CS 259: Data Structures with Java
Hello World with the IntelliJ IDE

Instructor: **Joel Castellanos**
e-mail: joel.unm.edu

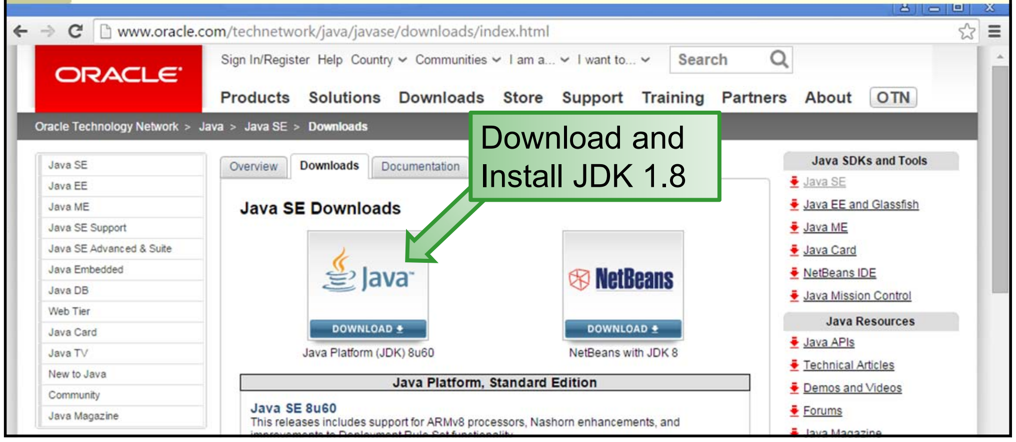
Web: http://cs.unm.edu/~joel/
Office: Electrical and Computer Engineering building (ECE). Room 233

9/9/2016

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.



Java Platform, Standard Edition

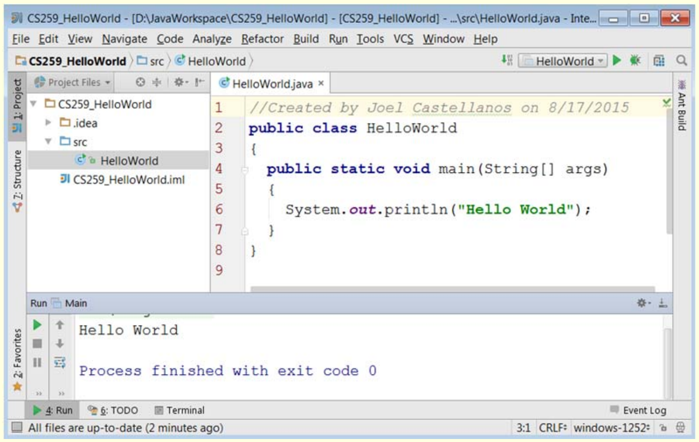
Java SE 8u60
This releases includes support for ARMv8 processors, Nashorn enhancements, and

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***.


CS259_HelloWorld - [D:\JavaWorkspace\CS259_HelloWorld] - [CS259_HelloWorld] - ...\src\HelloWorld.java - Inte...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
CS259_HelloWorld src HelloWorld
HelloWorld
Project Files
CS259_HelloWorld
.idea
src
HelloWorld
CS259_HelloWorld.iml
HelloWorld.java
1: Project
7: Structure
2: Favorites
Ant Build
Run Main
4: Run
6: TODO
Terminal
Event Log
3:1 CRLF windows-1252

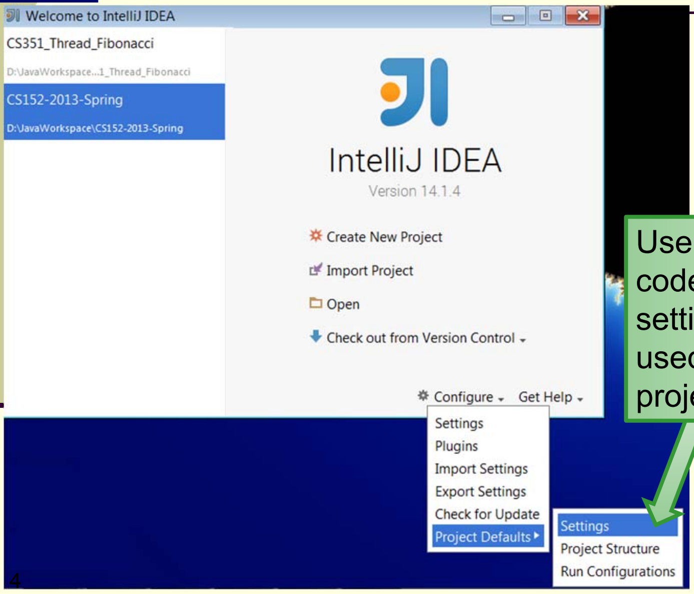
```java
//Created by Joel Castellanos on 8/17/2015
public class HelloWorld
{
  public static void main(String[] args)
  {
    System.out.println("Hello World");
  }
}
```

```
Hello World

Process finished with exit code 0
```

All files are up-to-date (2 minutes ago)

3

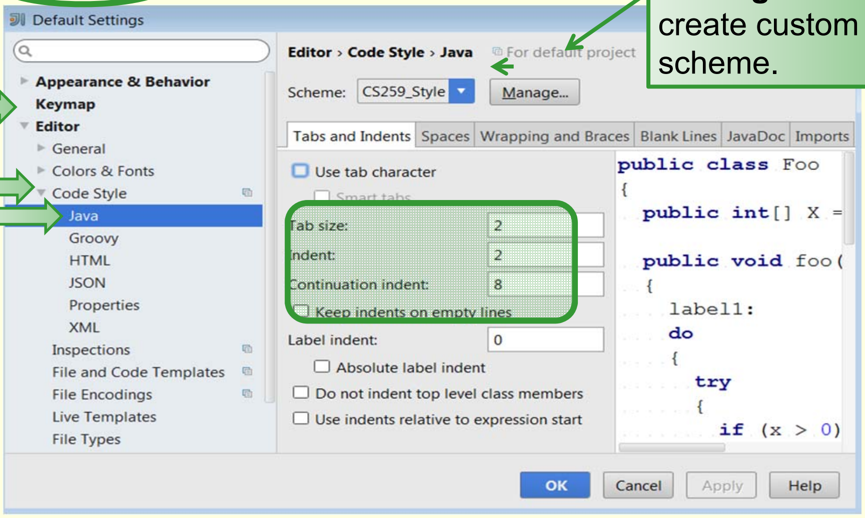
Project Default Settings (Code Style)



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

4

Code Style: IntelliJ → File → Settings…



Scheme: CS259_Style

- [ ] Use tab character
- [ ] Smart tabs

Tab size: 2
Indent: 2
Continuation indent: 8
- [ ] Keep indents on empty lines

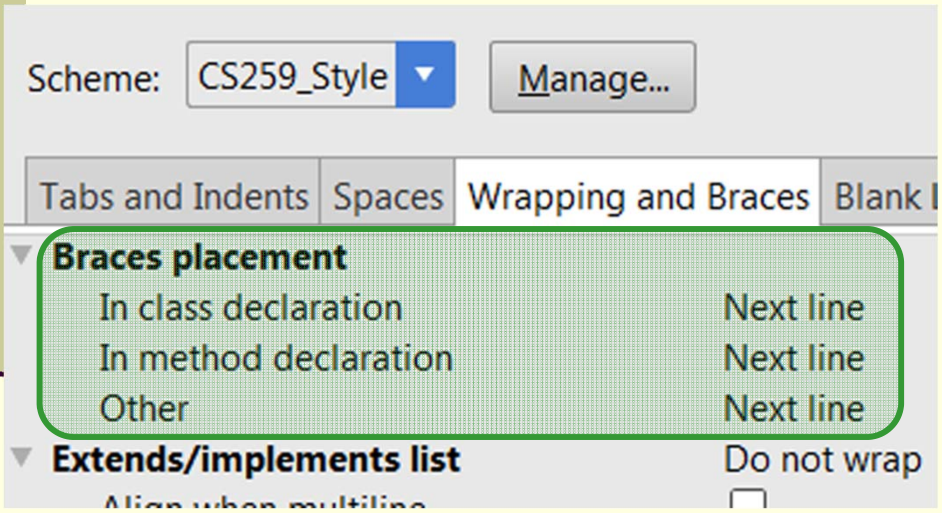
Label indent: 0
- [ ] Absolute label indent
- [ ] Do not indent top level class members
- [ ] Use indents relative to expression start

```
public class Foo
{
  public int[] X =

  public void foo(
  {
    label1:
    do
    {
      try
      {
        if (x > 0)
```

"**Manage…**" to create custom scheme.

5

Code Style: Braces



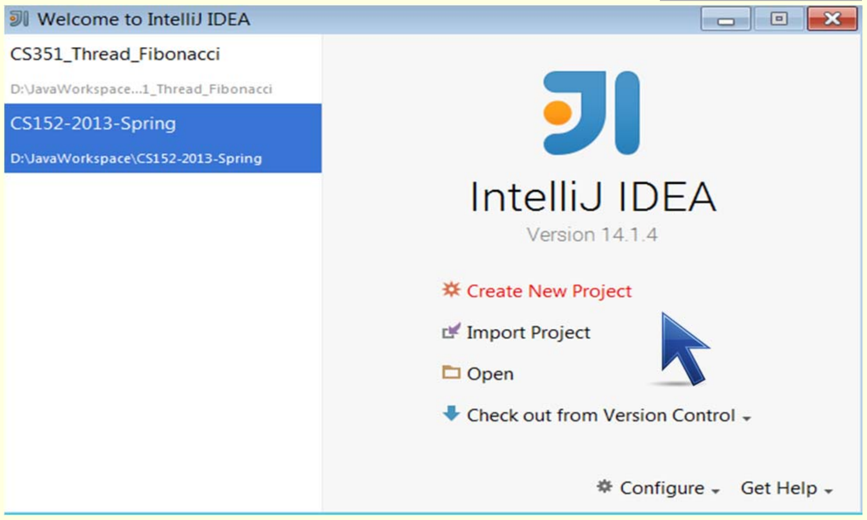
Scheme: CS259_Style

Braces placement

In class declaration	Next line
In method declaration	Next line
Other	Next line
Extends/implements list	Do not wrap
Align when multiline	☐

6

IntelliJ: Create New Project (1 of 5)



Welcome to IntelliJ IDEA

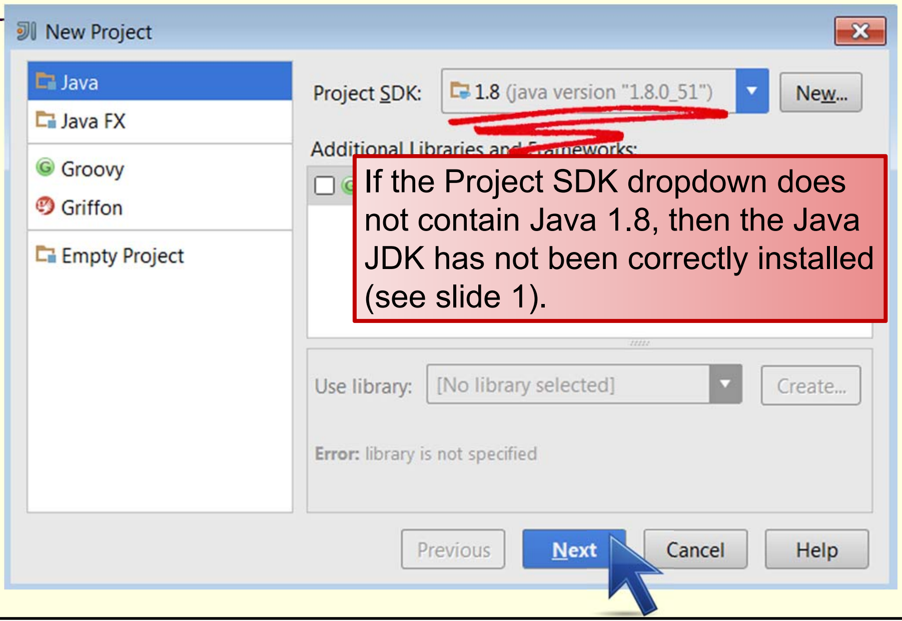
CS351_Thread_Fibonacci
D:\JavaWorkspace...1_Thread_Fibonacci

CS152-2013-Spring
D:\JavaWorkspace\CS152-2013-Spring

IntelliJ IDEA
Version 14.1.4

7

IntelliJ: Create New Project (2 of 5)



New Project

Project SDK: 1.8 (java version "1.8.0_51")

Additional Libraries and Frameworks:

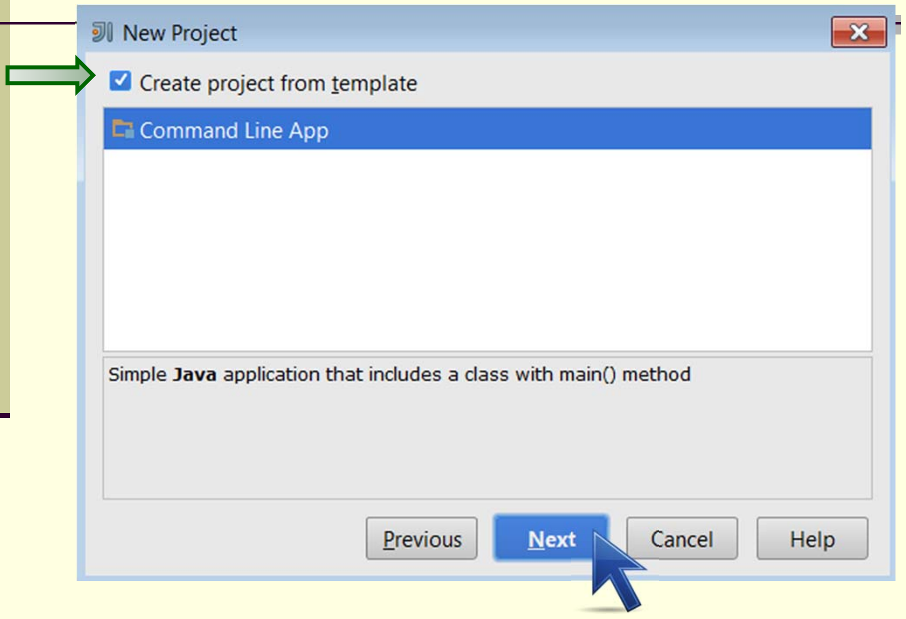
Use library: [No library selected]

Error: library is not specified

If the Project SDK dropdown does not contain Java 1.8, then the Java JDK has not been correctly installed (see slide 1).

8

IntelliJ: Create New Project (3 of 5)



New Project

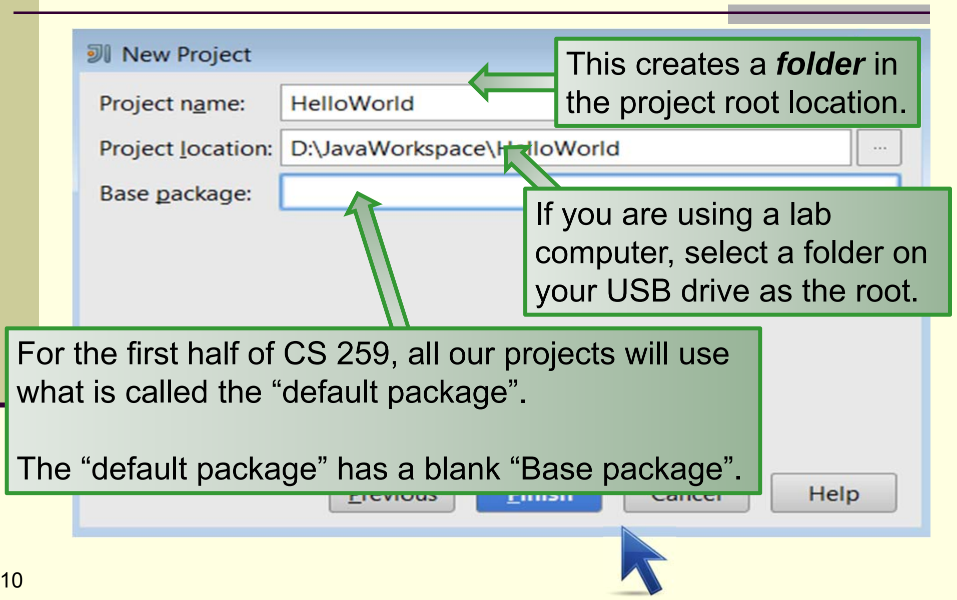
Create project from template: ✓

Command Line App

Simple **Java** application that includes a class with main() method

9

IntelliJ: Create New Project (4 of 5)



New Project

Project name: HelloWorld

Project location: D:\JavaWorkspace\HelloWorld

Base package:

This creates a ***folder*** in the project root location.

If you are using a lab computer, select a folder on your USB drive as the root.

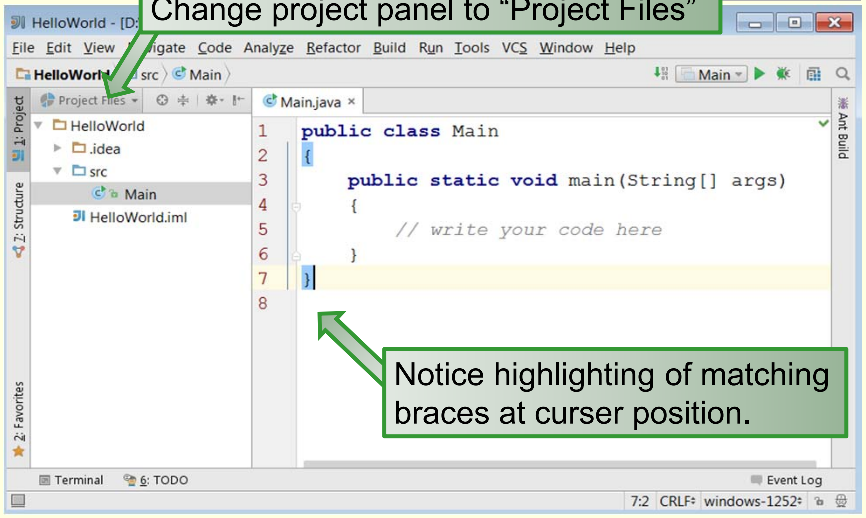
For the first half of CS 259, all our projects will use what is called the "default package".

The "default package" has a blank "Base package".

10

IntelliJ: Create New Project (5 of 5)

Change project panel to “Project Files”



```
public class Main
{
    public static void main(String[] args)
    {
        // write your code here
    }
}
```

Notice highlighting of matching braces at curser position.

Application: Add Inputs (Similar to Listing 1.1)

```
import java.util.Scanner;


public class Toy
{
  public static void main(String[] args)
  {
    System.out.println("Hello World");
    System.out.println("I want two numbers!");

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

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

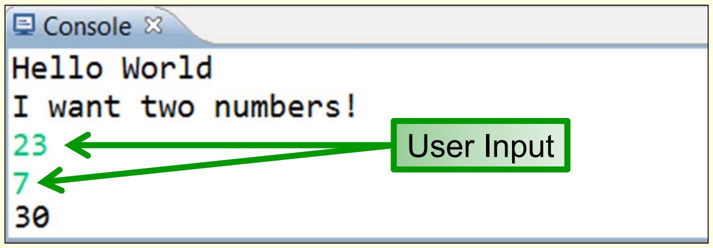
Filename ***MUST*** be **Toy.java**

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

Compile, Run, and the Console

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

Scanner keyboard = new Scanner(System.in);
int n1 = keyboard.nextInt(); //reads characters until ↵
int n2 = keyboard.nextInt();

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

Compile and Run



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

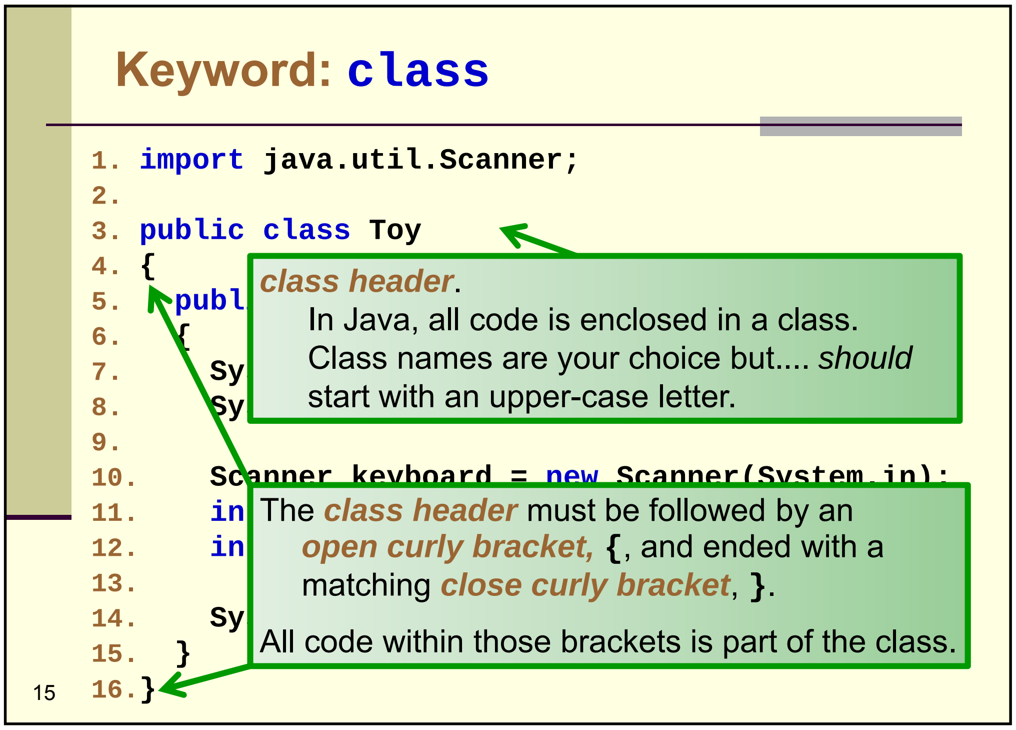
Keyword: `import`

```
import java.util.Scanner;






public class Toy_1_1
{
  public static void main(String[] args)
  {
    System.out.pri

```

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*.



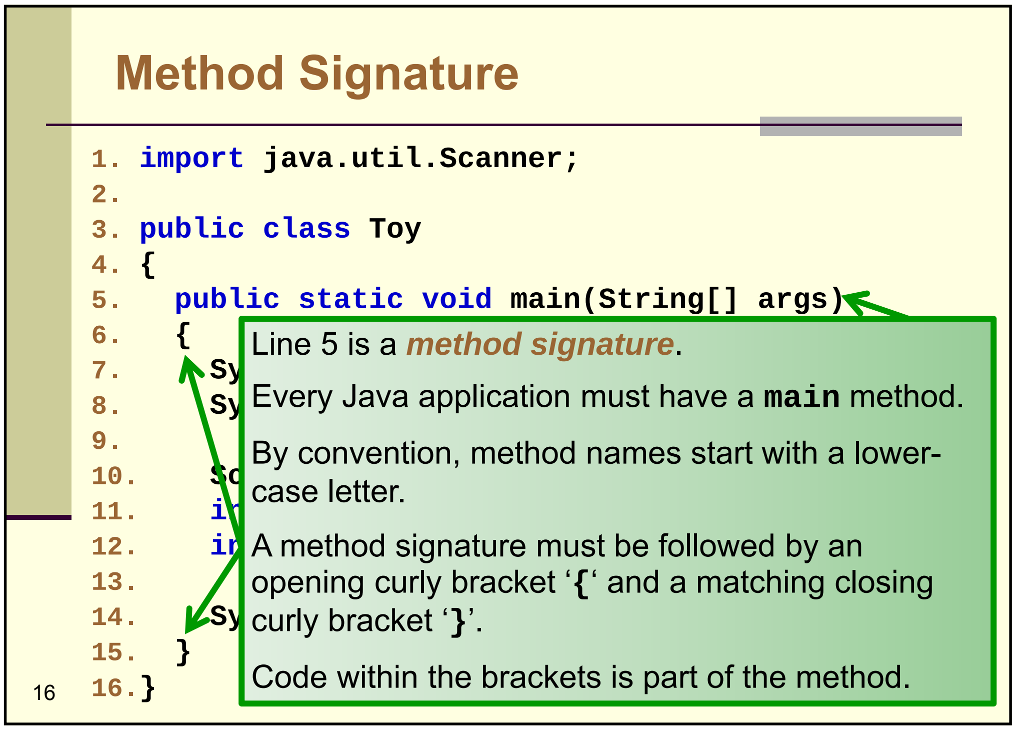
Keyword: `class`

```
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.    i
12.    in
13.
14.    Sy
15.  }
16.}
```

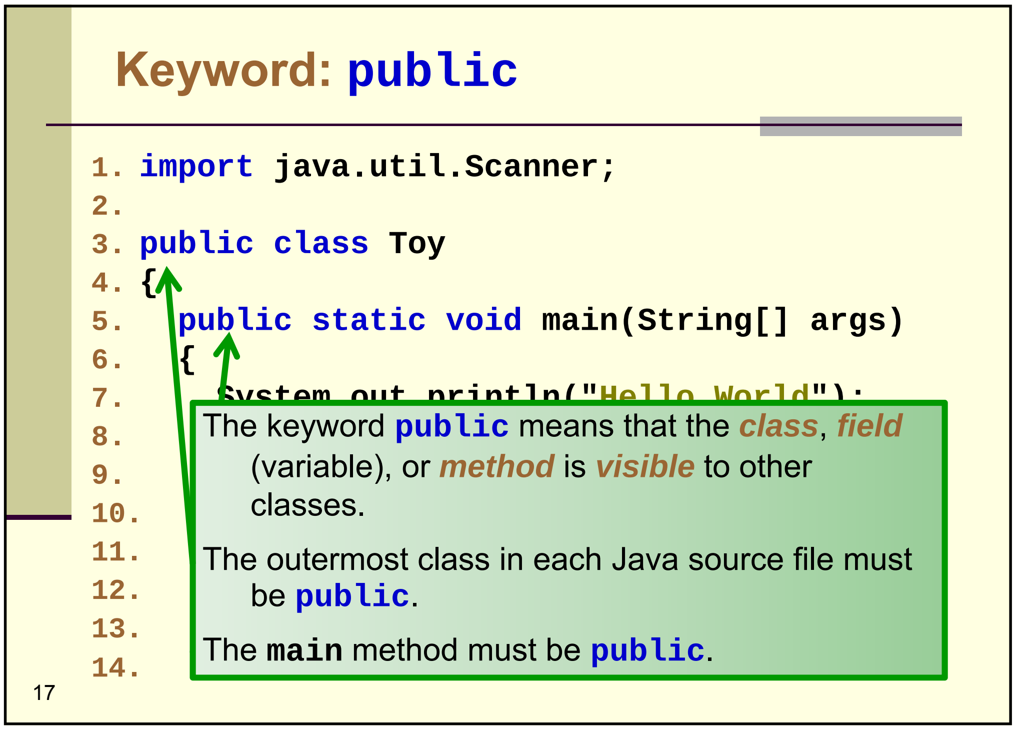
Line 5 is a *method signature*.

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.



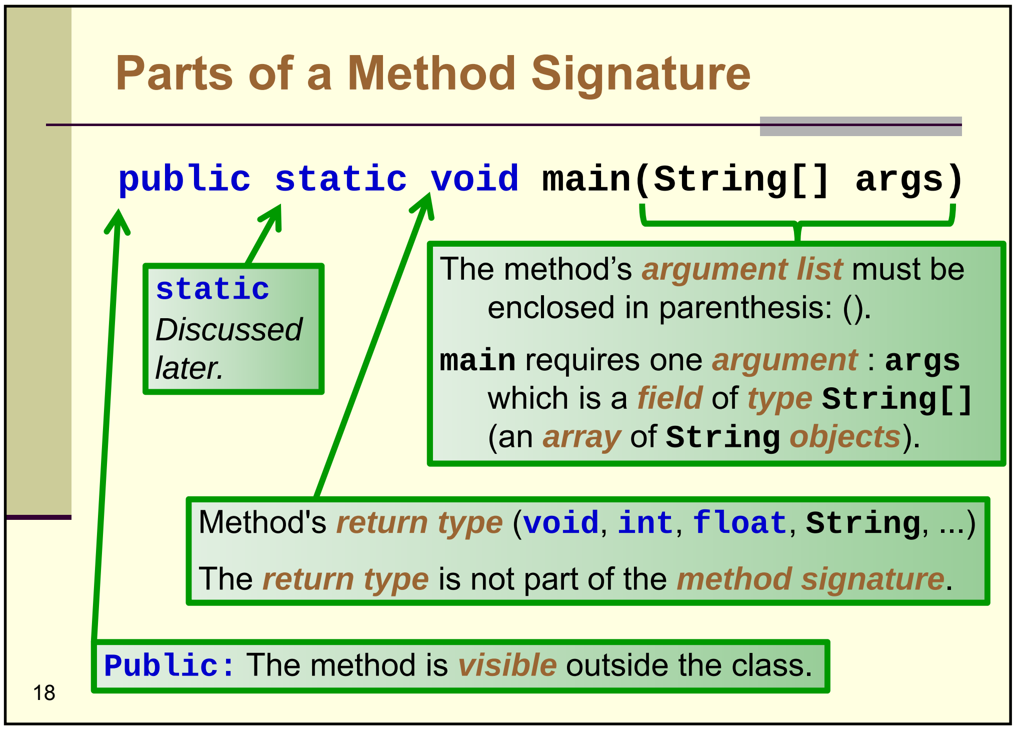
Keyword: public

```
import java.util.Scanner;

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

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: ().

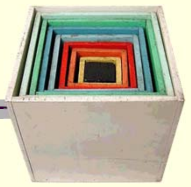
main requires one *argument* : **args** which is a *field* of *type* **String[]** (an *array* of **String** *objects*).

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



```
public class Toy
{
  // Comment inside Toy.
  public static void main(String[] args)
  {
    // Comment inside main which is inside Toy.
    System.out.println("Hello World");
  }
}
```

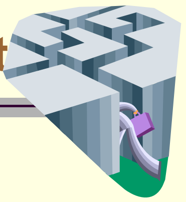
The *nesting* of curly brackets defines the block structure.

Good programming *style* requires block indentation.

In CS-259, we will use exactly two spaces per level.

19

Program Execution Entry Point



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

```

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.

20

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

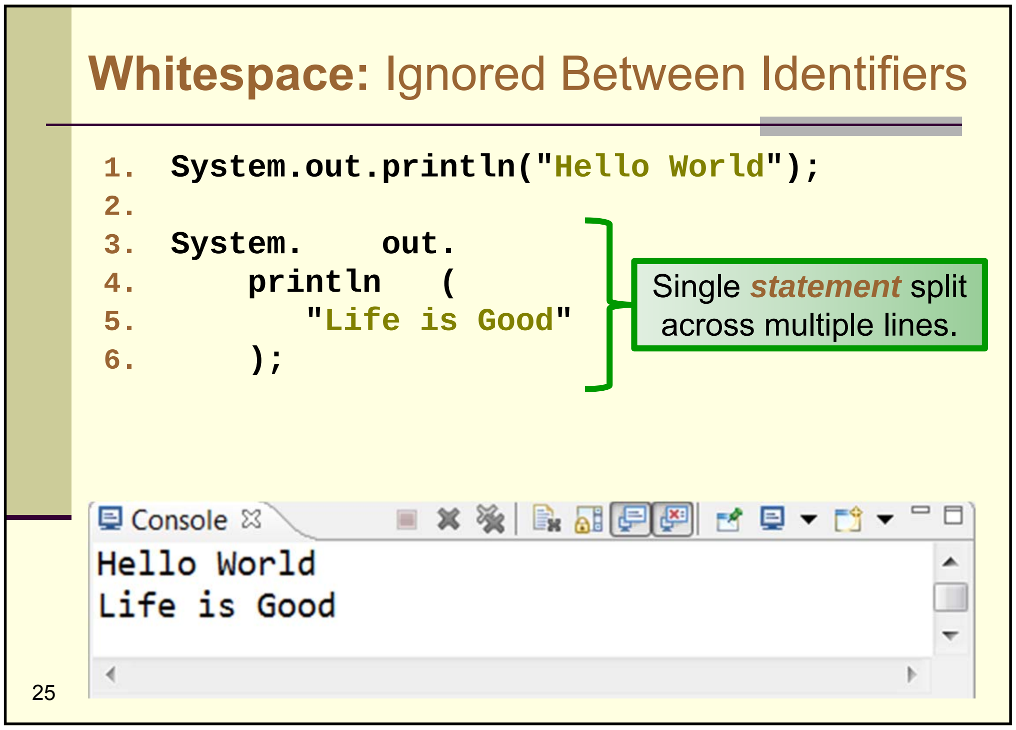
```
1. Scanner keyboard = new Scanner(System.in);
2. int n1 = keyboard.nextInt();
```

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 $\boxed{;}$ Start of Block $\boxed{\{}$

```
import java.util.Scanner;

public class Toy
{
  public static void main(String[] args)
  {
    System.out.println("Hello World");
    System.out.println("I want 2 numbers!");

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

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

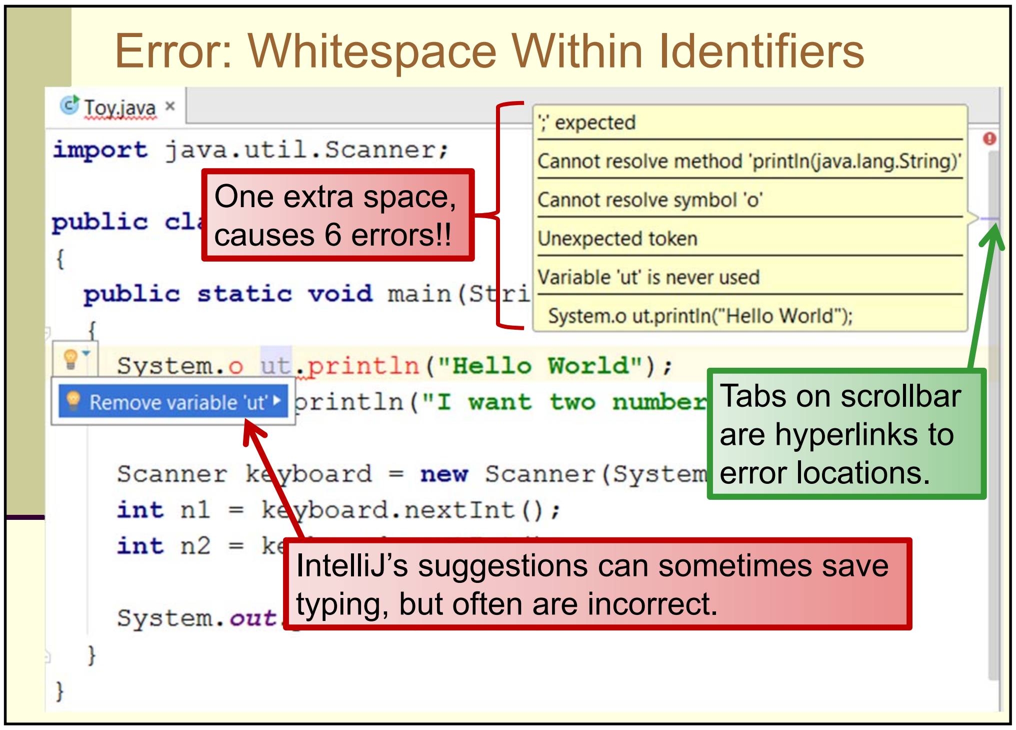


Whitespace: Ignored Between Identifiers

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

System.    out.
     println    (
         "Life is Good"
     );
```

Single *statement* split across multiple lines.

```
Hello World
Life is Good
```



Error: Whitespace Within Identifiers

```
import java.util.Scanner;

public cla
{
   public static void main(Stri
   {
      System.o ut.println("Hello World");
               println("I want two number

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

      System.out.
   }
}
```

```
';' 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");
```

One extra space, causes 6 errors!!

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

```java
import java.util.Scanner;

public class Toy
{
  public static void main(String[] args)
  {
    System.out.println("Input 2 numbers");

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

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

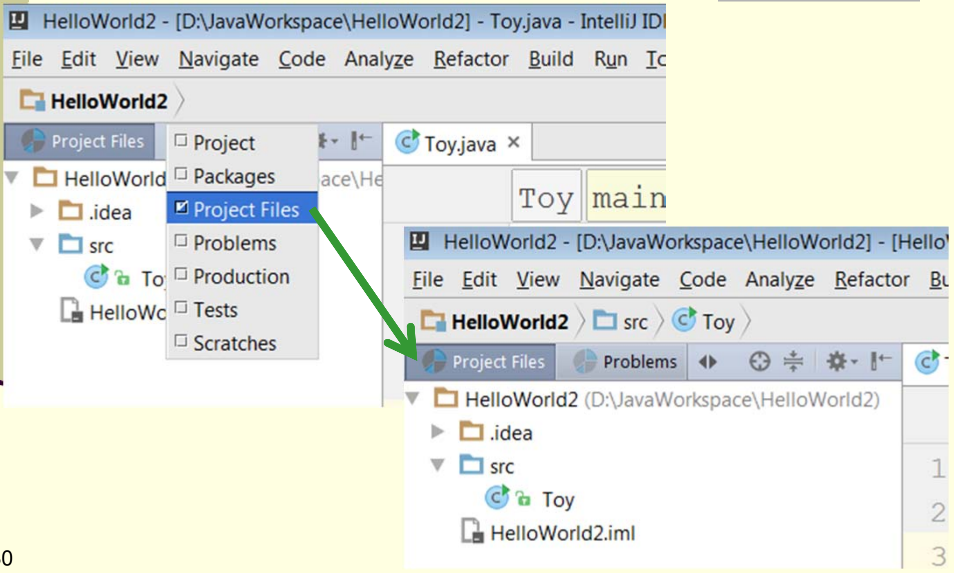
Find and Fix the Syntax Error: #2

```java
import java.util.Scanner;

public class Toy
{
  public static void main(String[] args)

    System.out.println("Input 2 numbers");

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

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

Find and Fix the Syntax Error: #3

```java
import java.util.Scanner;

public class Toy
{
  public static void main(String[] args);
  {
    System.out.println("Input 2 numbers");

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

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

Setting IntelliJ to **Project Files** View


HelloWorld2 - [D:\JavaWorkspace\HelloWorld2] - Toy.java - IntelliJ IDI
File Edit View Navigate Code Analyze Refactor Build Run To
HelloWorld2
Project Files
Project
Packages
Project Files
Problems
Production
Tests
Scratches
Toy.java
HelloWorld
.idea
src
To
HelloWo
Toy main
HelloWorld2 - [D:\JavaWorkspace\HelloWorld2] - [Hello
File Edit View Navigate Code Analyze Refactor Bu
HelloWorld2 src Toy
Project Files Problems
HelloWorld2 (D:\JavaWorkspace\HelloWorld2)
.idea
src
Toy
HelloWorld2.iml