

Basic Structure of Java Programming

Sungchul Lee

Learning Object

- Make First Java Program
 - Create new project and java file
 - Explain of the Code
- Basic Structure of Java Programs
 - Keywords
 - Statements
 - Code block
 - Indentation
 - Comments
- Console Window

First Java Program

- Execute Eclipse
- Create New Project
 - Choose the right JRE
- Create a Java Class file in project
 - Make new file (.java)
 - Make own code on the java file
- Compile and run the file

Step 1: Install Java in VSCode

➤ Java in VS Code

– <https://code.visualstudio.com/docs/languages/java>

Install Visual Studio Code for Java

To help you set up quickly, we recommend you use the **Coding Pack for Java**, which is the bundle of VS Code, the Java Development Kit (JDK), and a collection of suggested extensions by Microsoft. The Coding Pack can also be used to fix an existing development environment.

Install the Coding Pack for Java - Windows

Install the Coding Pack for Java - macOS

Click

JavaCodingPack-0.4.2.exe

Click

Installing Components

Download JDK

[오류 5:37:59] AdoptOpenJDK-17-x64

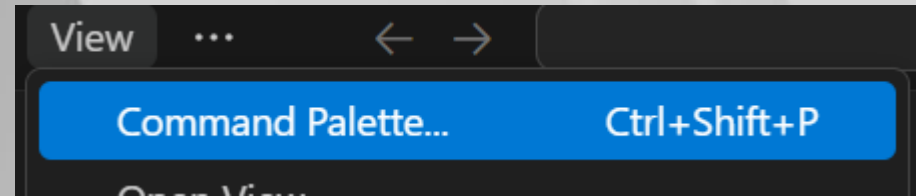
Click

Back Next

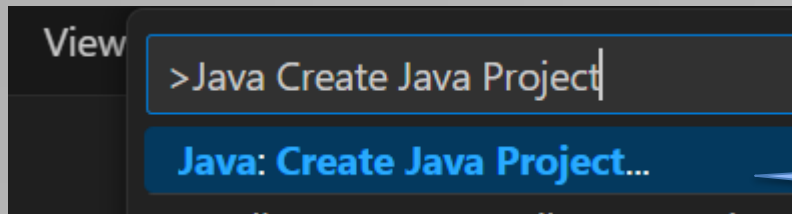
Step 2: Create a Java Class File

➤ 보기 -> 명령 팔레트

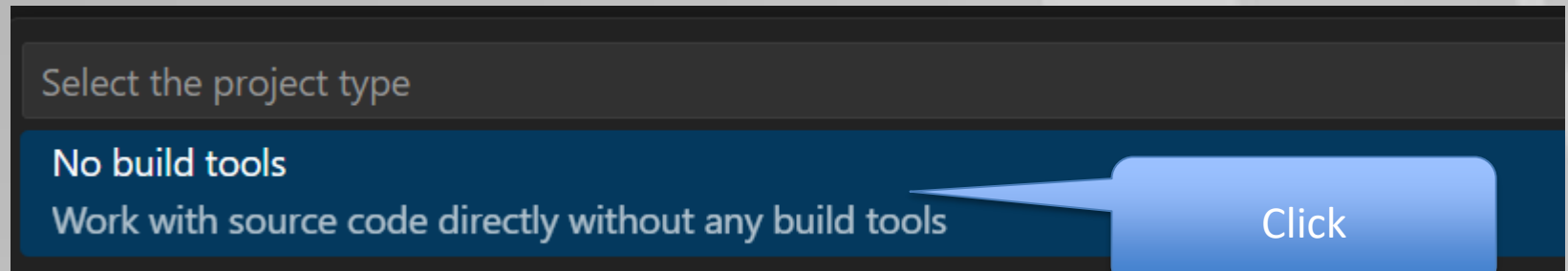
– Ctrl + shift + p



➤ Java Create Java Project



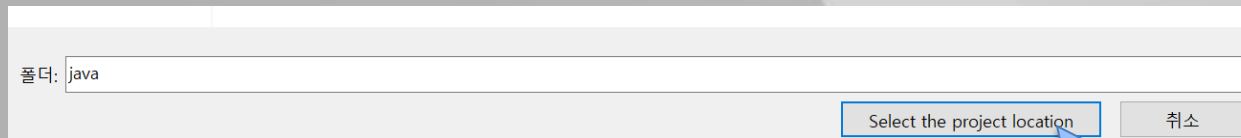
Click



Click

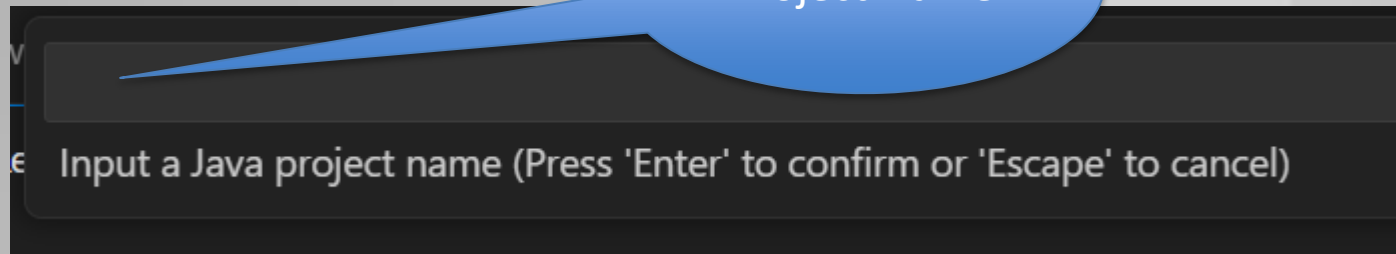
Step 2: Create a Java Class File

➤ Project Location



A screenshot of a Java IDE dialog box. It features a text input field with the label '폴더:' (Folder:) and the text 'java' entered. To the right of the input field are two buttons: 'Select the project location' and '취소' (Cancel).

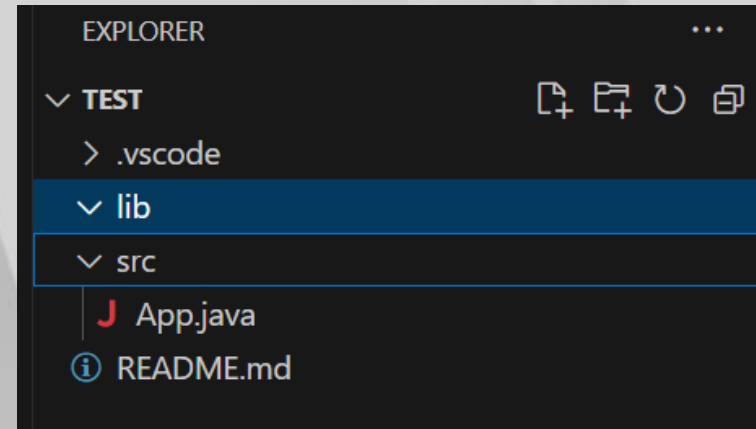
➤ Input Project Name



A screenshot of a Java IDE dialog box with a dark background. It contains a text input field at the top. Below the input field is the text 'Input a Java project name (Press 'Enter' to confirm or 'Escape' to cancel)'. Two blue callout bubbles are present: one pointing to the input field labeled 'Project Name', and another pointing to the right side of the dialog labeled 'Project Location'.

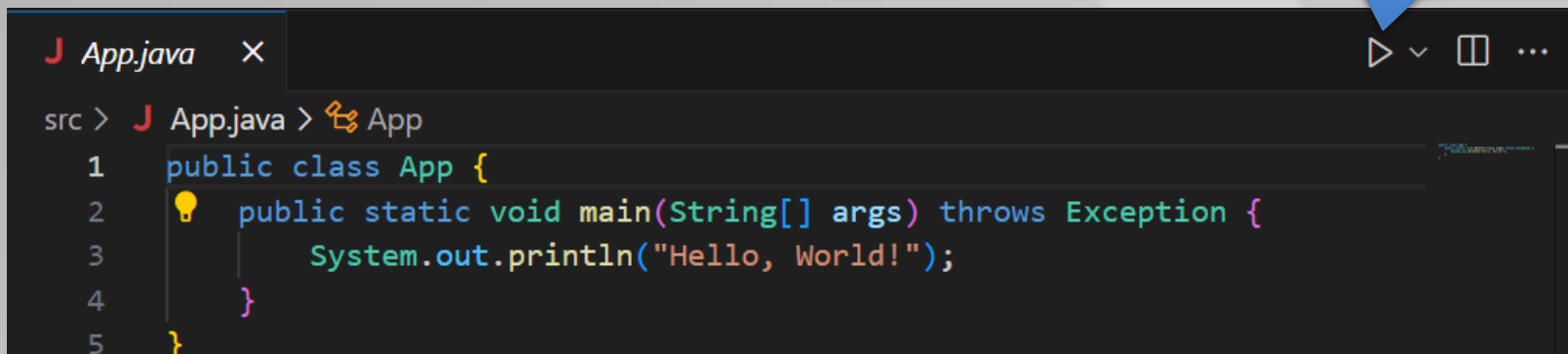
Step 3: Coding

➤ Project Folder



➤ Coding on App.java File

Click for
run



A screenshot of the Visual Studio Code editor showing the 'App.java' file. The editor has a tab at the top labeled 'App.java' with a close button. The code is written in a dark theme with syntax highlighting. The code is as follows:

```
src > App.java > App
1 public class App {
2     public static void main(String[] args) throws Exception {
3         System.out.println("Hello, World!");
4     }
5 }
```

On the right side of the editor, there is a toolbar with a play button (run icon), a dropdown arrow, a window icon, and a menu icon (three dots). A blue speech bubble with the text 'Click for run' points to the play button.

Step 3: Coding

```
public class App {  
    public static void main(String[] args) throws  
Exception {  
        System.out.println("Hello, World!");  
    }  
}
```



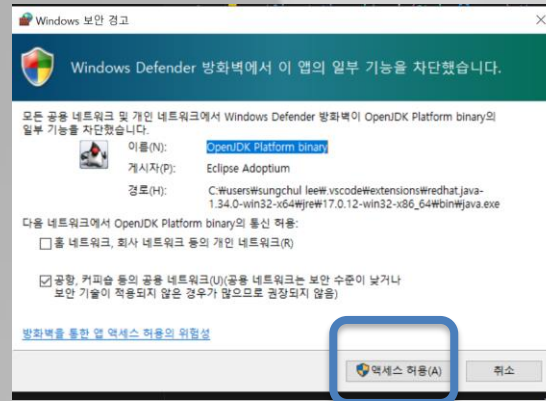
A screenshot of an IDE window titled 'App.java'. The code editor shows the following Java code:

```
src > J App.java > App  
1 public class App {  
2     public static void main(String[] args) throws Exception {  
3         System.out.println("Hello, World!");  
4     }  
5 }
```

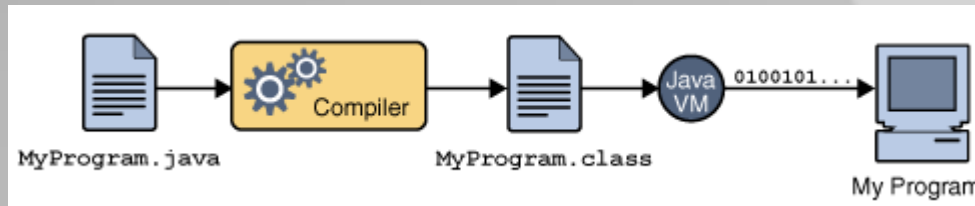

Step 4: Run

➤ Run

– Compiler Makes compiled class file



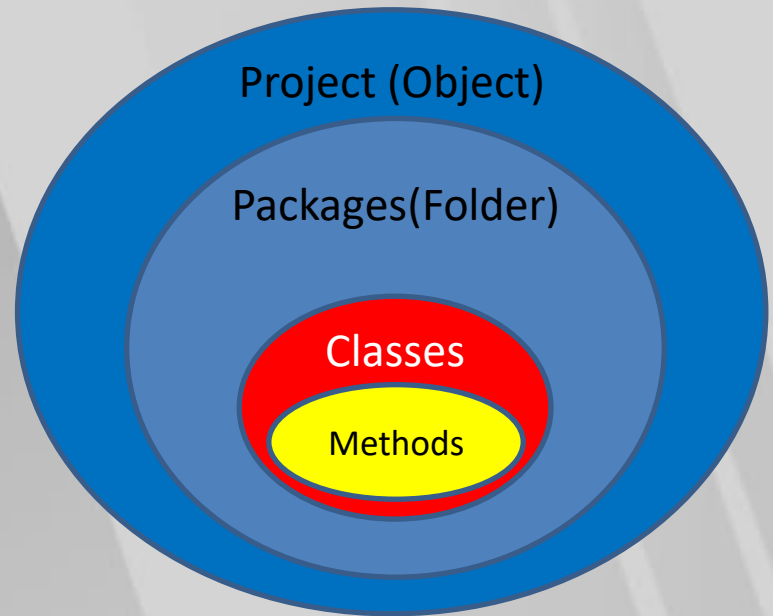
Click for
run



➤ Result on Console Windows

```
C:\Users\Sungchul Lee\Downloads\java\Test\bin
Hello, World!
PS C:\Users\Sungchul Lee\Downloads\java\Test>
```

Explain of the Code



Everyone
can access

This is
class

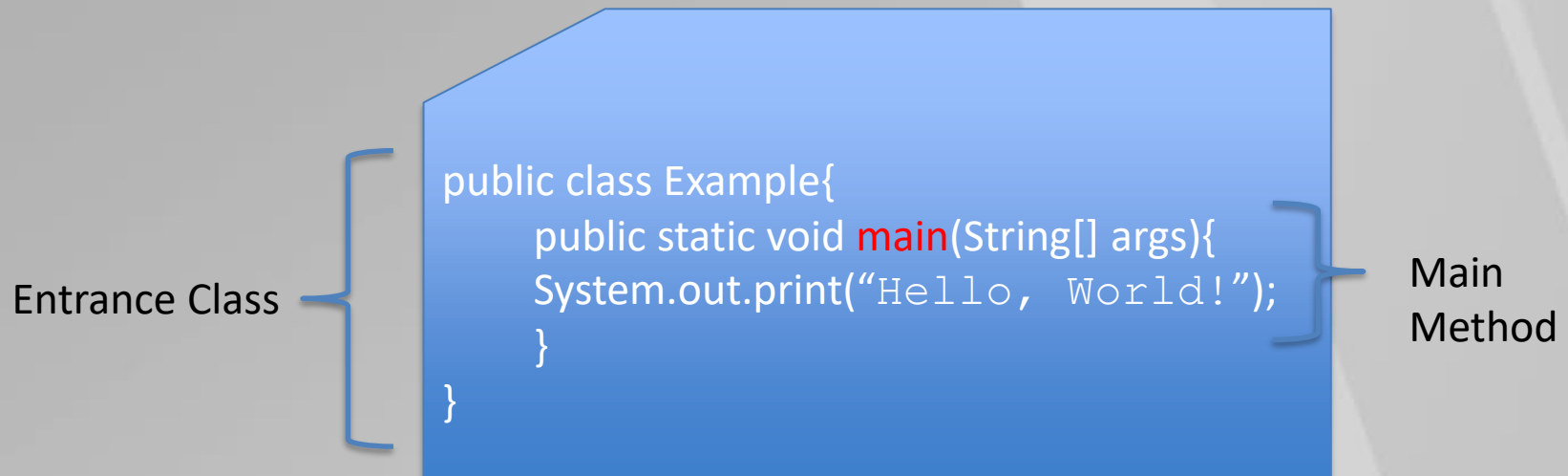
Name of
this class

```
public class Example {  
    // A Java program begins with a main() method.  
    public static void main(String[] args) {  
        System.out.println("Hello, World!");  
    }  
}
```

Method

Basic Structure of Java Programs

- There is an entrance class file
 - Called main class
 - A main class has a single method: main



Simple code to print out "Java" on the screen

Keywords

- Reserved words, that have a specific meaning to the compiler.
 - **Cannot be used for other purposes** in the program
 - such as name.
 - Specific purpose and Location
 - E.g. public, class, void, static, ...etc
 - main is not a keywords. it is predefined like function

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

Statements

- A **statement** represents an action or a sequence of actions.
- Every statement in Java ends with a **semicolon(;).**
- There are three statements
 - Declaration
 - Expression
 - Control flow

```
public class Example{  
    public static void main(String[] args){  
        //Expression Statement  
        System.out.print("Hello, World!");  
    }  
}
```

Code Block

- A list of statements inside braces ({ ...}).
 - Does not need to end in a semicolon.
- The Case of Code block:
 - Every class has a **class block** that groups the data and methods of the class.
 - Every method has a **method block** that groups the statements in the method.
 - Most control flow statement has a **statement block** that groups of expression/declaration statements

```
public class Example{ //Start class block
    public static void main(String[] args){ //Start method block
        System.out.print("Hello, World!");
    } //End method block
} //End class block
```

Indentation

- The **Java compiler doesn't care** about indentation.
- Use indentation to make your code more readable.
- Indent one level for each opening brace and move back out after each closing brace.

Bad Indentation

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

Good Indentation

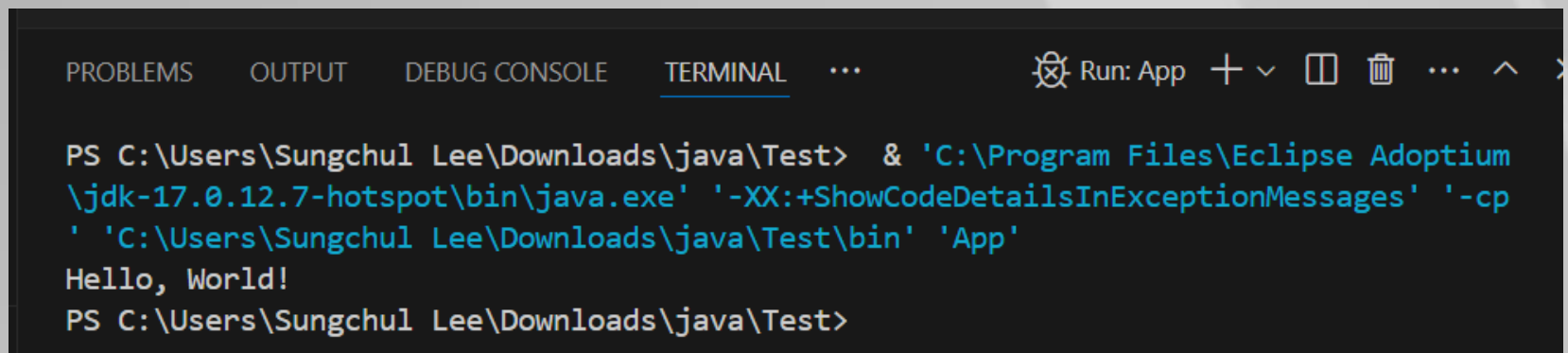
```
public class Example{  
    public static void main(String[] args){  
        System.out.print("Hello, World!");  
    }  
}
```

Comments

- The compiler ignores comments in program
- Three kinds of comments:
 - `/* text */`
 - The compiler ignores everything from `/*` to `*/`.
 - `/** documentation */`
 - This indicates a documentation comment (doc comment, for short). The JDK javadoc tool uses doc comments when preparing automatically generated documentation.
 - `// text`
 - The compiler ignores everything from `//` to the end of the line.

Console Window

- Console window (also called standard output window)
 - Is a text user interface
 - Used to output data (e.g. computation results)
 - through a system object `System.out`



The screenshot shows the Eclipse IDE's interface with the 'TERMINAL' tab selected. The terminal displays the command to run a Java application and its output.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  ...  Run: App  +  -  [Icon]  [Icon]  ...  ^  >

PS C:\Users\Sungchul Lee\Downloads\java\Test> & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.12.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Sungchul Lee\Downloads\java\Test\bin' 'App'
Hello, World!
PS C:\Users\Sungchul Lee\Downloads\java\Test>
```

Summary

