



# CS 152: Data Structures with Java

## Hello World with the IntelliJ IDE

Instructor: **Joel Castellanos**

e-mail: [joel.unm.edu](mailto:joel.unm.edu)

Web: <http://cs.unm.edu/~joel/>

Office: Electrical and Computer  
Engineering building (ECE).  
Room 233



# Install Java Development Kit (JDK) 1.8

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Most computers already have the **Java Runtime Environment** installed. However, to create Java programs, the **Java Development Kit** is required. Note: the JDK includes a copy of the matching version of the JRE.

The screenshot shows the Oracle Java SE Downloads page. A green callout box with the text "Download and Install JDK 1.8" has an arrow pointing to the "DOWNLOAD" button for the "Java Platform (JDK) 8u60". The page layout includes a top navigation bar with the Oracle logo, a search bar, and links for Sign In/Register, Help, Country, Communities, I am a..., I want to..., and OTN. Below this is a breadcrumb trail: Oracle Technology Network > Java > Java SE > Downloads. The main content area is titled "Java SE Downloads" and features two large download buttons: "DOWNLOAD" for "Java Platform (JDK) 8u60" and "DOWNLOAD" for "NetBeans with JDK 8". A sidebar on the left lists various Java products and support links, while a sidebar on the right lists Java SDKs and Tools, Java Resources, and Java APIs. The bottom section of the page is titled "Java Platform, Standard Edition" and provides details about the "Java SE 8u60" release.

ORACLE

Sign In/Register Help Country Communities I am a... I want to... Search

Products Solutions Downloads Store Support Training Partners About OTN

Oracle Technology Network > Java > Java SE > Downloads

Overview Downloads Documentation

Java SE Downloads

Java SE  
Java EE  
Java ME  
Java SE Support  
Java SE Advanced & Suite  
Java Embedded  
Java DB  
Web Tier  
Java Card  
Java TV  
New to Java  
Community  
Java Magazine

Java Platform (JDK) 8u60

NetBeans with JDK 8

Java SDKs and Tools

- Java SE
- Java EE and Glassfish
- Java ME
- Java Card
- NetBeans IDE
- Java Mission Control

Java Resources

- Java APIs
- Technical Articles
- Demos and Videos
- Forums
- Java Magazine

Java Platform, Standard Edition

Java SE 8u60

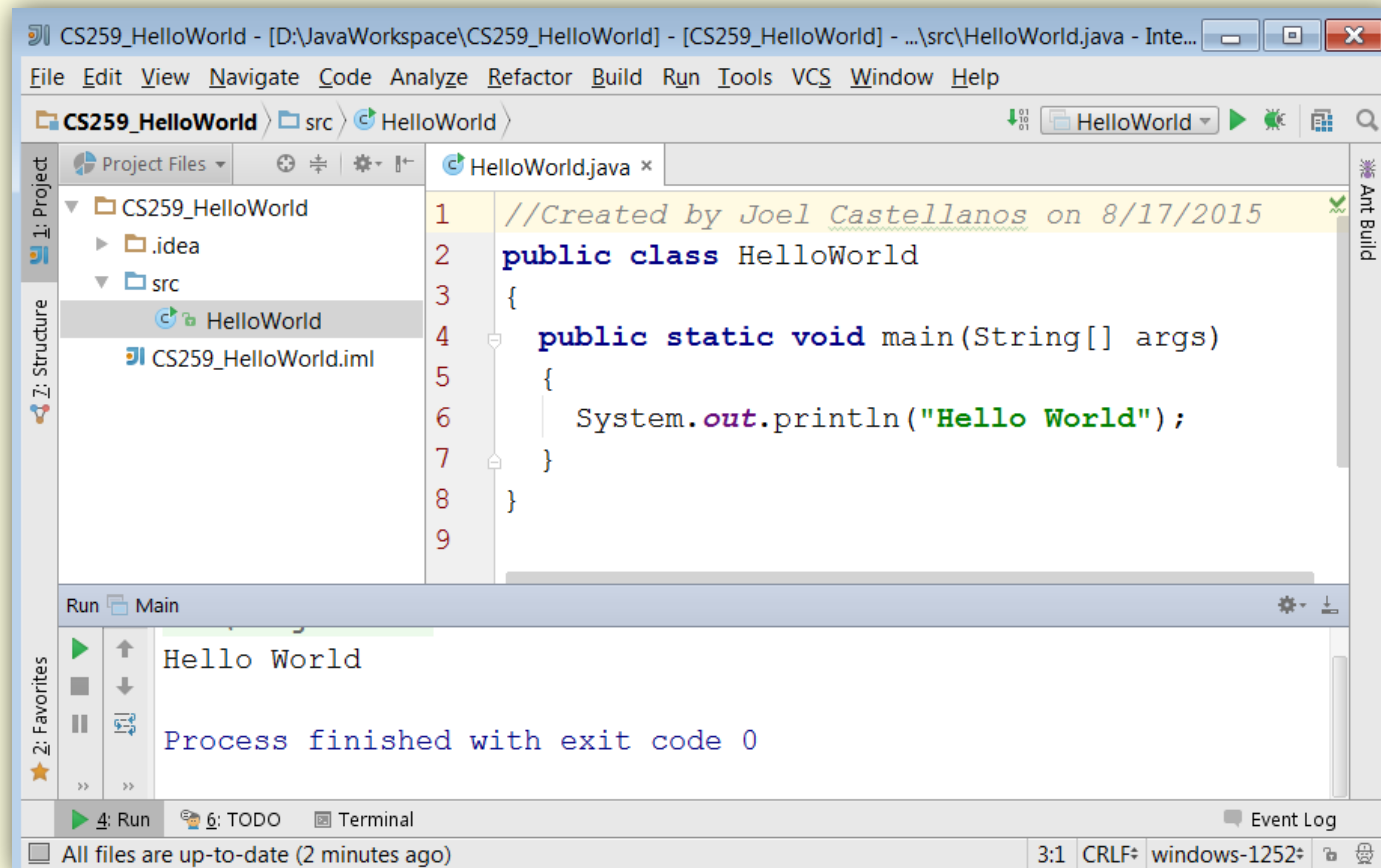
This releases includes support for ARMv8 processors, Nashorn enhancements, and improvements to Deployment Rule Set functionality

# IntelliJ IDE (Integrated Development Environment)

- <https://www.jetbrains.com/idea/download/>
- IntelliJ Community Edition (free) version 14.1.

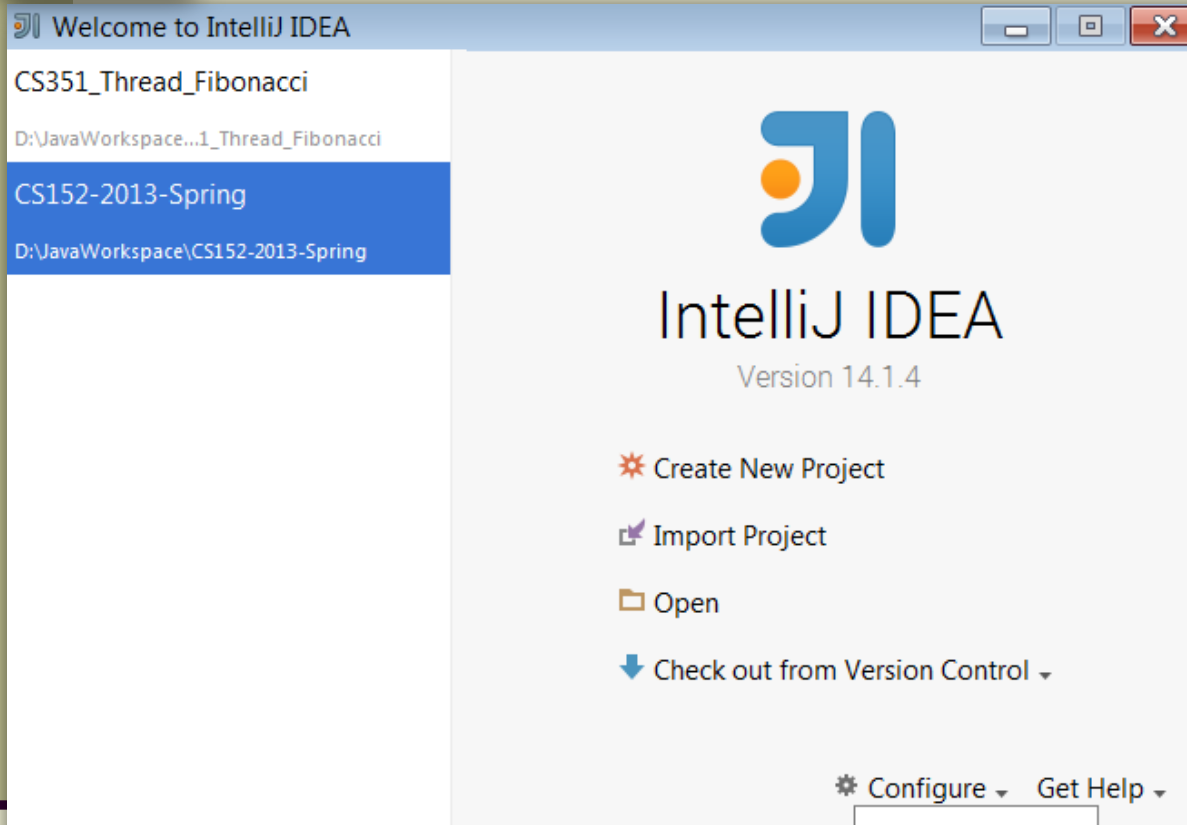
**Java** is a  
**Programming  
Language.**

**IntelliJ** is an  
**Integrated  
Development  
Environment.**





# Project Default Settings (Code Style)



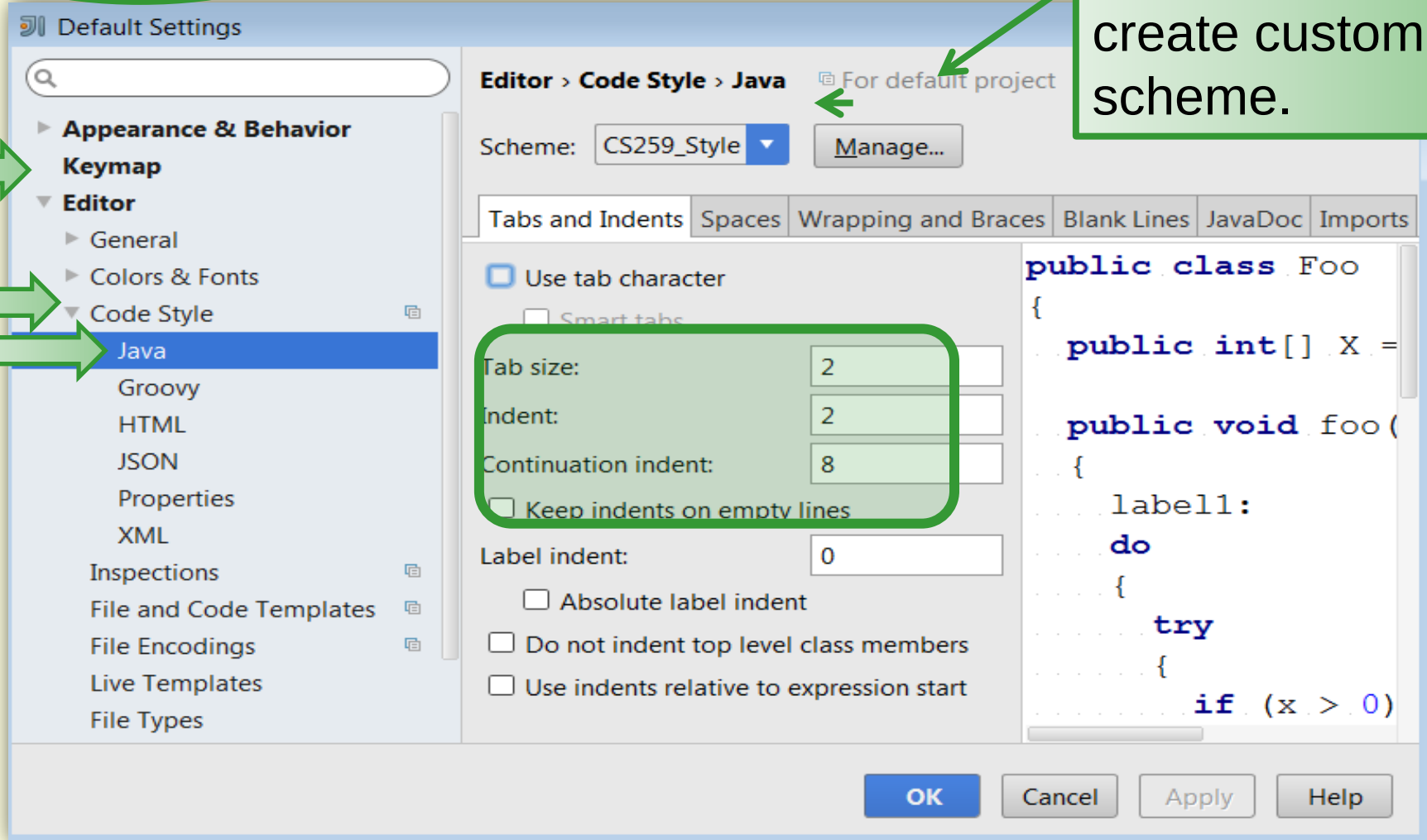
Use to change the code style and other settings that will be used for all new projects created.

⚙ Configure ▾ Get Help ▾

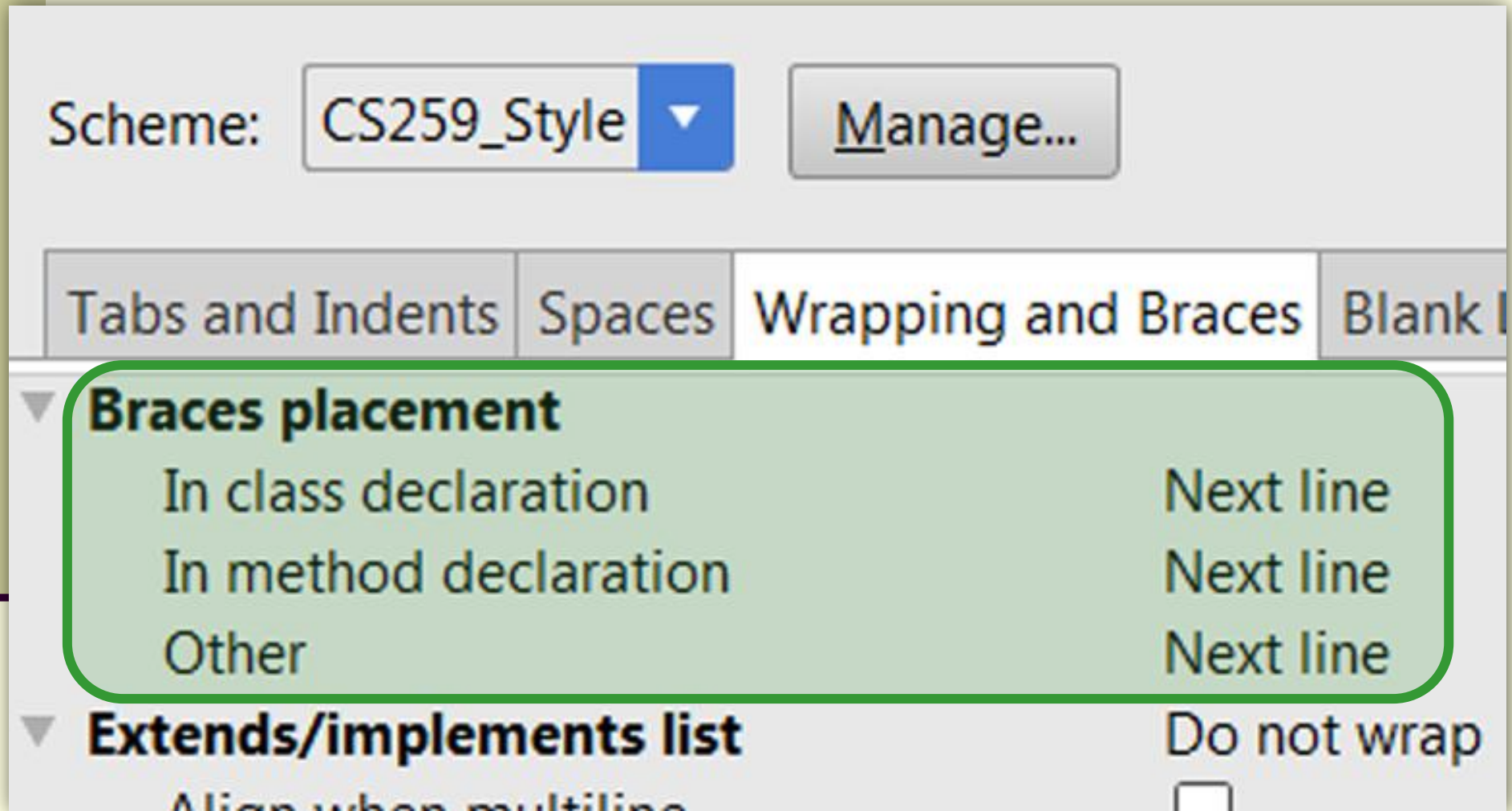
- Settings
- Plugins
- Import Settings
- Export Settings
- Check for Update
- Project Defaults ▸

- Settings
- Project Structure
- Run Configurations

# Code Style: IntelliJ → File → Settings...



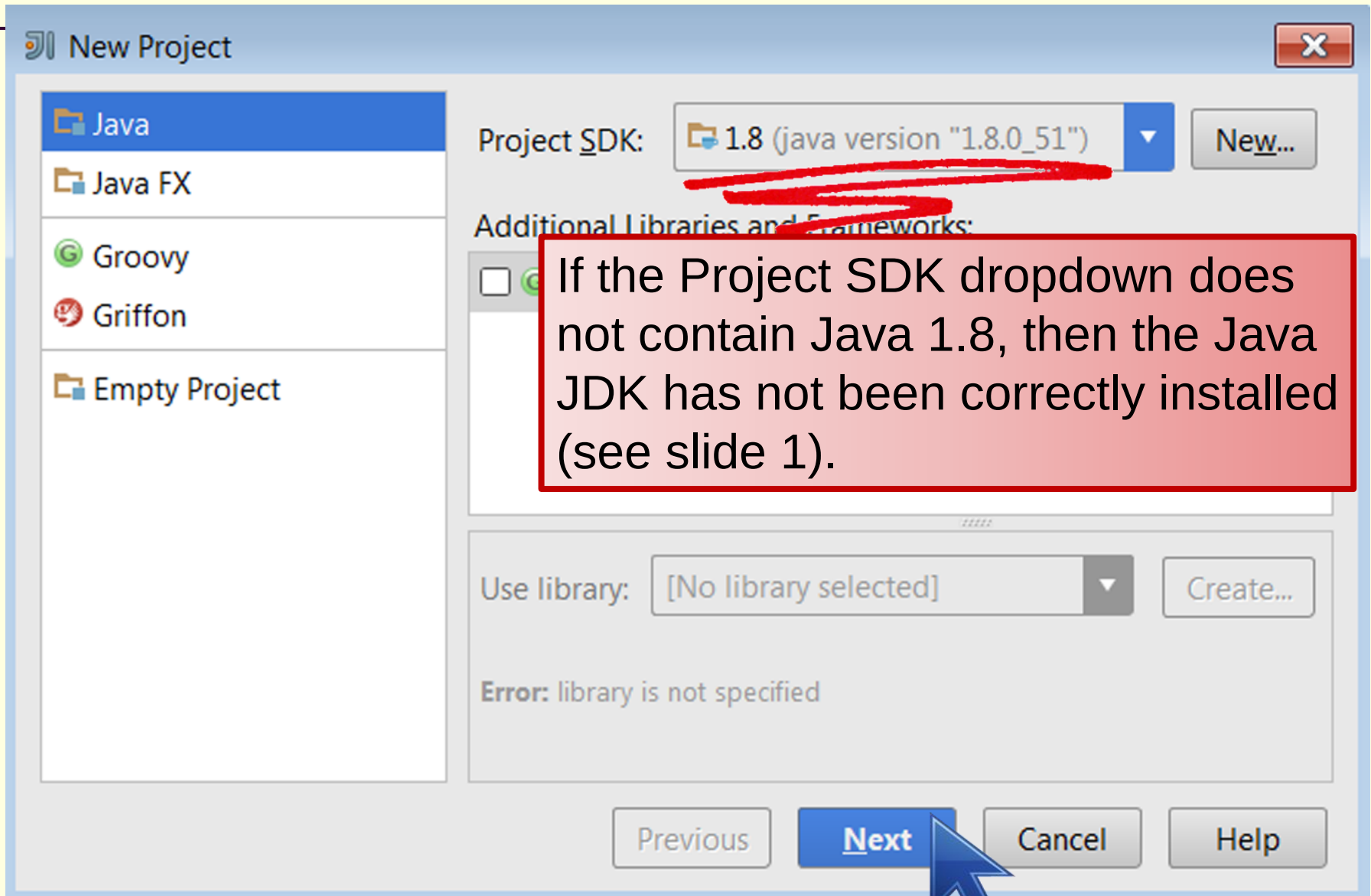
# Code Style: Braces



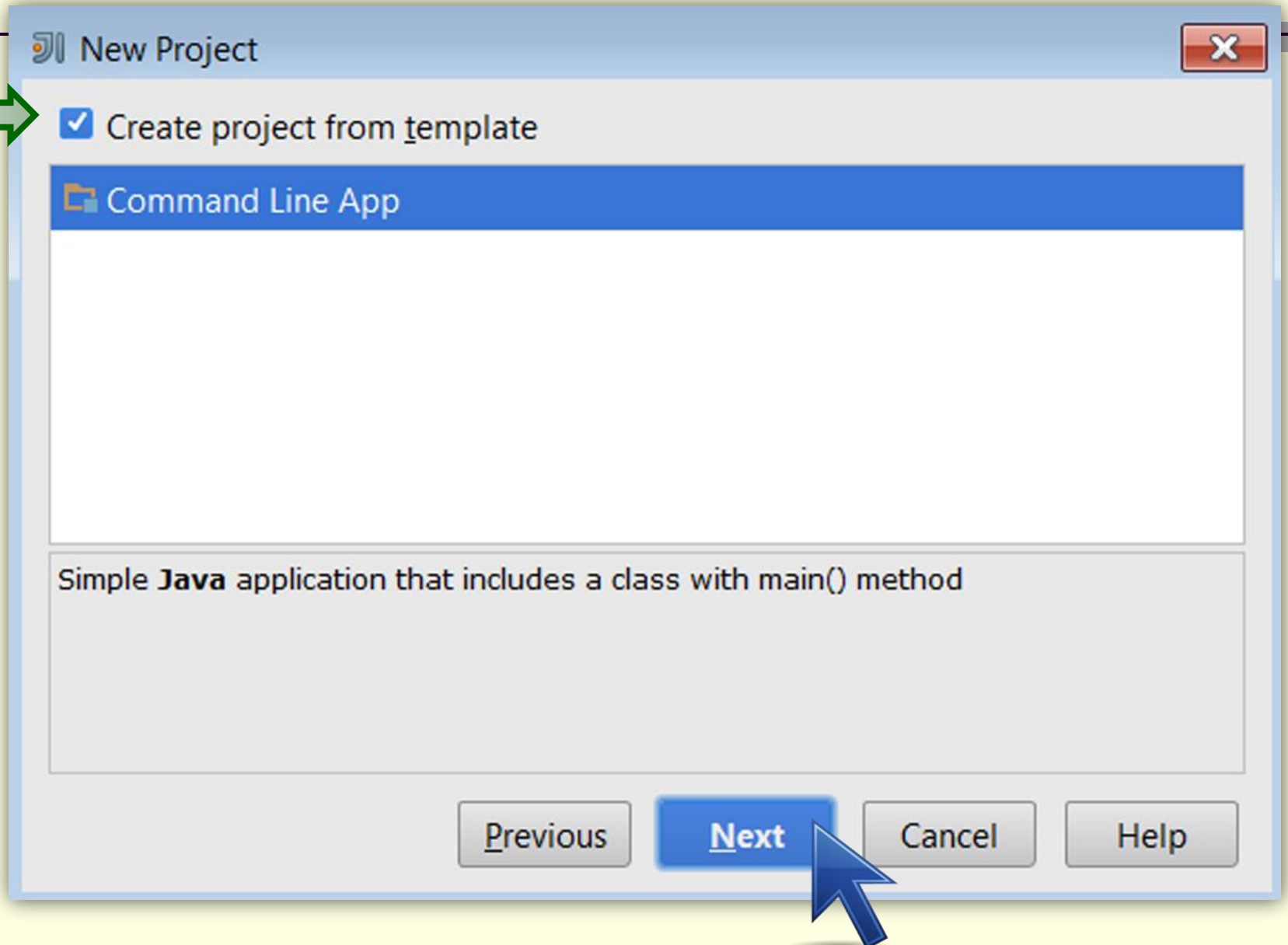
# IntelliJ: Create New Project (1 of 5)



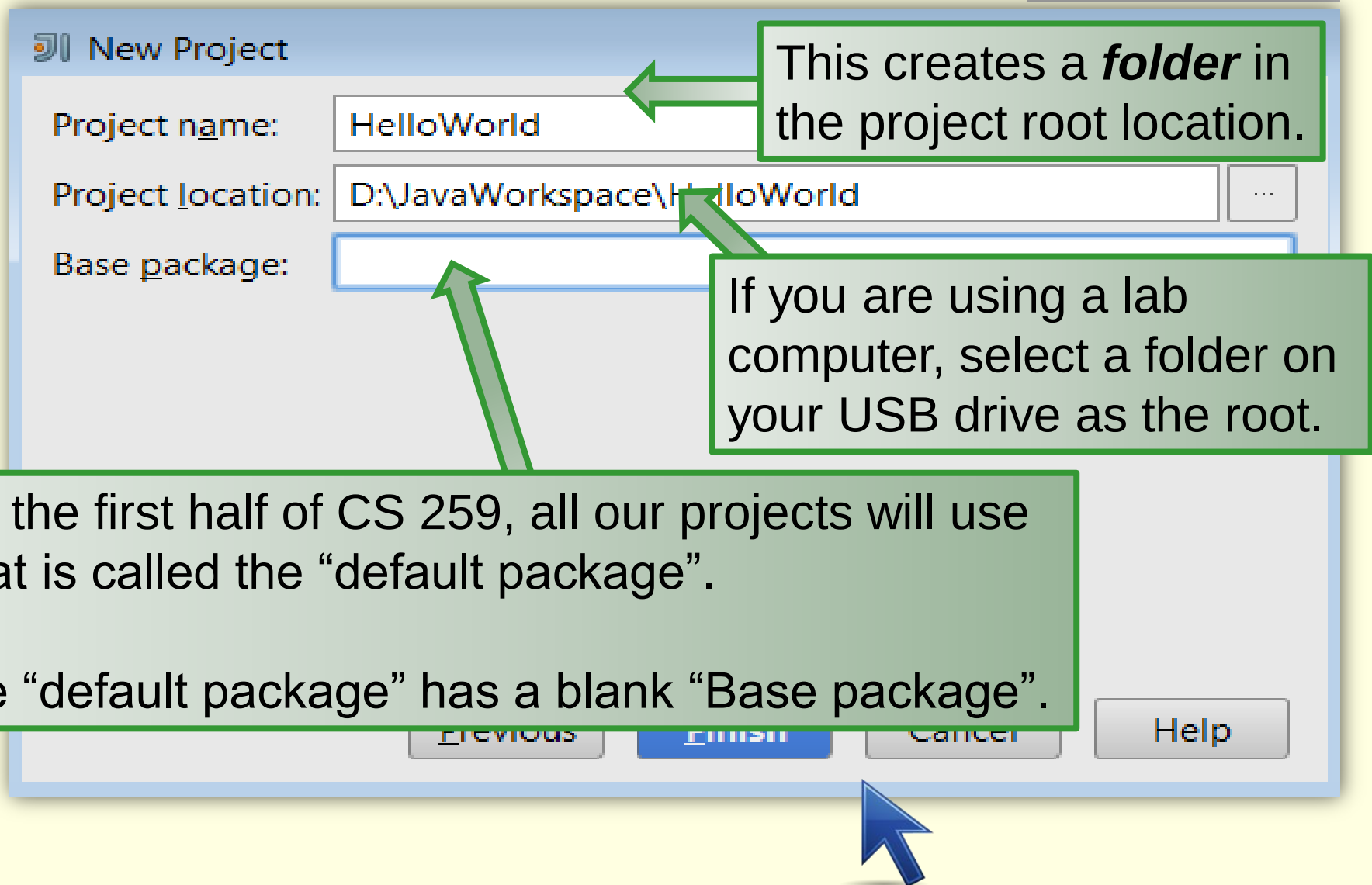
# IntelliJ: Create New Project (2 of 5)



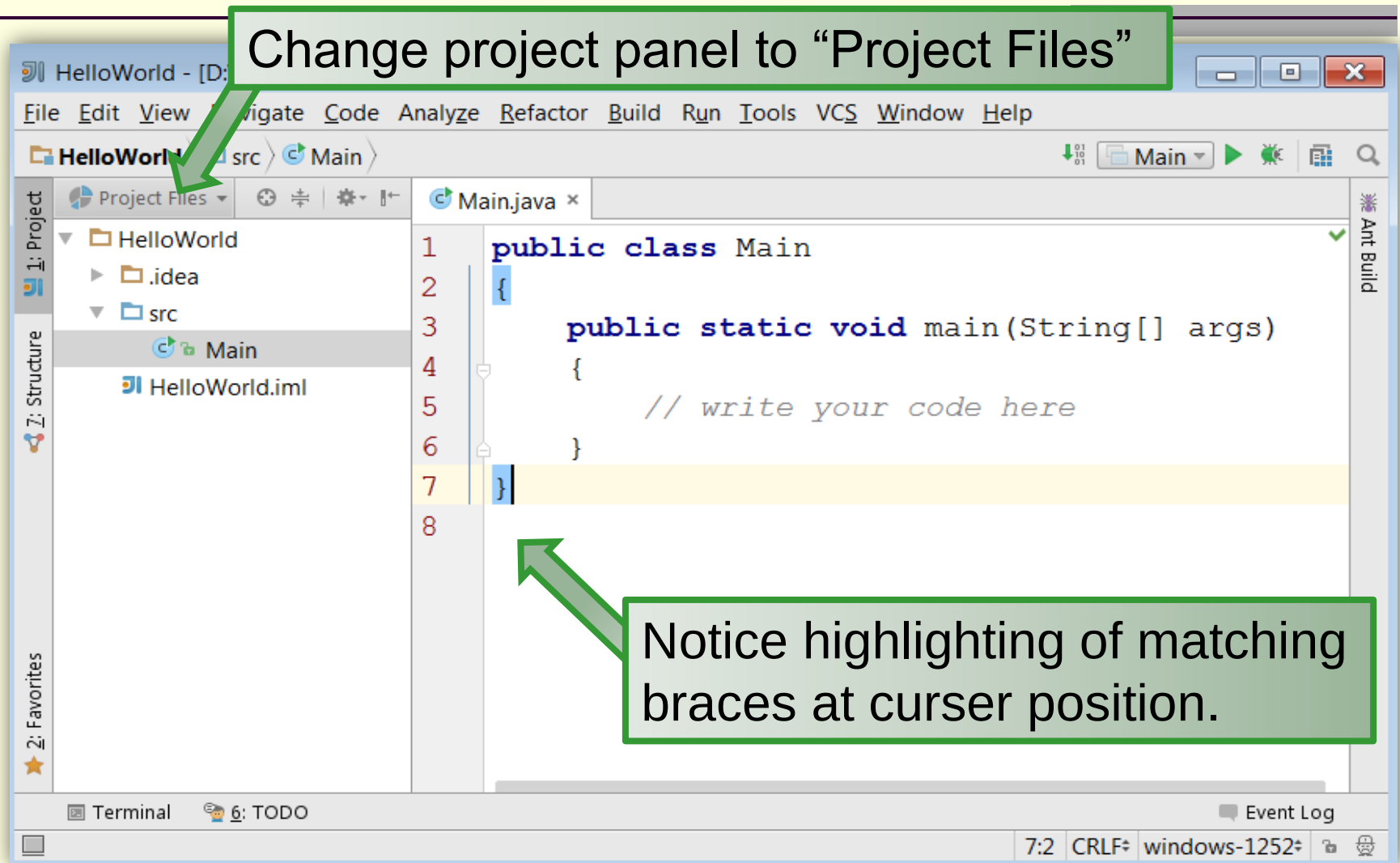
# IntelliJ: Create New Project (3 of 5)



# IntelliJ: Create New Project (4 of 5)



# IntelliJ: Create New Project (5 of 5)



# Application: Add Inputs (Similar to Listing 1.1)

```
1. import java.util.Scanner;
2.
3.
4. public class Toy
5. {
6.     public static void main(String[] args)
7.     {
8.         System.out.println("Hello World");
9.         System.out.println("I want two numbers!");
10.
11.         Scanner keyboard = new Scanner(System.in);
12.         int n1 = keyboard.nextInt();
13.         int n2 = keyboard.nextInt();
14.
15.         System.out.println(n1+n2);
16.     }
17. }
```

Filename ***MUST*** be Toy.java

From IntelliJ Project Files panel,  
Right-click filename and select:  
**Refactor → Rename**

# Compile, Run, and the Console

```
1. System.out.println("Hello World");
2. System.out.println("I want two numbers!");
3.
4. Scanner keyboard = new Scanner(System.in);
5. int n1 = keyboard.nextInt(); //reads characters until ↵
6. int n2 = keyboard.nextInt();
7.
8. System.out.println(n1+n2);
```



Compile  
and Run

```
Console X
Hello World
I want two numbers!
23
7
30
```

# Keyword: `import`

```
1. import java.util.Scanner;
```

```
2.
```

```
3.
```

```
4.
```

```
5.
```

```
6.
```

```
7.
```

```
8. public class Toy_1_1
```

```
9. {
```

```
10.   public static void main(String[] args)
```

```
11.   {
```

```
12.       System.out.print
```

```
13.
```

Gets the `Scanner` class from the *package* (library) `java.util`.

All import statements are placed at the top of the source file.

**Note:** The Java programming language is *case-sensitive*.

# Keyword: class

```
1. import java.util.Scanner;
```

```
2.
```

```
3. public class Toy
```

```
4. {
```

```
5.     publ
```

```
6.     {
```

```
7.         Sy
```

```
8.         Sy
```

```
9.
```

```
10.         Scanner keyboard = new Scanner(System.in);
```

```
11.         in
```

```
12.         in
```

```
13.
```

```
14.         Sy
```

```
15.     }
```

```
16. }
```

## **class header.**

In Java, all code is enclosed in a class.

Class names are your choice but.... *should* start with an upper-case letter.

The **class header** must be followed by an **open curly bracket**, {, and ended with a matching **close curly bracket**, }.

All code within those brackets is part of the class.

# Method Signature

```
1. import java.util.Scanner;
2.
3. public class Toy
4. {
5.     public static void main(String[] args)
6.     {
7.         Sy
8.         Sy
9.
10.        Sc
11.        in
12.        in
13.
14.        Sy
15.    }
16. }
```

Line 5 is a *method signature*.

Every Java application must have a **main** method.

By convention, method names start with a lower-case letter.

A method signature must be followed by an opening curly bracket '{' and a matching closing curly bracket '}'.

Code within the brackets is part of the method.

# Keyword: `public`

```
1. import java.util.Scanner;  
2.  
3. public class Toy  
4. {  
5.     public static void main(String[] args)  
6.     {  
7.         System.out.println("Hello World");  
8.  
9.  
10.  
11.  
12.  
13.  
14.
```

The keyword `public` means that the *class*, *field* (variable), or *method* is *visible* to other classes.

The outermost class in each Java source file must be `public`.

The `main` method must be `public`.

# Parts of a Method Signature

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

`static`  
*Discussed later.*

The method's *argument list* must be enclosed in parenthesis: ().

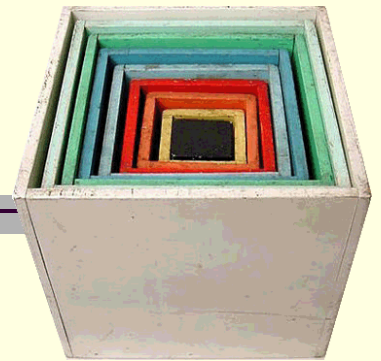
`main` requires one *argument* : `args` which is a *field* of *type* `String[]` (an *array* of `String` *objects*).

Method's *return type* (`void`, `int`, `float`, `String`, ...)

The *return type* is not part of the *method signature*.

**Public:** The method is *visible* outside the class.

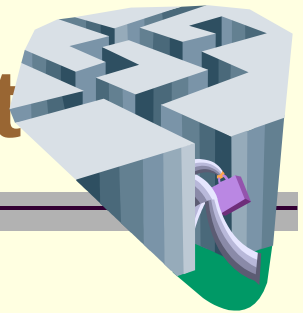
# Block Structure



```
1. public class Toy
2. {
3.     // Comment inside Toy.
4.     public static void main(String[] args)
5.     {
6.         // Comment inside main which is inside Toy.
7.         System.out.println("Hello World");
8.     }
9. }
```

The *nesting* of curly brackets defines the block structure. Good programming *style* requires block indentation. In CS-259, we will use exactly two spaces per level.

# Program Execution Entry Point



```
1. public class Toy
2. {
3.     public static void main(String[] args)
4.     {
5.
```

When a Java program runs, code *execution* starts at **main**.

Every Java *application* must have **at least one** class with a **main** method.

It is ok for an application to have more than one class with **main**.

When a user runs a Java application, the user must specify which class contains the **main** he or she wants to use.

# Method Calls



1. `System.out.println("Hello World");`
2. `System.out.println("I want two numbers!");`

Line 1 ***Calls*** the `println` ***method*** of the ***object*** `System.out` with the ***argument*** `"Hello World"`.

The `println` ***method*** takes a `String` ***object*** for an ***argument*** (input).

The `println` ***method*** displays its argument in the Java Console and then displays a newline character, `'/n'`.

# Keyword **new**: Instantiating a Class

```
Scanner keyboard = new Scanner(System.in);
```

*Declares* a *field* (with the programmer defined name **keyboard**) to be of *type* **Scanner**.

The *type* **Scanner** is defined in **import java.util.Scanner**

*argument list*

Creates an *instance* of the **Scanner** class.

The = symbol *assigns* **keyboard** to the *memory address* of the newly created *instance* of **Scanner**.

# Reading User Input from the Keyboard

```
1. Scanner keyboard = new Scanner(System.in) ;  
2. int n1 = keyboard.nextInt() ;
```

Line 2 does many things:

- 1) **Defines** `n1` to be a **field** of **type** `int`.
- 2) **Allocates** memory for `n1`.
- 3) **Calls** the `nextInt` **method** of the keyboard **instance** of `Scanner`. The `nextInt` **method**:
  - i. **Reads** input from the keyboard.
  - ii. **Echoes** the input to the **Console**.
  - iii. **Tries to convert** the user input to an `int`.
  - iv. **Returns** the converted input.
- 4) **Assigns** the returned `int` value to `n1`.

End of Statement

;

Start of Block

{

1. `import java.util.Scanner;`

2.

3. `public class Toy`

4. `{`

5.  `public static void main(String[] args)`

6. `{`

7.  `System.out.println("Hello World");`

8.  `System.out.println("I want 2 numbers!");`

9.

10.  `Scanner keyboard = new Scanner(System.in);`

11.  `int n1 = keyboard.nextInt();`

12.  `int n2 = keyboard.nextInt();`

13.

14.  `System.out.println(n1+n2);`

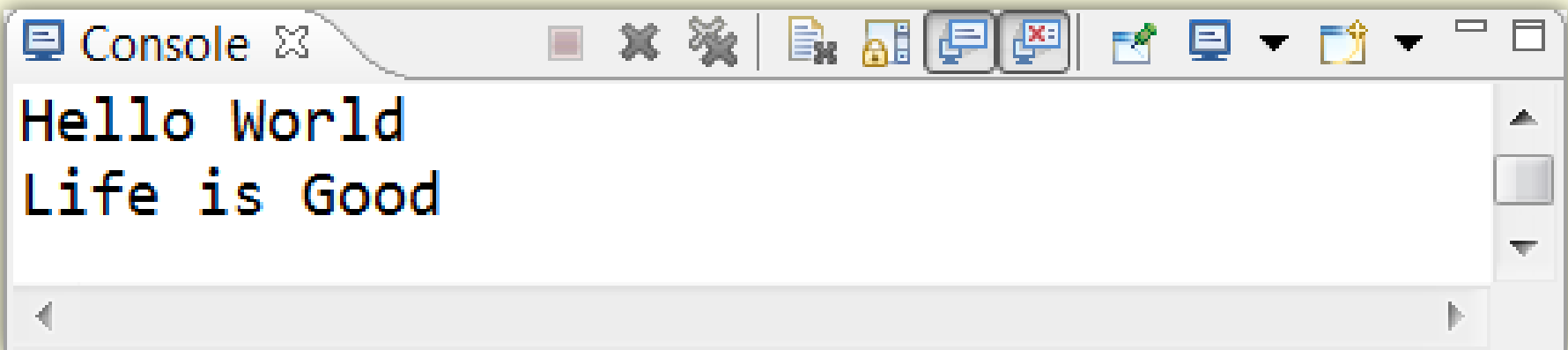
15. `}`

16. `}`

# Whitespace: Ignored Between Identifiers

```
1. System.out.println("Hello World");
2.
3. System.    out.
4.     println (
5.         "Life is Good"
6.     );
```

Single *statement* split  
across multiple lines.



The screenshot shows a Java IDE's console window. The title bar reads 'Console'. The output area displays two lines of text: 'Hello World' and 'Life is Good', each on a new line. The console has a standard toolbar with icons for running, debugging, and other IDE functions. A scrollbar is visible on the right side of the output area.

# Error: Whitespace Within Identifiers

Toy.java x

```
import java.util.Scanner;

public class Toy {
    public static void main(String[] args) {
        System.o ut.println("Hello World");
        println("I want two number");

        Scanner keyboard = new Scanner(System.in);
        int n1 = keyboard.nextInt();
        int n2 = keyboard.nextInt();

        System.out.println(n1 + n2);
    }
}
```

One extra space, causes 6 errors!!

- ';' expected
- Cannot resolve method 'println(java.lang.String)'
- Cannot resolve symbol 'o'
- Unexpected token
- Variable 'ut' is never used
- System.o ut.println("Hello World");

Remove variable 'ut' ▶

Tabs on scrollbar are hyperlinks to error locations.

IntelliJ's suggestions can sometimes save typing, but often are incorrect.

# Find and Fix the Syntax Error: #1

```
1. import java.util.Scanner;
2.
3. public class Toy
4. {
5.     public static void main(String[] args)
6.     {
7.         System.out.println("Input 2 numbers");
8.
9.         Scanner in = new Scanner(System.in);
10.        int n1 = keyboard.nextInt();
11.        int n2 = keyboard.nextInt();
12.
13.        System.out.println(n1+n2);
14.    }
15. }
```

# Find and Fix the Syntax Error: #2

```
1. import java.util.Scanner;
2.
3. public class Toy
4. {
5.     public static void main(String[] args)
6.
7.         System.out.println("Input 2 numbers");
8.
9.         Scanner bob = new Scanner(System.in);
10.        int n1 = bob.nextInt();
11.        int n2 = bob.nextInt();
12.
13.        System.out.println(n1+n2);
14.    }
15. }
```

# Find and Fix the Syntax Error: #3

```
1. import java.util.Scanner;
2.
3. public class Toy
4. {
5.     public static void main(String[] args) ;
6.     {
7.         System.out.println("Input 2 numbers") ;
8.
9.         Scanner in = new Scanner(System.in) ;
10.        int n1 = in.nextInt() ;
11.        int n2 = in.nextInt() ;
12.
13.        System.out.println(n1+n2) ;
14.    }
15. }
```

# Setting IntelliJ to Project Files View

