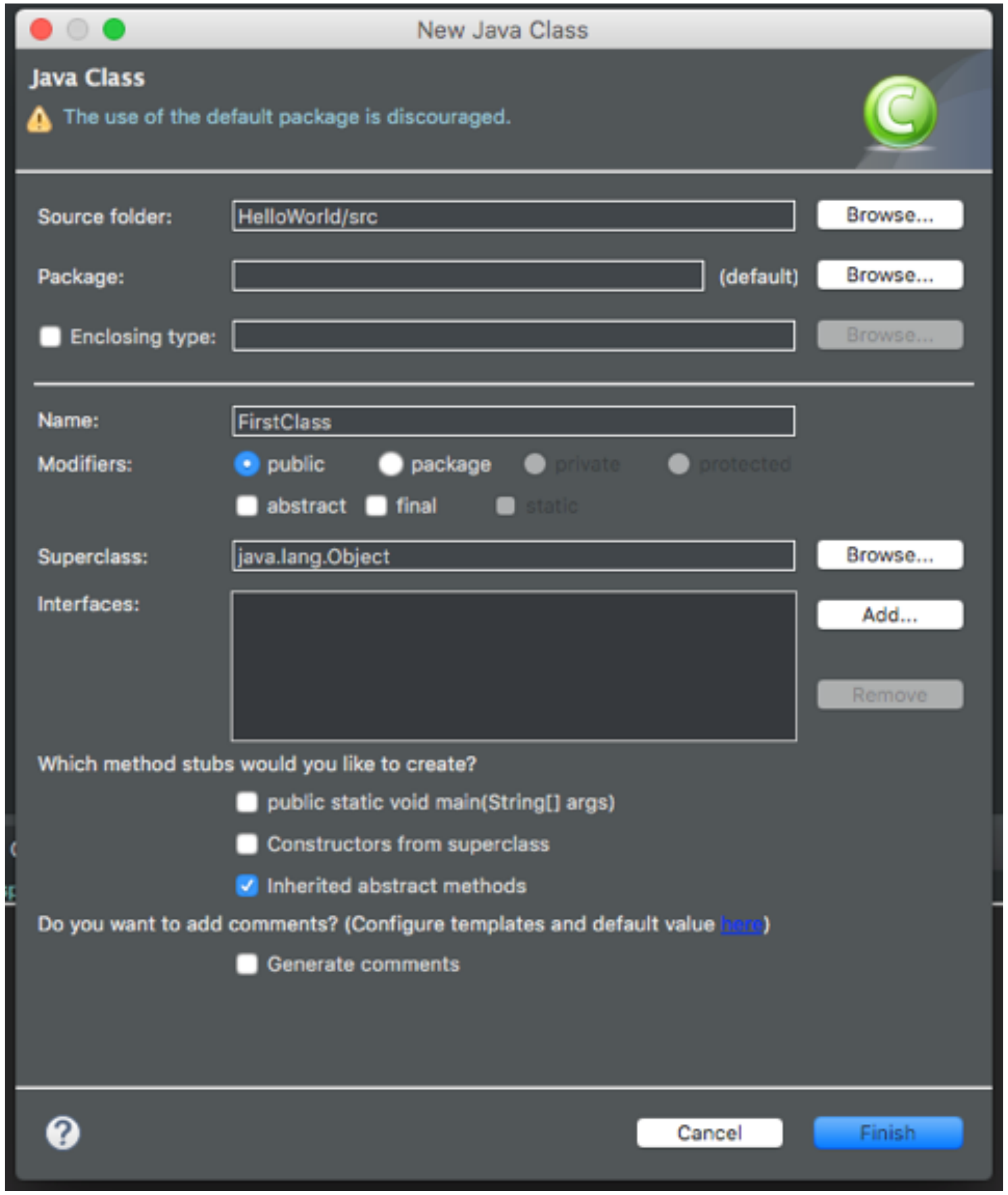
Project setup



Any code goes to src folder

Right click src folder select new -> class

Project setup



⚠ The use of the default package is discouraged.

Source folder: HelloWorld/src

Package: (default)

☐ Enclosing type:

Name: FirstClass

Modifiers: [✓ public] package private protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

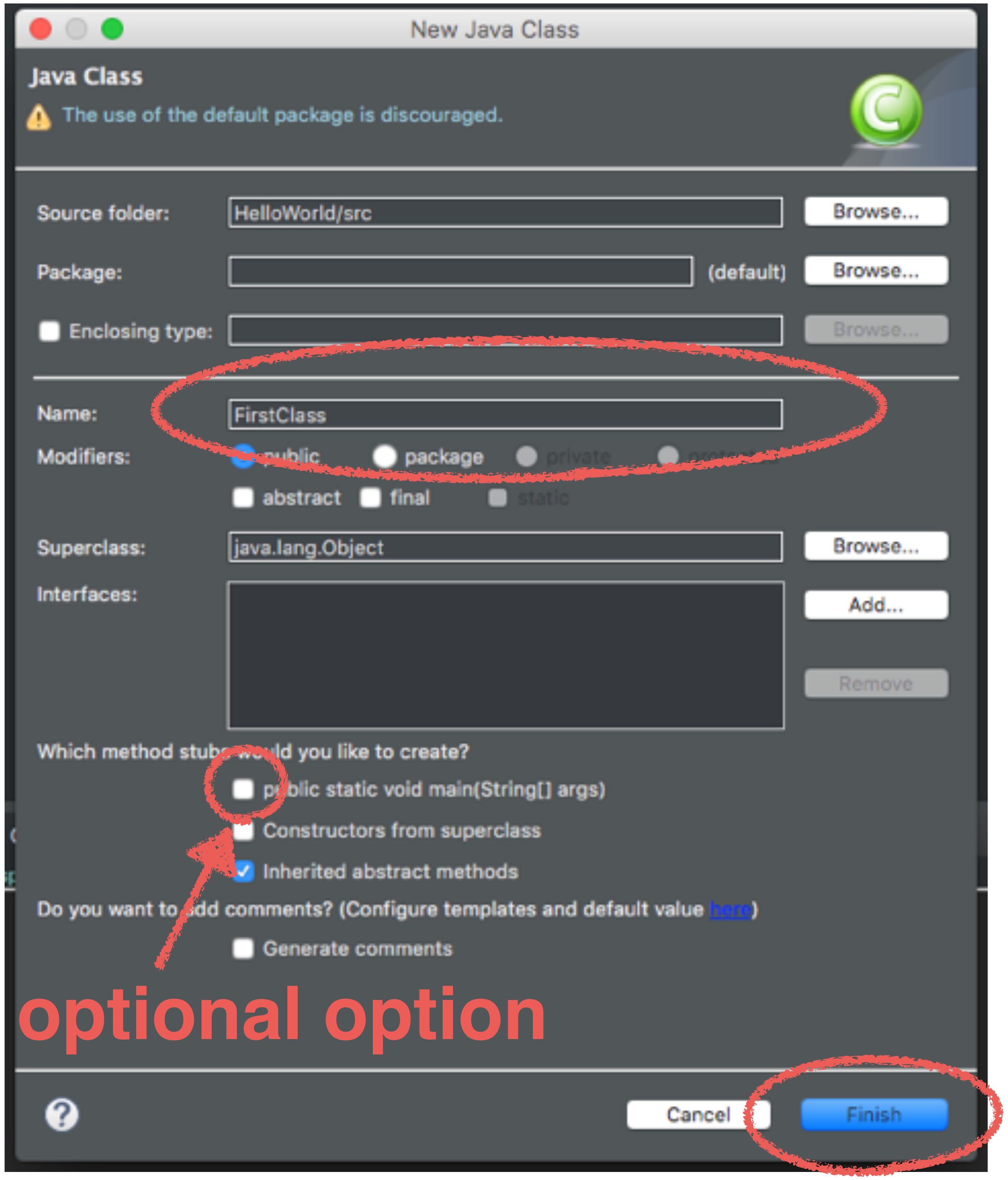
Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Fill in the Name
for class with Name
Convention Rule
(Each word's first
letter is Capitalized)

then click Finish

Project setup



New Java Class

Java Class

The use of the default package is discouraged.

Source folder: HelloWorld/src

Package: (default)

Enclosing type:

Name: FirstClass

Modifiers: [✓ public] package private protected

abstract final static

Superclass: java.lang.Object

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)

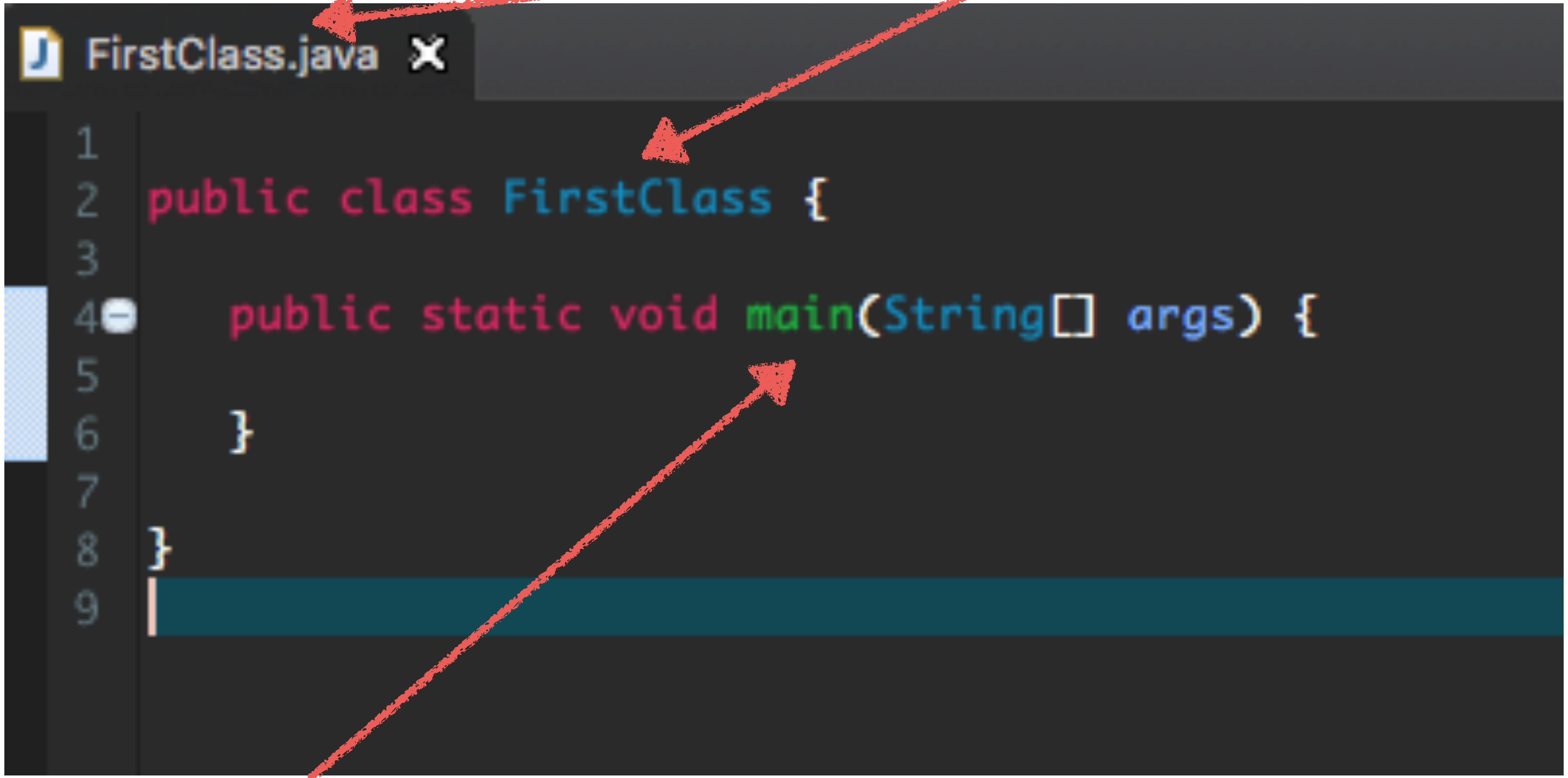
- Generate comments

Fill in the Name for class with Name Convention Rule (Each word's first letter is Capitalized)

then click Finish

Project setup

This is class name



```java

public class FirstClass {

    public static void main(String[] args) {

    }

}

```

This is a method, and we should write code in the method

Main Method

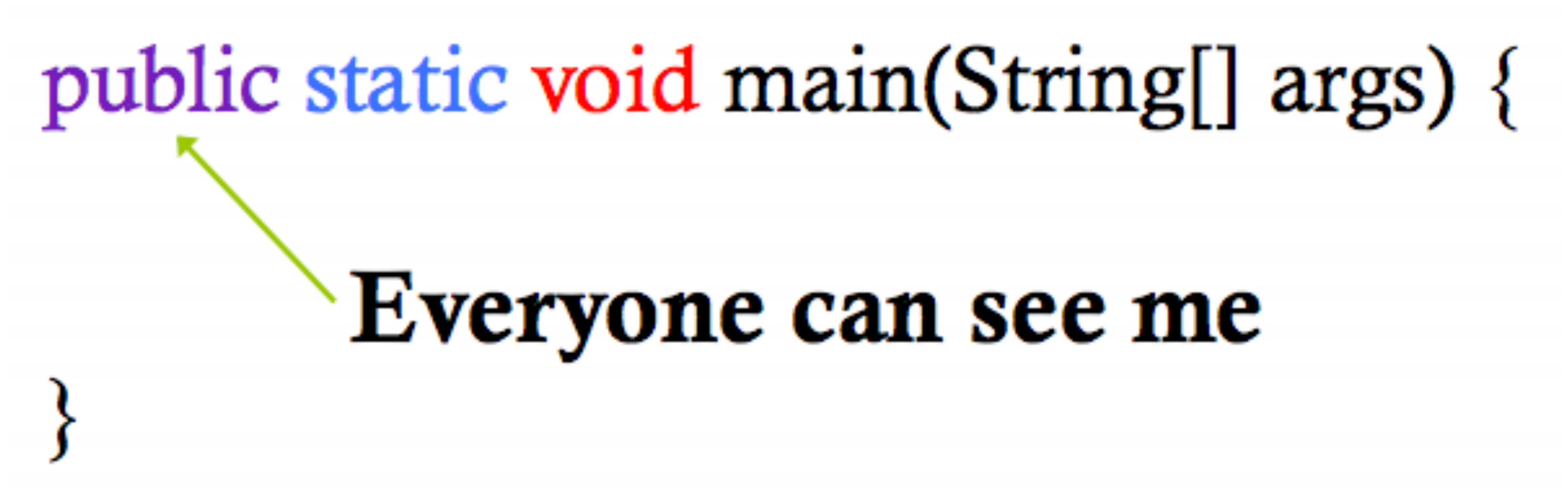
```
public static void main(String[] args) {


}
```

The program runs from the first line of main method

When you try to execute a java code using "java" command, the runtime will load the public class that you are trying to execute and then call the main method defined in the class. The runtime knows that "main" is the method to look for as it is designed that way.

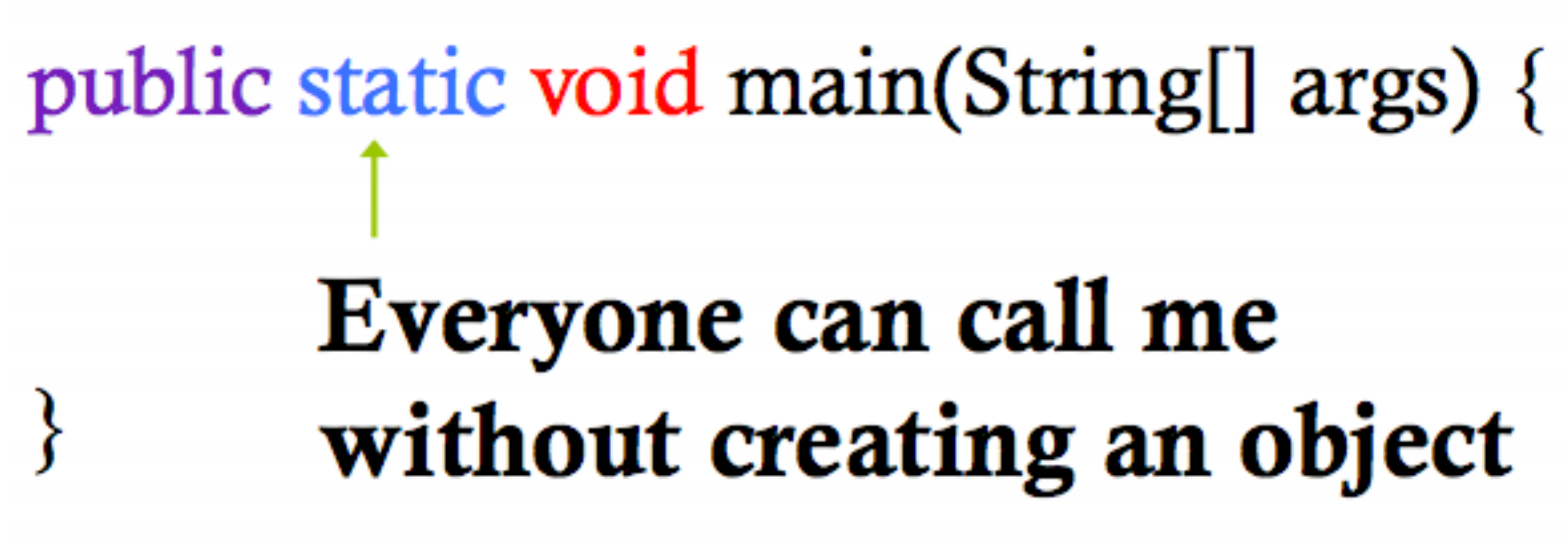
Main Method



```
public static void main(String[] args) {

        Everyone can see me
}
```

Main Method



```
public static void main(String[] args) {

}
```

Everyone can call me without creating an object

Main Method

```
public static void main(String[] args) {
                    ↑
        I have nothing to return
}
```

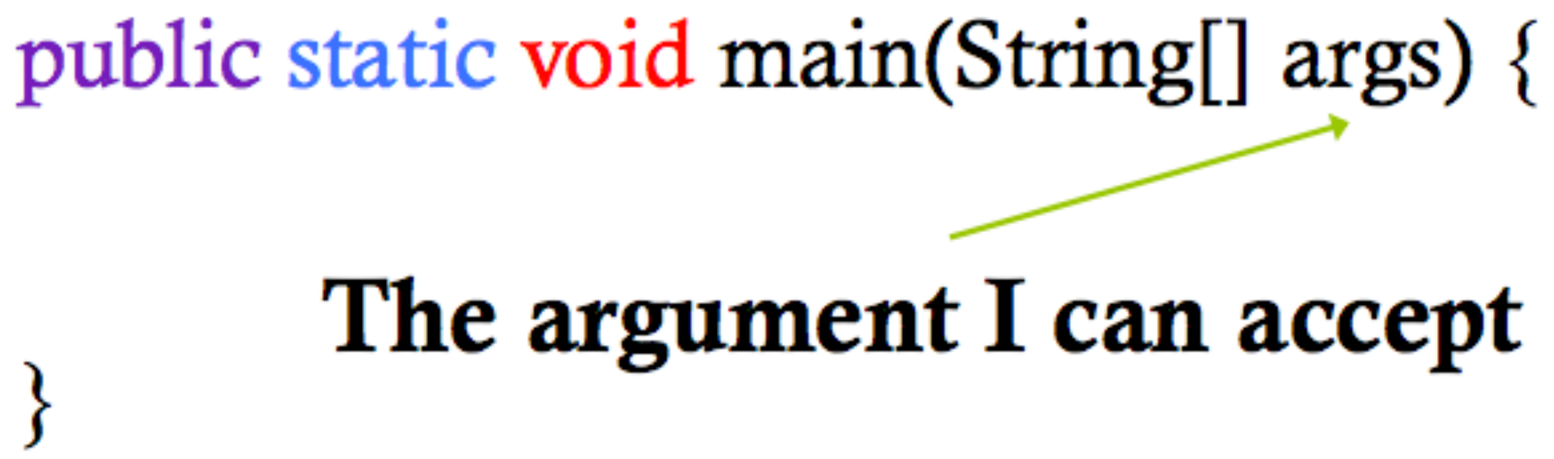
Main Method

```
public static void main(String[] args) {

}
```

My name and I am special

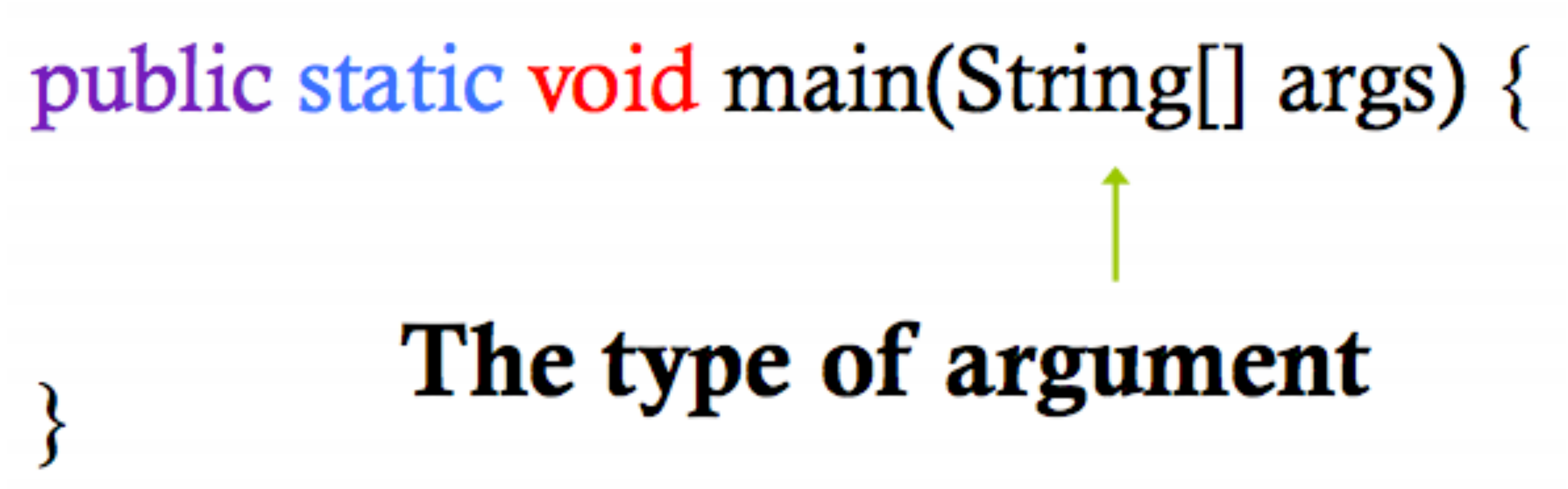
Main Method



```
public static void main(String[] args) {

}
```

The argument I can accept



```
public static void main(String[] args) {

}
```

The type of argument

Print Statement

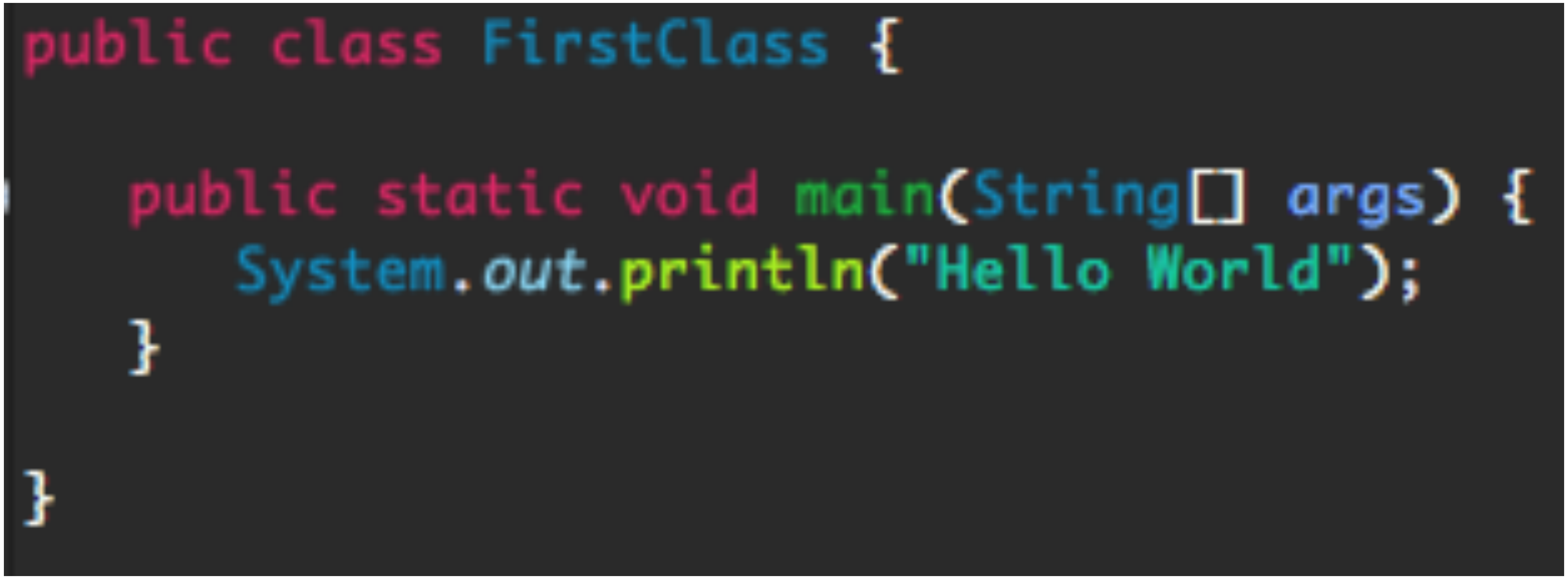
Print statement will allow you to print output to console and debug your code

Each statement ends with a semicolon

```
System.out.print();
System.out.println();
```

Print Statement

Example:



```
public class FirstClass {

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

}
```

Print Statement

```
System.out.print(“Hello World”);
System.out.print(“Hello World”);
```

```
System.out.println(“Hello World”);
System.out.println(“Hello World”);
```

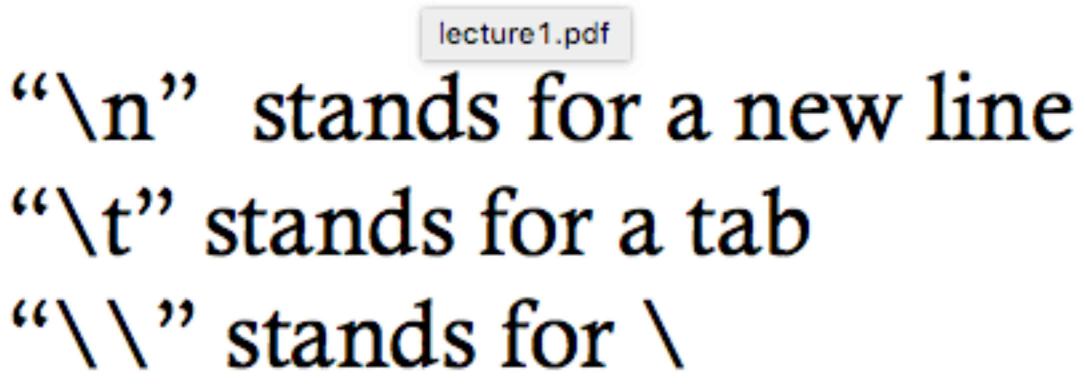
Print Statement

```
System.out.print(“Hello World”);
System.out.print(“Hello World”);
```

```
System.out.println(“Hello World”);
System.out.println(“Hello World”);
```

Print Statement

Special Cases


lecture1.pdf

“\n” stands for a new line

“\t” stands for a tab

“\\” stands for \

“\”” stands for ”

“\’” stands for ’

Print Statement

```
System.out.print(“HelloWorld\n”);

System.out.println(“HelloWorld”);
```

Print Statement

```
 / \
/_ _\
```

Print Statement

```
public static void main(String[] args) {
        System.out.println(“ /\\”);
        System.out.println(“/__\\”);
}
```

Print Statement

```
public static void main(String[] args) {
        int a = 10;
        System.out.println(a);
}
```

Output will be 10

Print Statement

```
public static void main(String[] args) {
    int a = 10;
    int b = 5;
    System.out.println(a + b);
}
```

Output will be 15

Print Statement

```
public static void main(String[] args) {
        int a = 10;
        int b = 5;
        int s = “HelloWorld”;
        System.out.println(a + s + b);
}
```

Output will be 10HelloWorld5

Print Statement

Anything + with a String, the result will be String
For example:

5 + “s” —> “5s”
5 + “” —> “5”
“5” + “1” —> “51”
5 + 1 —> 6

```
System.out.println(5 + “1”);
```

Print Statement

Anything + with a String, the result will be String
For example:

5 + “s” —> “5s”
5 + “” —> “5”
“5” + “1” —> “51”
5 + 1 —> 6

System.out.println(1 + 5 + “1”);

Output will be: “61”