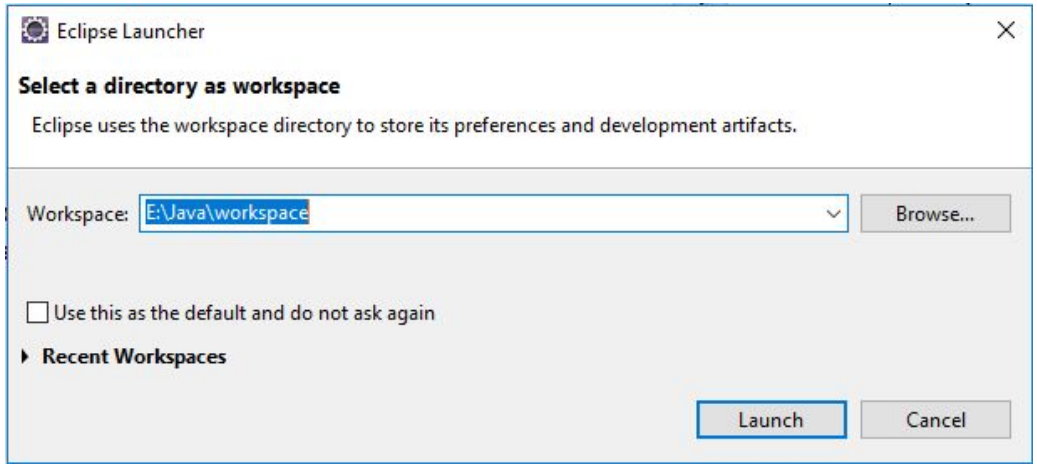
Lecture 03- Lecture 04
How to write Java code in Eclipse

Typical Steps to to write Java code in Eclipse

1. Create an Eclipse Java Project.
2. Create packages in the Eclipse Java project.
3. Add a java class (*.java source file) in the project.
4. Add comments and Javadoc comments frequently whiling coding.
5. Define a class name and its block.
6. Implement a class. (Define class members)
7. Create a main method in the class that starts our program.
8. Instantiate an object.
9. Call an method and an instance! (Method invocation)

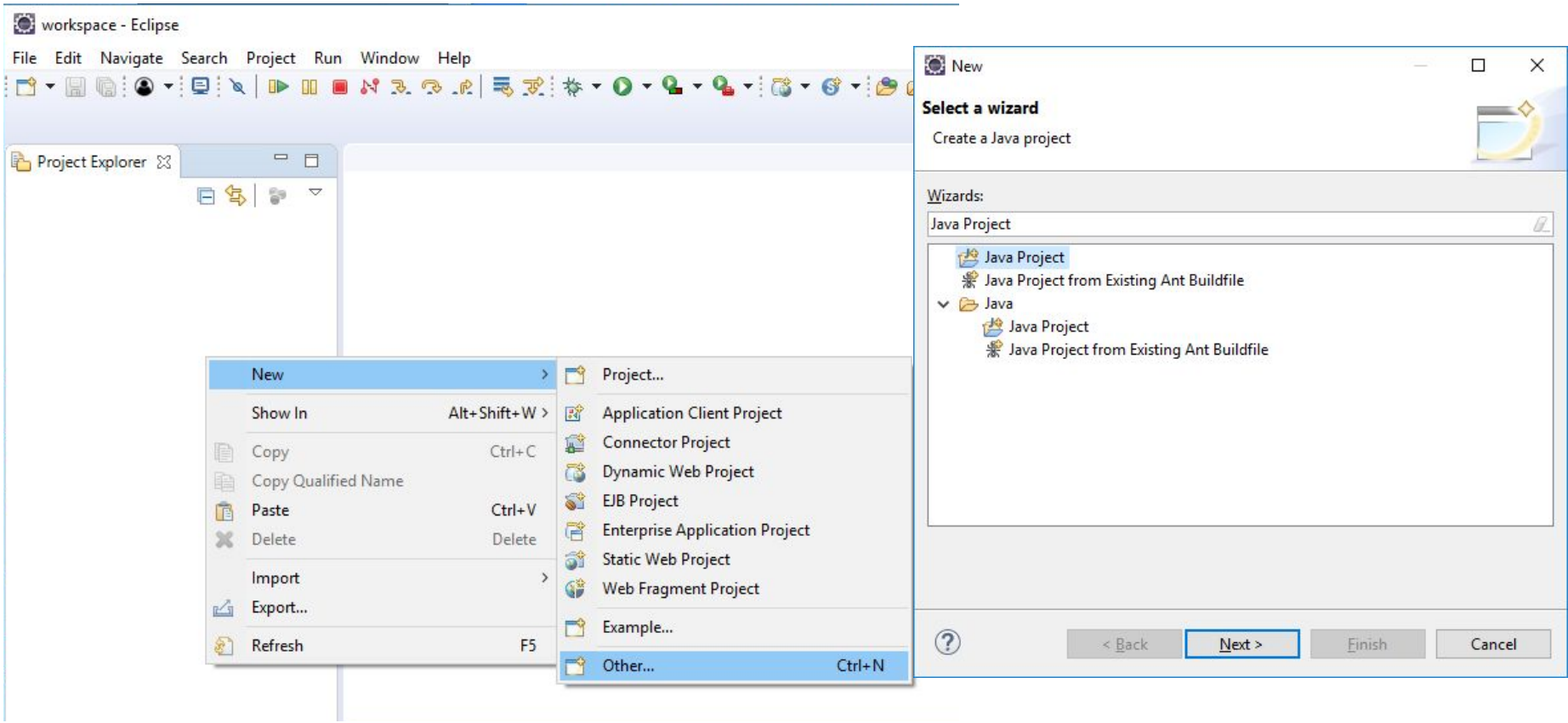
Open Eclipse

- Set the workspace path carefully.
  - **Do not choose any folder whose whole path contains any Korean characters.**
  - **Do not put any Korean characters in the workspace path.**



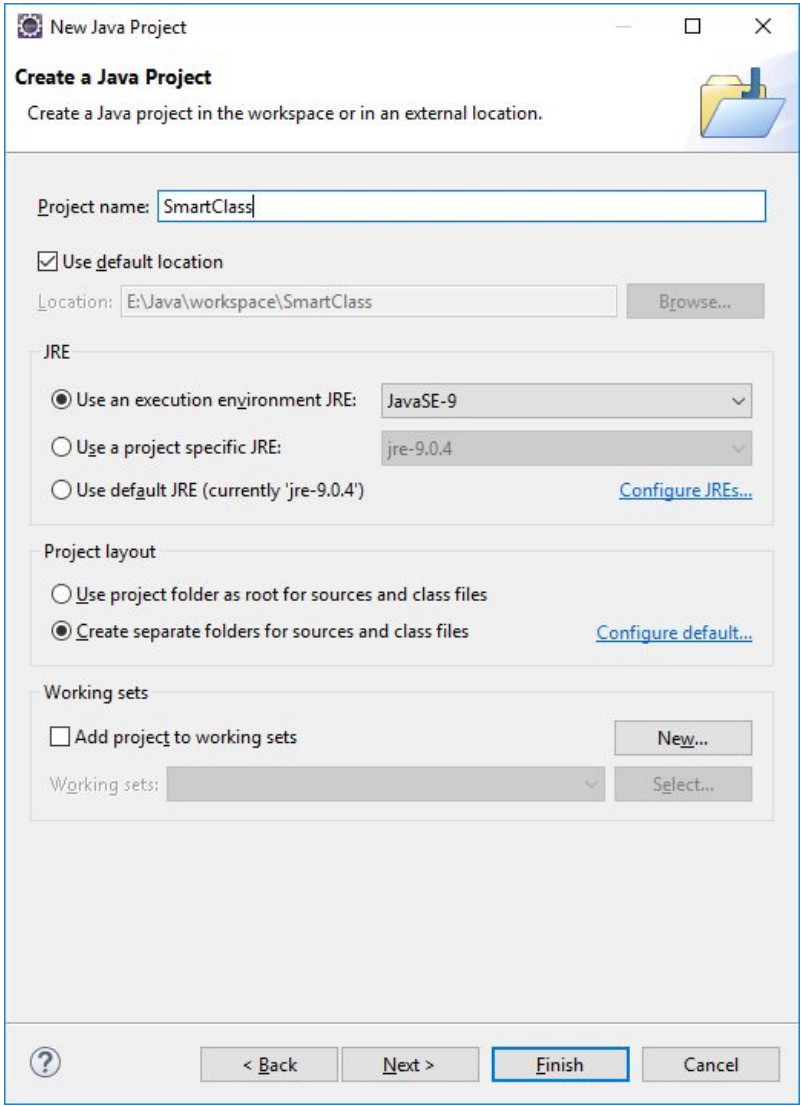
1. Create an Eclipse Java Project (1)

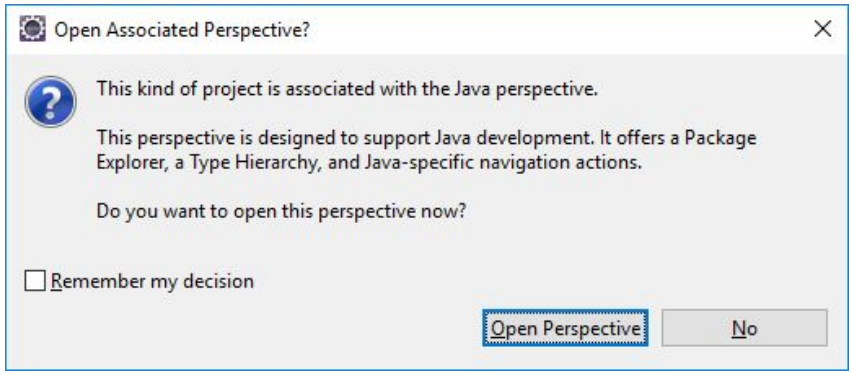
- In Project Explorer
  - Pop-up menu → New → Java Project (if not there, go to Other and seach Java Project in Wizards.)
  - Click Next



1. Create an Eclipse Java Project (2)

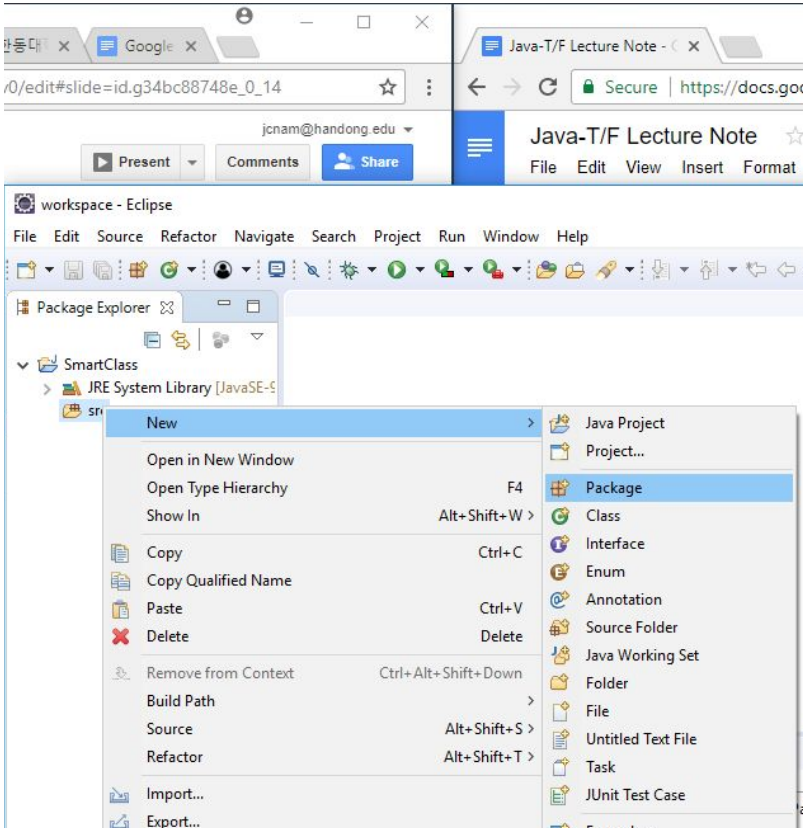
- Type a project name you want to use.
  - e.g., SmartClass
  - Click Finish
  - Click 'Open Perspective' if you see the alert window

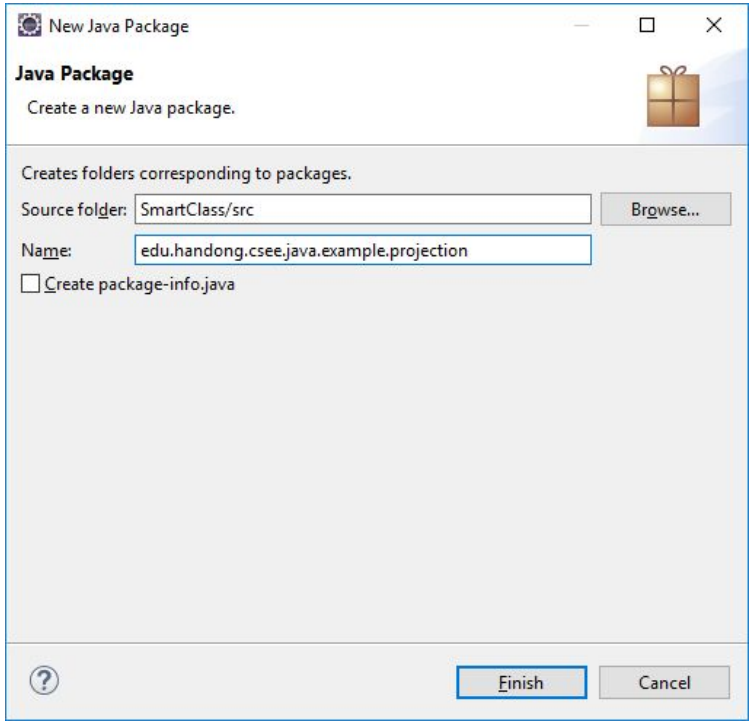




2. Create a package in the 'src' folder

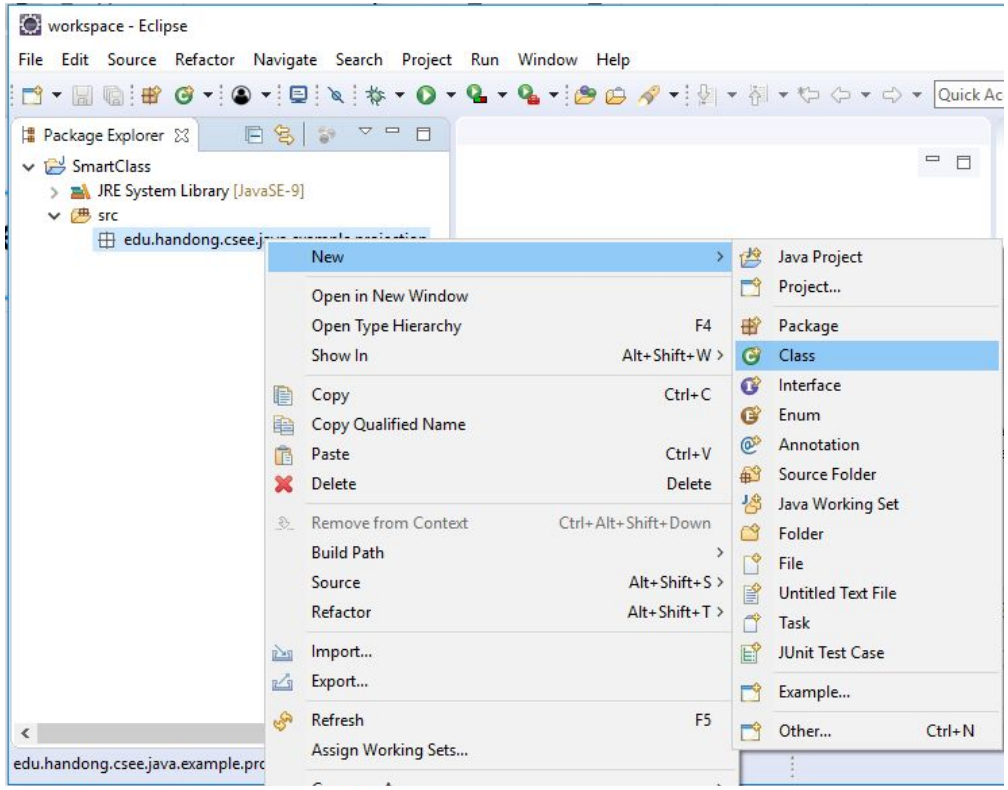
- Pop-up menu on the src folder → New → Package
- Put a package name in "Name:"
  - A package name usually use a domain-like name in a reverse order to make our package unique in the world. e.g., edu.handong.csee.java.mypackage, net.lifove.android.app.lifovebible

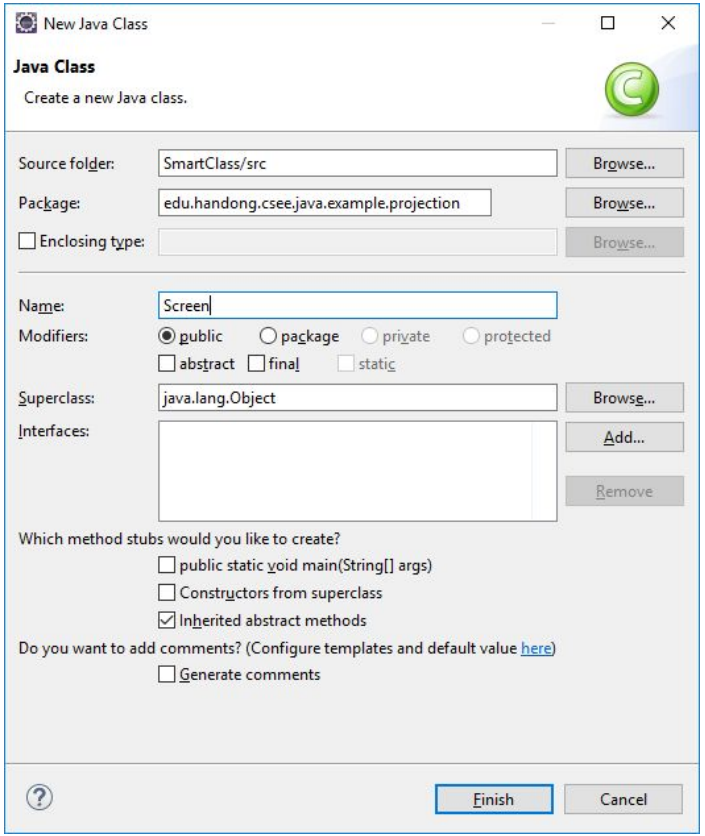




3. Add a Java class in the package

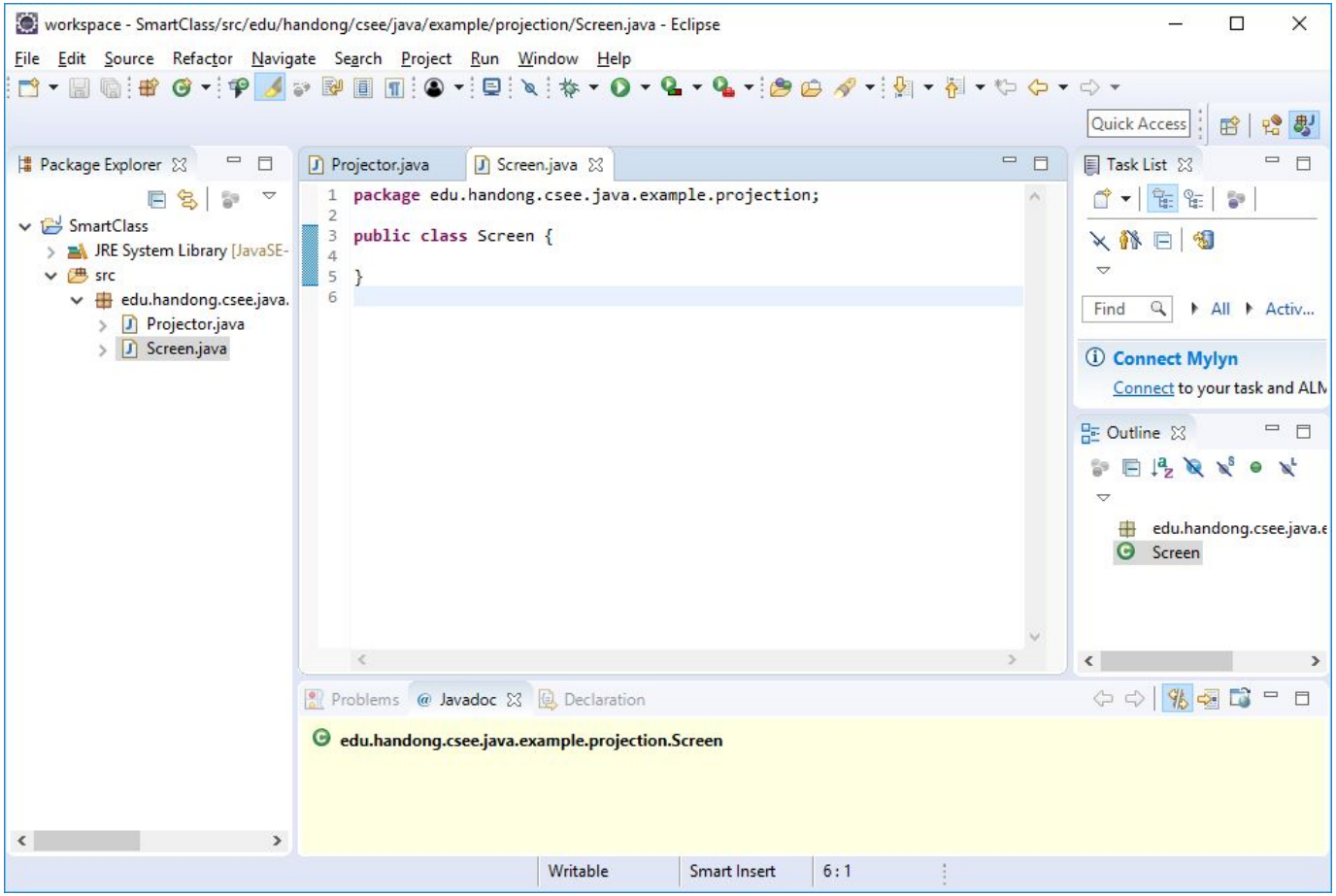
- Pop-up on the package where you want to add a java source file for a class. → New → Class
- Put a class name in "Name:" → Finish
  - The first character must be a capital. e.g., Screen (O) screen (X)





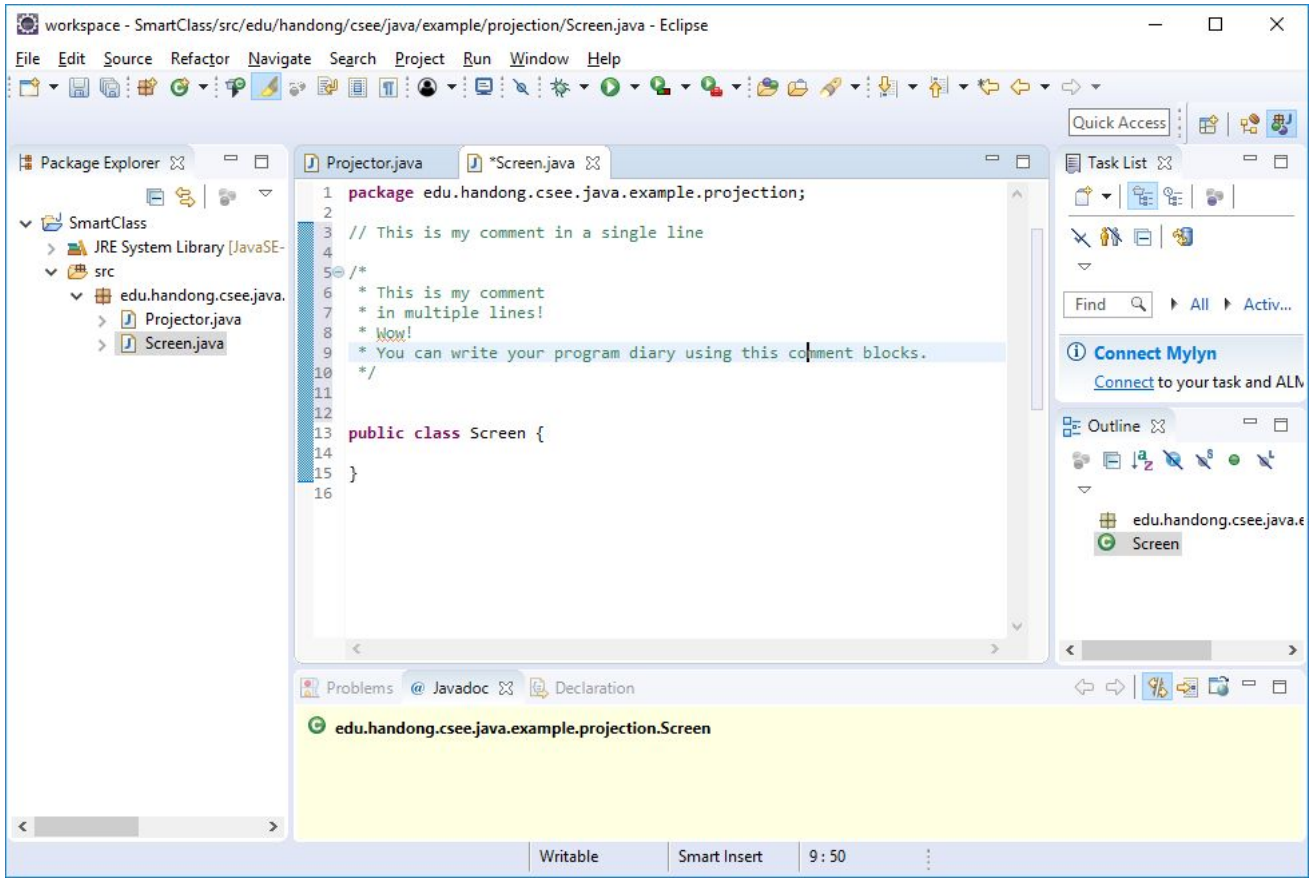
3. Add a Java class in the package (2)

- Then, we can have a java source file in the editor.



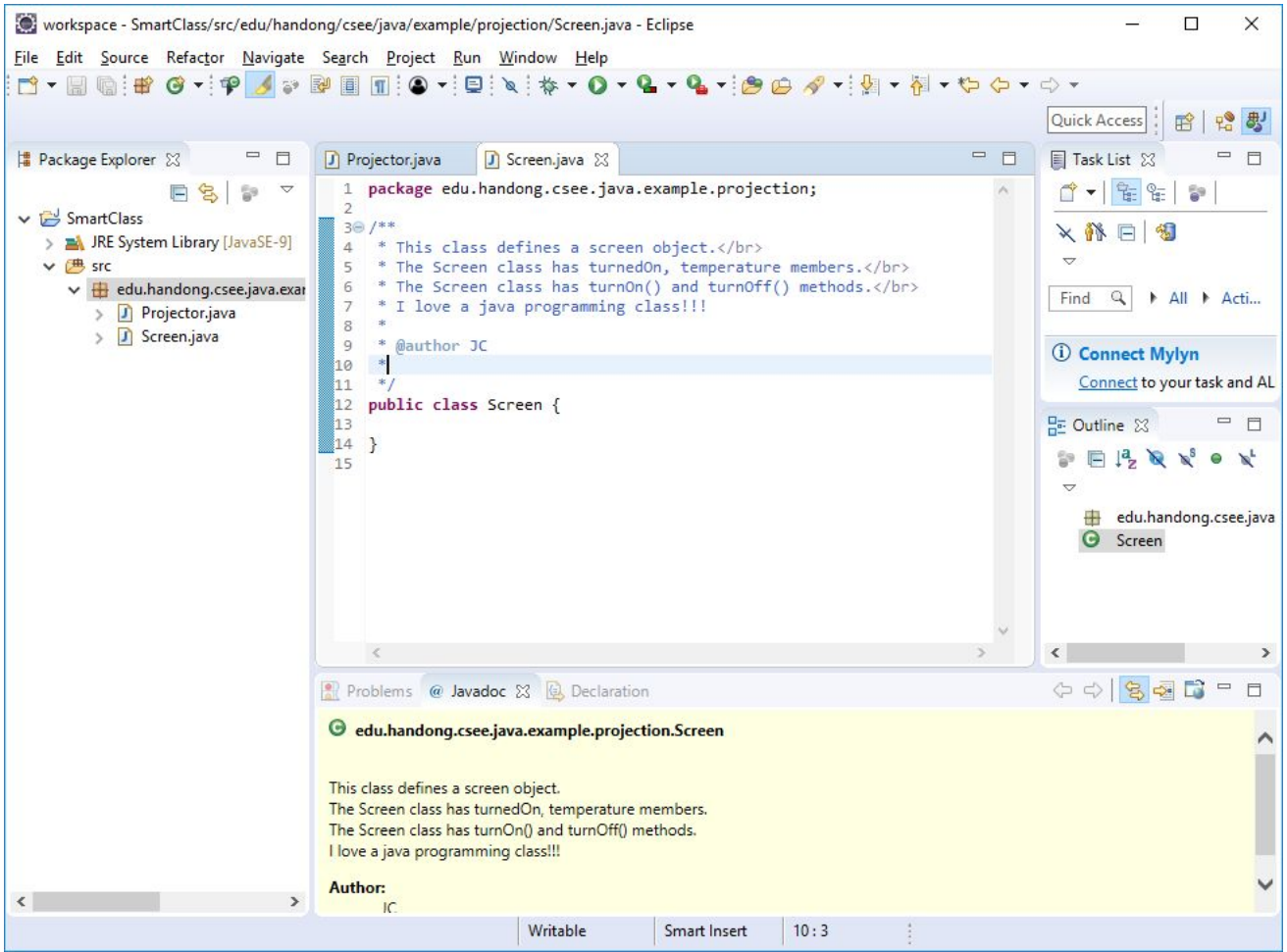
4. Add comments and Javadoc comments

- // : for the single line comment
- /* */: for the multiple-line comment (Start with '/*' and end with '*/')



4. Add comments and Javadoc comments (2)

- Javadoc comment (Slides: How to generate javadoc htmls https://goo.gl/dbfc6m)
  - /** */: for the javadoc comments (**Start with '/**'** and end with '*/')



5. Define a class name and its block

- We already added a class block in Eclipse (See Step 3)
- A class block consists of:
  - modifier: public or nothing
    - public means my class can be used by other developers by importing it in other developers java code.
    - If we don't write any modifier, this means my class can be used only within its package. So we say the class is protected.

	public	(default)
Same package	O	O
Others	O	

  - class: Just tells to JVM we define a class.
  - any name given by me
  - block { … }

5. Define a class name and its block (2)

- A java file may contain more than two classes but it must have only one public class.
- The first character of a class name must be a capital
  - e.g., MyClass (O), myClass (X)
- A java file name must be same as a public class name.
  - MyClass.java

```
package edu.handong.csee.java.lab;

// the first character of a class name must be a capital.
public class MyClass { // <modifiers> class <Name>

}

class MyProtectedClass {

}
```

6. Implement a class (Define class members)

- Add an instance variable
  - An instance variable: data for a class a.k.a a member variable, a field
  - Naming convention
    - Starts with a lowercase, m, as a prefix. the 'm' stands for a 'm'ember variable.
    - Than put an actual name.
    - Use a uppercase for the first character of a word.
    - There must not be a space and avoid to use special characters except for '_' for one variable name
    - e.g. mName, mMyVariable_1, mMyProjector, mCounterForTurnedOnProjectors,...
- Add a method
  - A method: an action for a class
  - Naming convention
    - Use a lowercase for the first character of the name
    - Use a uppercase of the first character of the second word and its all next words
    - e.g., turnOnMyProjector, turnOffMyProjector, getTemperature, setTemperature,...
  - <modifiers> <return type> <method name>(parameters)
    - e.g., public void myMethod() {}

6. Implement a class

- Decide access modifiers for a method.

	public	protected	(default)	private
Insider class	O	O	O	O
Same Package Class	O	O	O	X
Same Package Sub-Class	O	O	O	X
World	O	X	X	X

- Add a return (output) type
  - void: nothing to output
  - ...

6. Implement a class

Projector.java | Screen.java | Screen | Controller.java

```java
package edu.handong.csee.java.example.projection;

public class Projector {

    int mLampTemperature = 20; // this is an instance variable

    /**
     * This instance variable shows a description of this class.
     */
    public String mDescription = "This is a projector class";

    /**
     * This is a method for turning on a projector
     */
    public void turnOn() {

        System.out.println("My project is turning on!!!");
        getLampTemperature(); // call a private method in my current class

    }

    /**
     * This is a method for turning off a projector
     */
    public void turnOff() {

        System.out.println("My project is turning off!!!");

    }

    /**
     * This is a method for getting the lamp temperature of the projector
     */
    private void getLampTemperature() {

        System.out.println("My project tenperature is " + mLampTemperature);

    }
}
```

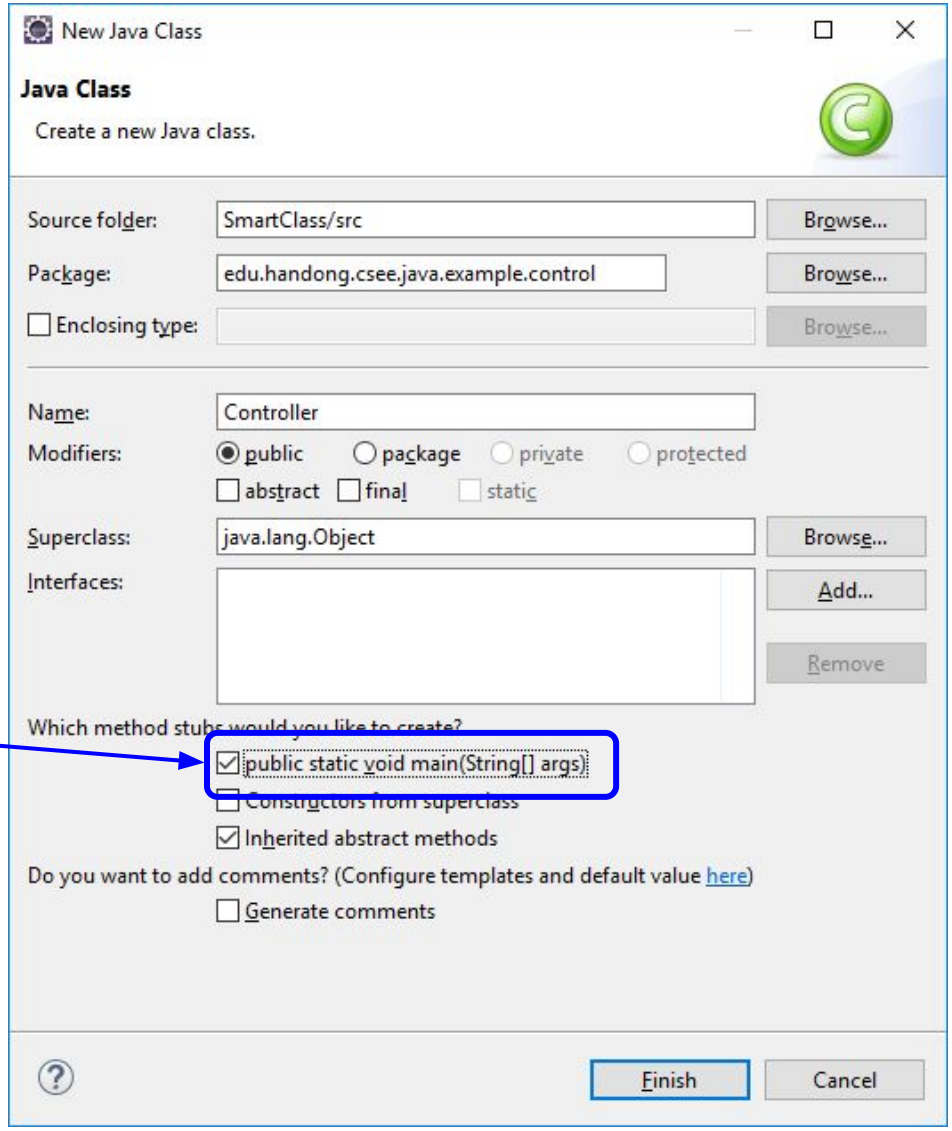
7. Create a main method in the class that starts our program.

- We need a entry point to execute a java program.
- We can add a main method in every class
- But each class cannot have more than one main methods.
- the main method always look same!

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

}
```

- When you add a new class, you can just need to tick this option to add the main method.



7. Create a main method in the class that starts our program. (2)

workspace - SmartClass/src/edu/handong/csee/java/example/control/Controller.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- SmartClass
  - JRE System Library [JavaSE-9]
  - src
    - edu.handong.csee.java.exa
      - Controller.java
    - edu.handong.csee.java.exa
      - Projector.java
      - Screen.java
  - doc

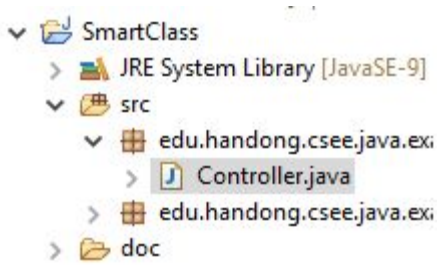
Projector.java | Screen.java | Screen | Controller.java

```
1  package edu.handong.csee.java.example.control;
2
3  public class Controller {
4
5      public static void main(String[] args) {
6
7          // Type these lines for you test
8          // These lines print out texts.
9          System.out.println("This is my controller!!");
10         System.out.println("My first word used as an input is" + args[0]);
11         System.out.println("My first word used as an input is" + args[1]);
12     }
13
14 }
15
```

Now you can run your program because you have a main method!!!

7. Create a main method in the class that starts our program. (3)

- Various ways to run your program in Eclipse!
  - Choose and open a java file that has a main method in Project Explorer.



  - Then click a play button or "Run as → 1 Java Application" or Ctrl+F11

7. Create a main method in the class that starts our program. (3)

-

Projector.java | Screen.java | Screen | Controller.java

```java
package edu.handong.csee.java.example.control;

public class Controller {

    public static void main(String[] args) {

        // Type these lines for you test
        // These lines print out texts.
        System.out.println("This is my controller!!");
        System.out.println("My first word used as an input is " + args[0]);
        System.out.println("My first word used as an input is " + args[1]);
    }

}
```

You may not get the following results because you did not add inputs. See the next slides to solve the problem.

Problems | Javadoc | Declaration | Console

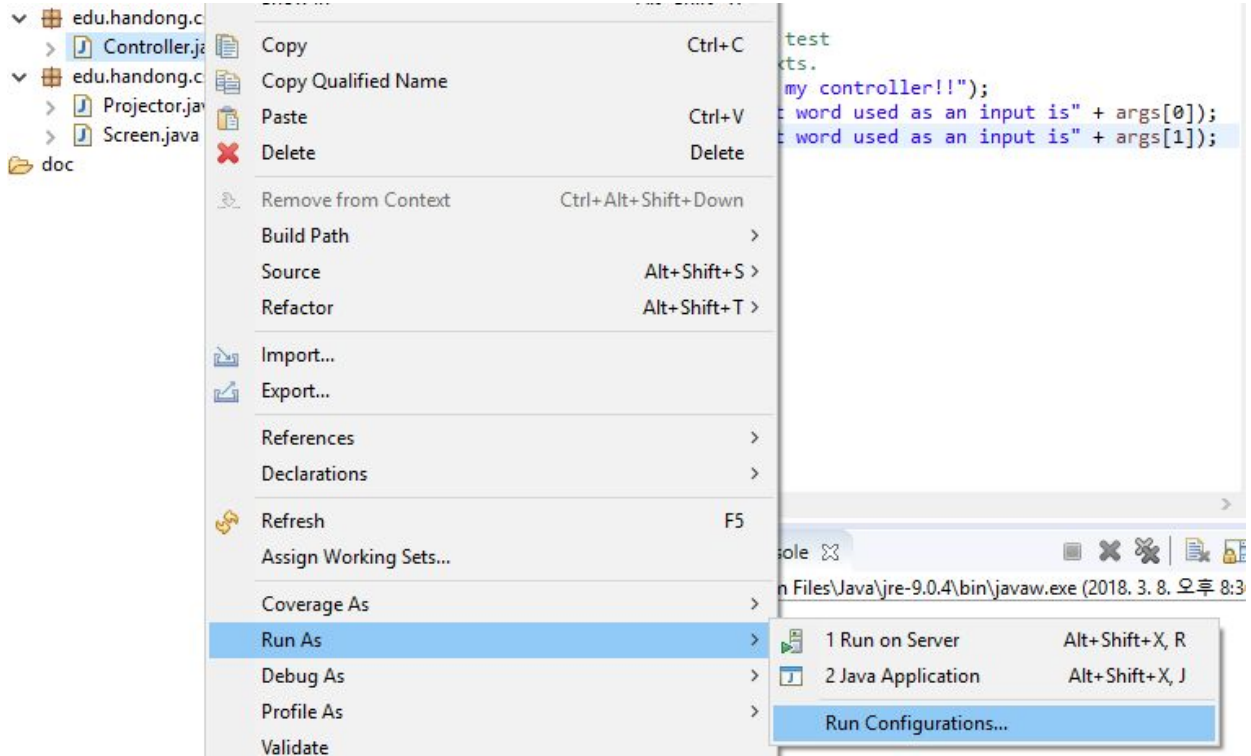
<terminated> Controller [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (2018. 3. 8. 오후 8:5

Error Results

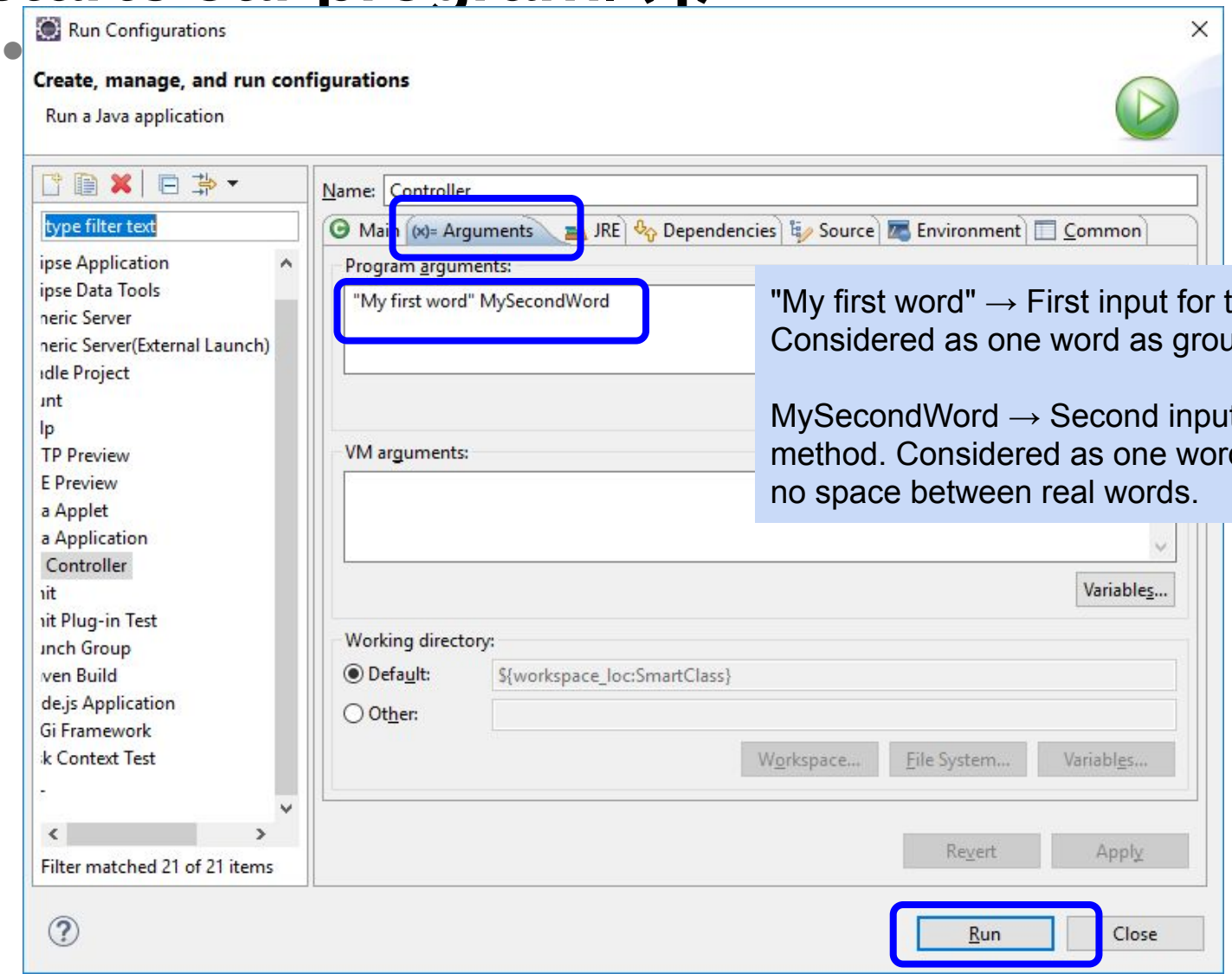
```
This is my controller!!
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
        at edu.handong.csee.java.example.control.Controller.main(Controller.java
```

7. Create a main method in the class that starts our program. (3)

- public static void main(String[] args)
  - String[] args
    - Initial text input when we run a program
      - it contains a series of words.
- Let's add our initial input when running the java program in Eclipse
  - Popup menu on a java file having a main method → Run as → Run Configuration



7. Create a main method in the class that starts our program. (4)



"My first word" → First input for the main method. Considered as one word as grouped by using "".

MySecondWord → Second input for the main method. Considered as one word because there is no space between real words.

7. Create a main method in the class that starts our program. (3)

- Projector.java | Screen.java | Screen | Controller.java

```java
package edu.handong.csee.java.example.control;

public class Controller {

    public static void main(String[] args) {

        // Type these lines for you test
        // These lines print out texts.
        System.out.println("This is my controller!!");
        System.out.println("My first word used as an input is " + args[0]);
        System.out.println("My first word used as an input is " + args[1]);
    }

}
```

Problems | Javadoc | Declaration | Console

```
<terminated> Controller [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (2018. 3. 8. 오후 8:5
This is my controller!!
My first word used as an input is My first word
My first word used as an input is MySecondWord
```

Results →

8. Instantiate an object

- To instantiate an object
  - <Class name we want to create as an instance> <instance name we use in our code> = new <Class name>()
    - e.g.,
      - Projector myProjector = new Projector();
  - Instantiate of an object may have some input
    - e.g.,
      - Computer myComputer = new Computer("name");
  - Constructors
    - new Projector() is calling a constructor
    - A constructor is a special method that creates an instance. (We will learn it later in detail.)
      - Slides: https://goo.gl/at77hR

9. Call members of an instance (Method invocation and access fields)

- We can run actions of an object
  - Method calls
  - Method invocations
- We can access public instance variables (fields).

Projector.java | Screen.java | Screen | Controller.java

```java
package edu.handong.csee.java.example.control;

import edu.handong.csee.java.example.projection.Projector;

public class Controller {

    public static void main(String[] args) {

        // Type these lines for your test
        // These lines print out texts.
        System.out.println("This is my controller!!");
        System.out.println("My first word used as an input is " + args[0]);
        System.out.println("My first word used as an input is " + args[1]);

        // instantiate my controller
        Controller nth413Controller = new Controller();

        // a method call of an instance, nth413Controller, of the Controller class
        nth413Controller.turnOnProjector();

    }

    public void turnOnProjector() {

        // instantiate the Projector class
        Projector nth413Projector = new Projector();

        // Print out Projector's description
        // by directly accessing the public instance variable.
        System.out.println("My project's description:" +  nth413Projector.mDescription);

        // a public method call of an instance, nth413Projector, of the Projector class
        nth413Projector.turnOn();

    }
}
```

Problems | Declaration | Console

```
<terminated> Controller [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (2018. 3. 9. 오전 8:44:41)
This is my controller!!
My first word used as an input is My first word
My first word used as an input is MySecondWord
My project's description:This is a projector class
My project is turning on!!!
My project tenperature is 20
```