



Lecture 1: Introduction to Java Programming

Basic lab instructions



- Talk to your classmates for help. You can even work on the lab with a partner if you like.
- You may want to bring your textbook or look up the internet for syntax and examples.
- Stuck? Confused? Have a question? Ask a Lab Assistance (LA) for help, or look at the book or past lecture slides.
- Complete as many problems as you can within the allotted time. You don't need to keep working on these exercises after you leave the lab.
- Feel free to complete problems in any way.

Goals



- Practice **writing**, **compiling**, and **running** Java programs
- Start using **Eclipse** editor software
- Gain familiarity with **syntax errors** and debugging

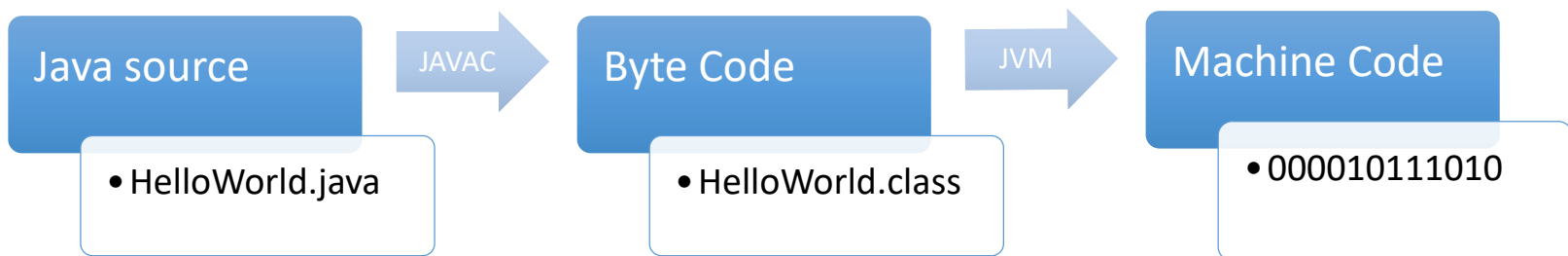
I. KEYNOTES

- Simple
 - No pointer
 - Automatic garbage collection
 - Not support multiple inheritances
 - Rich class library
- Object oriented:
 - All code is encapsulated in class.
 - All functions are associated with class.
 - Almost data types are objected

Java (cont)



- Interpreted:
 - Compiled into byte code for the JVM (Java Virtual Machine).
 - Byte code is dependent on the Java platform, but is typically independent of operating system specific features.



II. TUTORIAL



- ✓ Start project
- ✓ Your first program
- ✓ Running code
- ✓ Debug code
- ✓ Export to executable program

1. Software requirement

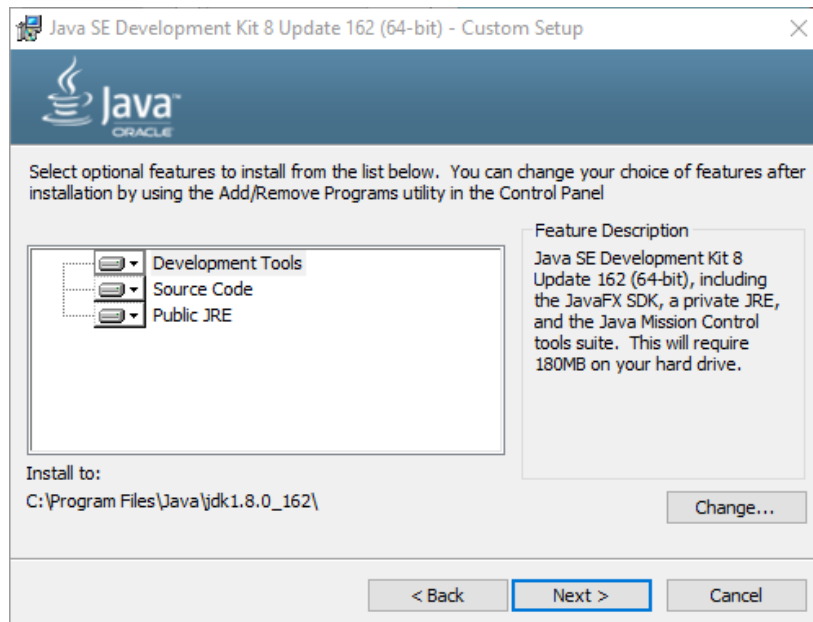


- Java SE Software Development Kit 8 (JDK 7)

Link:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

- Run the execute file and follow the install steps.



1. Software requirement



- Eclipse Oxygen (4.7) for Java Developer

Link:

<http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/oxygen2>

- Simply unwrap the zip file to some directory where you want to store the executables .

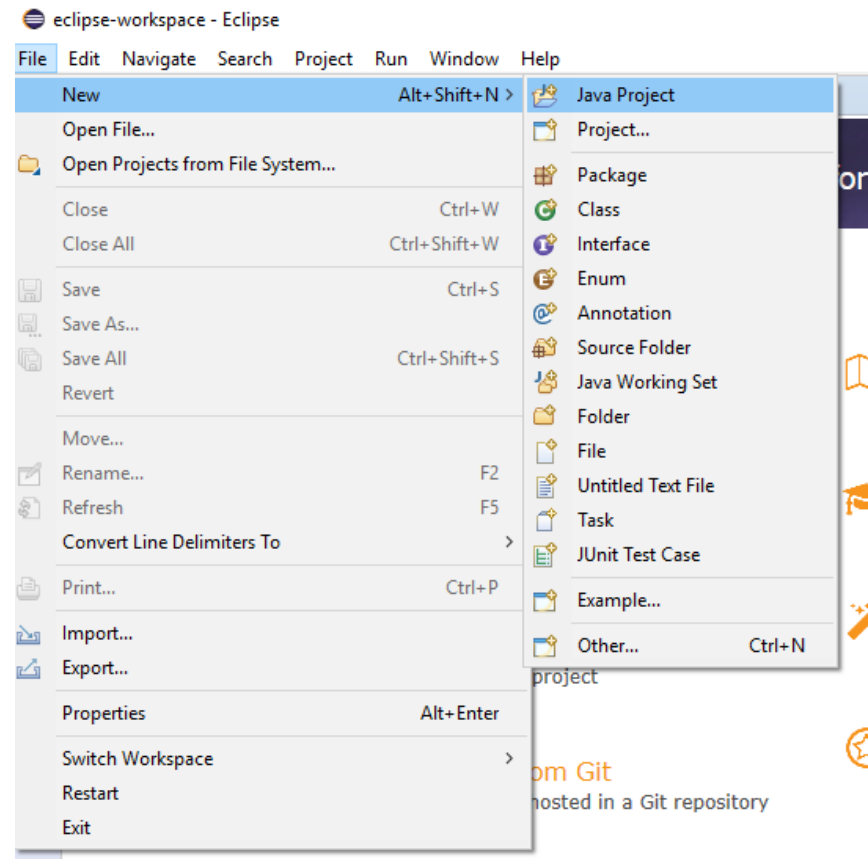


2. Working with Eclipse

Creating new project



- Open Eclipse, and create new project



2. Working with Eclipse

Creating new project



New Java Project

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

Project name:

☒ Use default location
Location: [Browse...](#)

JRE

☒ Use an execution environment JRE:
☐ Use a project specific JRE:
☐ Use default JRE (currently 'jre1.8.0_162') [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 [New...](#)
Working sets: [Select...](#)

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

New Java Project

Java Settings

Define the Java build settings.

Source Projects Libraries Order and Export

☒ HelloJava
 > ☒ src

▼ Details

[Create new source folder](#): use this if you want to add a new source folder to your project.

[Link additional source](#): use this if you have a folder in the file system that should be used as additional source folder.

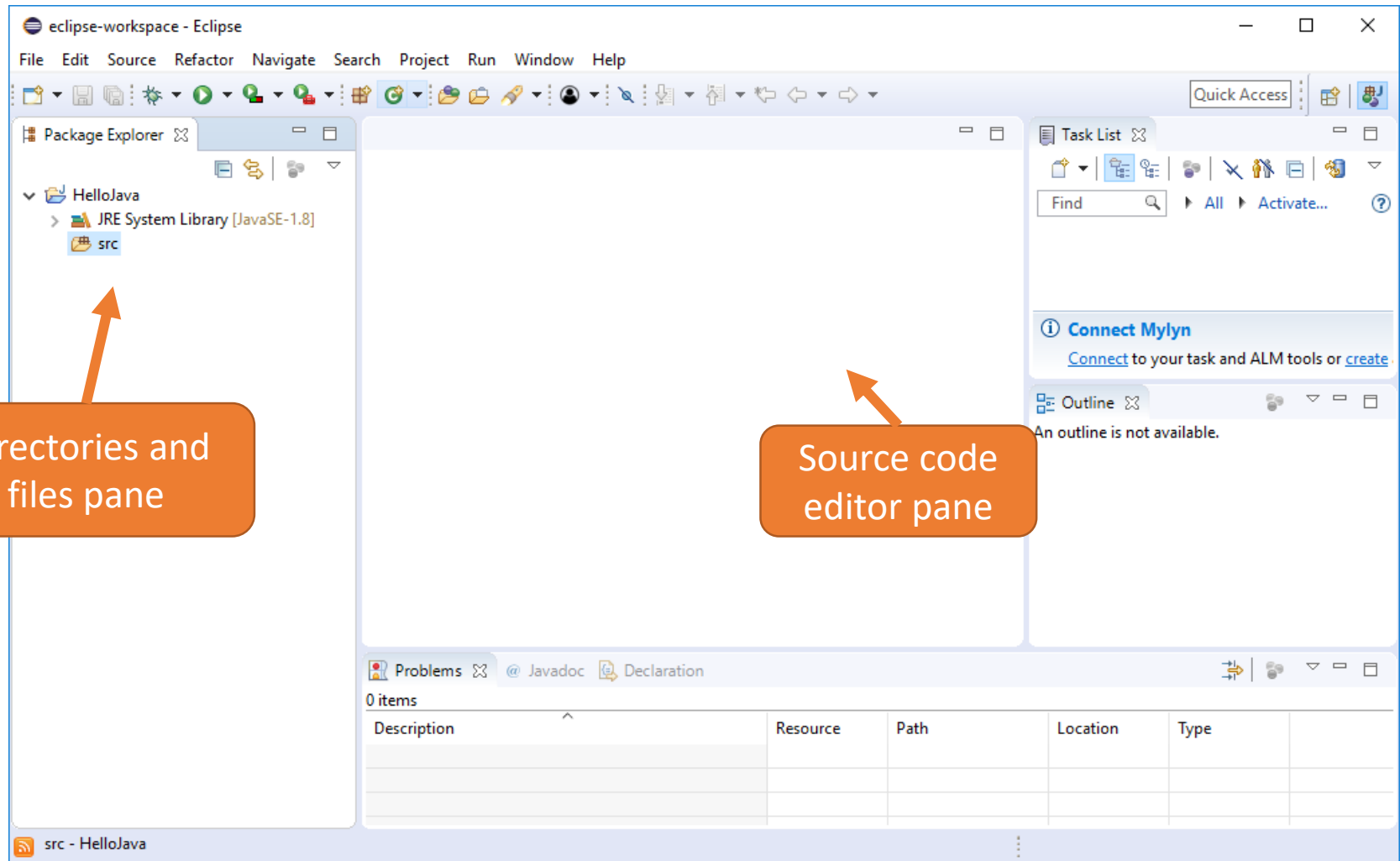
[Add project 'HelloJava' to build path](#): Add the project to the build path if the project is the root of packages and source files. Entries on the build path are visible to the compiler and used for building.

☐ Allow output folders for source folders

Default output folder:
 [Browse...](#)

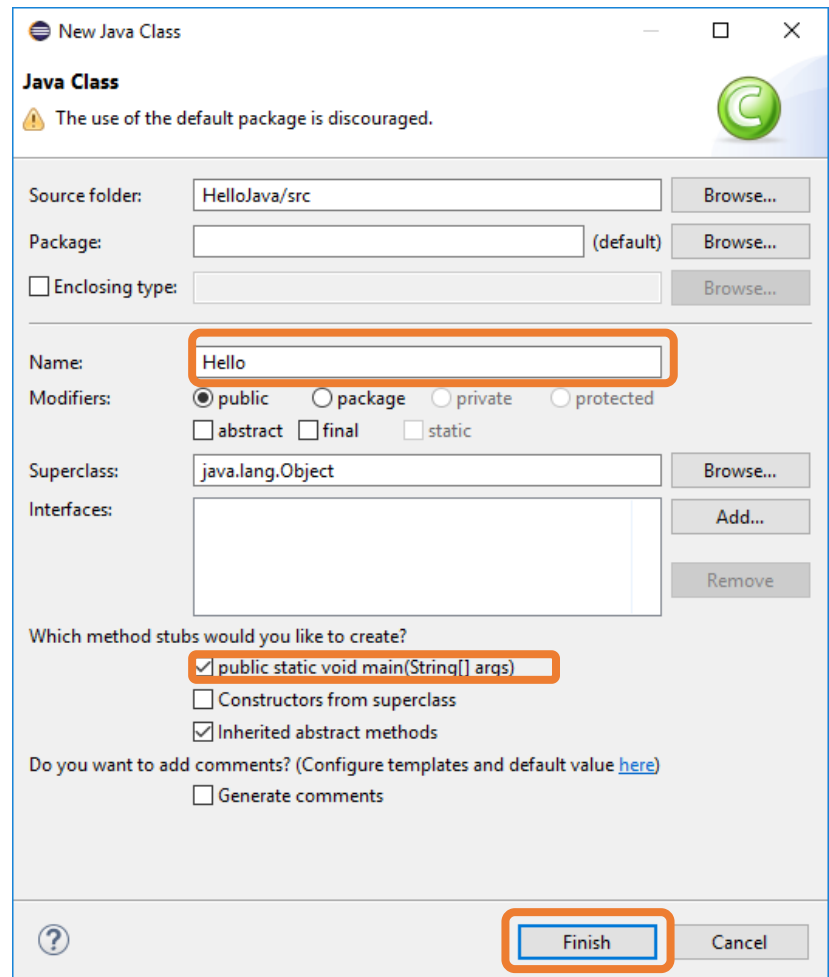
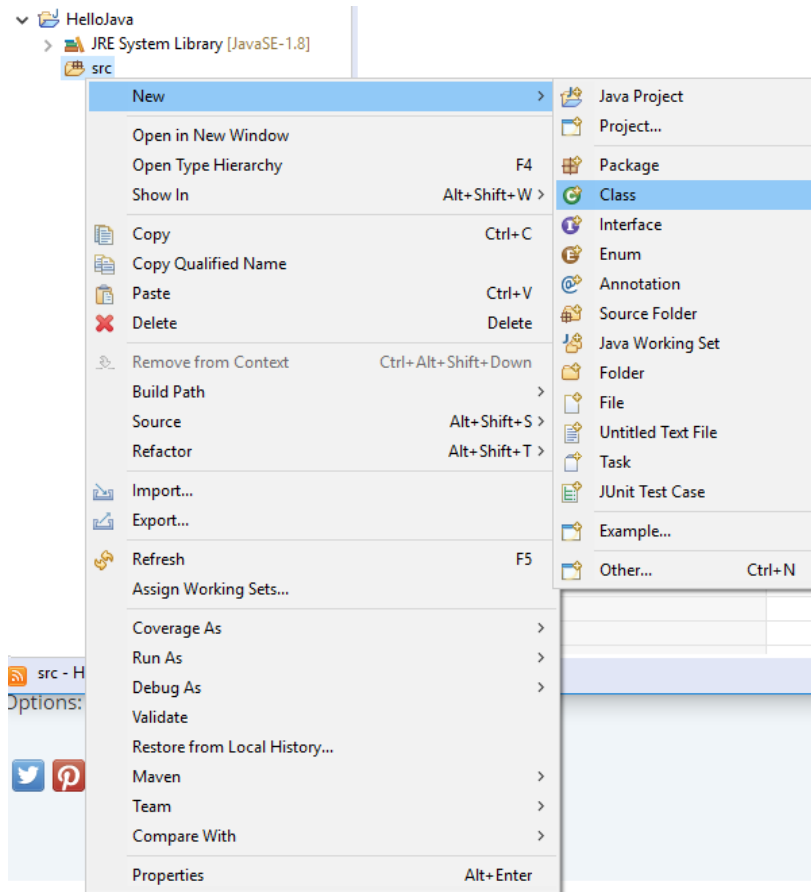
[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

2. Working with Eclipse Main GUI



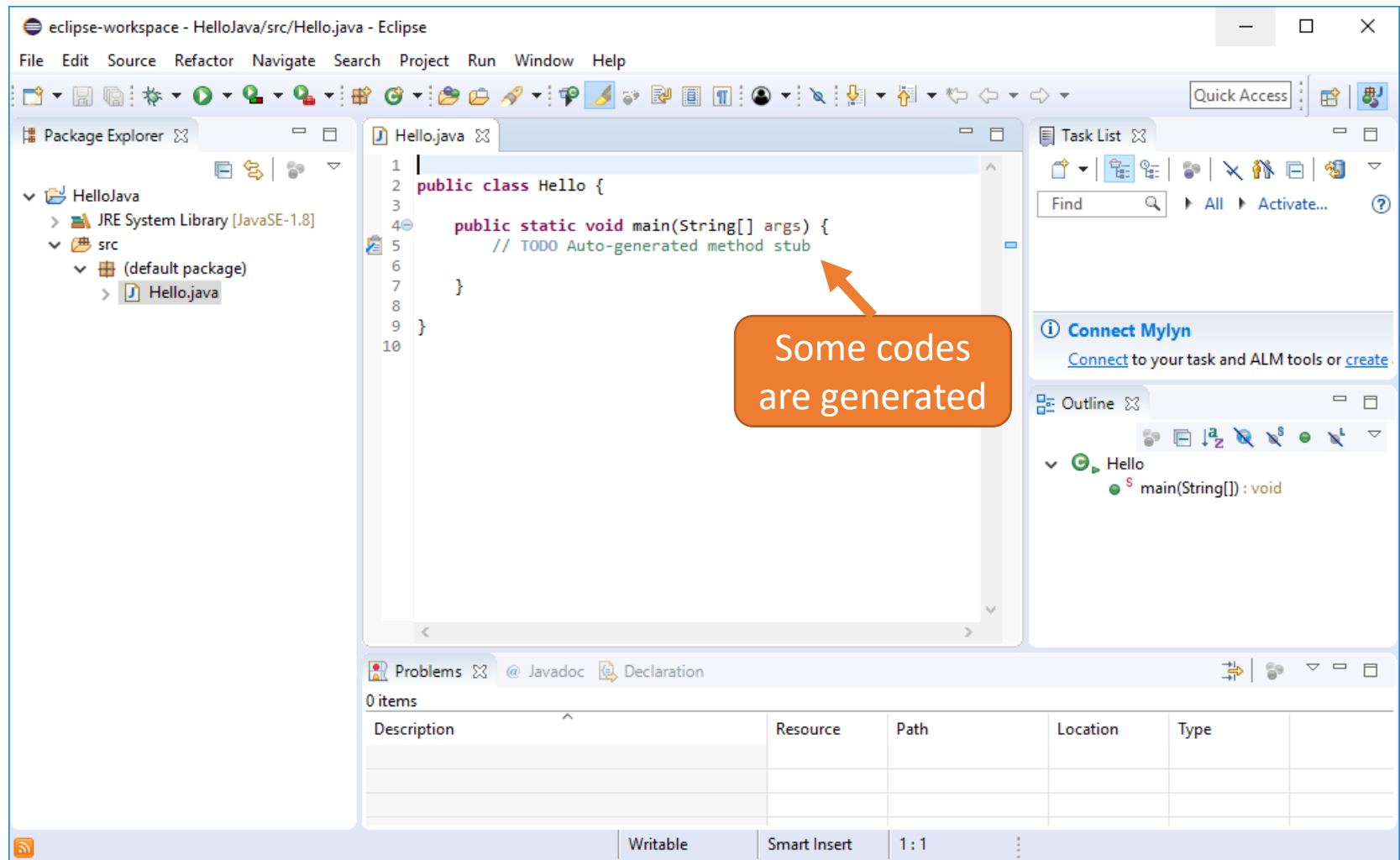
3. First Application

Creating a class



2. Working with Eclipse

Creating a class



3. First Application

Printing output



- Using to print function in java console application
- Standard
 - `System.out.print()`
 - `System.out.println()`
- With format
 - `System.out.printf()`
 - `System.out.format()`

```
int a = 10;
int b = 20;
// Tedious string concatenation.
System.out.println("a: " + a + " b: " + b);
// Output using string formatting.
System.out.printf("a: %d b: %d\n", a, b);
```

3. First Application

Printing output with format



- Format String:

- **% [flags] [width] [.precision] conversion-character** (square brackets denote optional parameters)

```
System.out.println(String.format("%-10d%-10d", 12345, 67890));  
12345    67890
```

- Flags:

```
System.out.println(String.format("%+10d%+10d", 100, -200));
```

- **-** : left-justify (default is to right-justify)+100 -200
- **+** : output a plus (+) or minus (-) sign for a numerical value
- **0** : forces numerical values to be zero-padded (default is blank padding)
- **,** : comma grouping separator (for numbers > 1000)
- : space will display a minus sign if the number is negative or a space if it is positive

- Conversion-Characters:

- **d** : decimal integer [byte, short, int, long]
- **f** : floating-point number [float, double]
- **c** : character Capital C will uppercase the letter
- **s** : String Capital S will uppercase all the letters in the string
- **h** : hashcode A hashcode is like an address. This is useful for printing a reference
- **n** : newline Platform specific newline character- use %n instead of \n for greater compatibility

3. First Application

Printing output with format example



```
long n = 461012;
```

```
System.out.format("%d%n", n);           // --> "461012"
```

```
System.out.format("%08d%n", n);         // --> "00461012"
```

```
System.out.format("% ,8d%n", n);        // --> " 461,012"
```

```
System.out.format("%+,8d%n", n);        // --> "+461,012"
```

```
double pi = Math.PI;
```

```
System.out.format("%f%n", pi);          // --> "3.141593"
```

```
System.out.format("%.3f%n", pi);        // --> "3.142"
```

```
System.out.format("%10.3f%n", pi);       // --> "      3.142"
```

```
System.out.format("%-10.3f%n", pi);      // --> "3.142"
```

3. First Application

Get the input



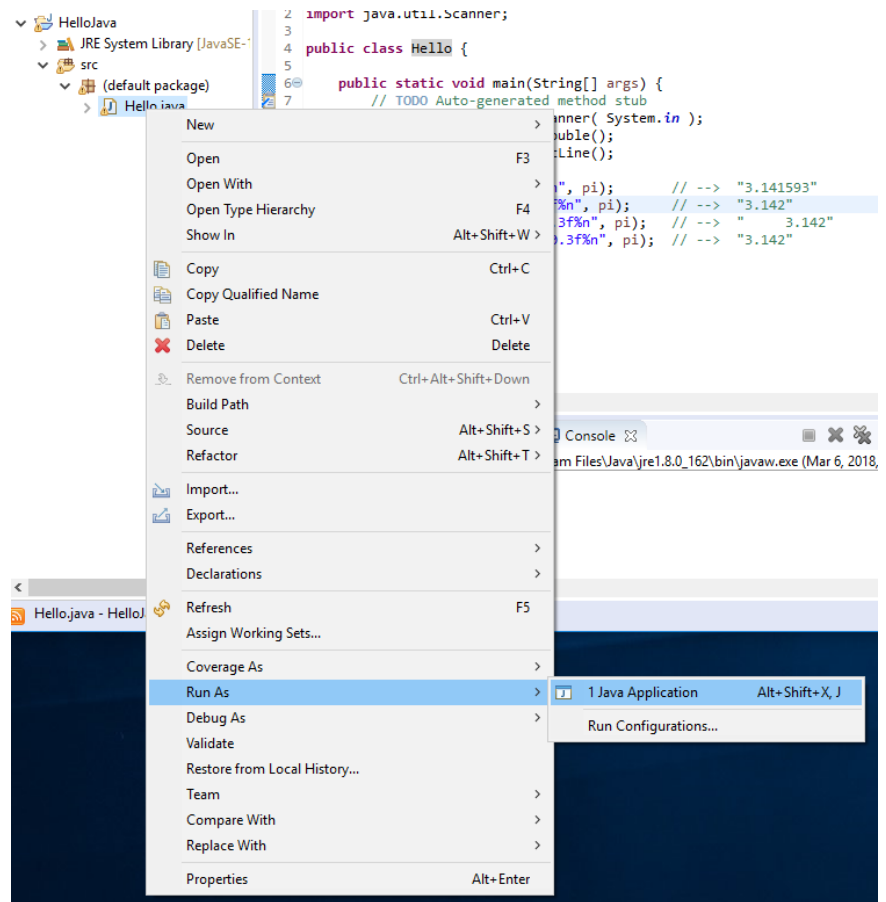
- Use Scanner object to read the input:
 - `Scanner input = new Scanner(System.in);`
- For different input type
 - `input.nextDouble()` to read in double
 - `input.nextInt()` to read in integer
 - `input.nextLine()` to read in a String

```
import java.util.Scanner;  
  
...  
// create Scanner to obtain input from command line  
Scanner input = new Scanner( System.in );  
int i= input.nextInt();  
String str = input.nextLine();
```

4. Running Your App

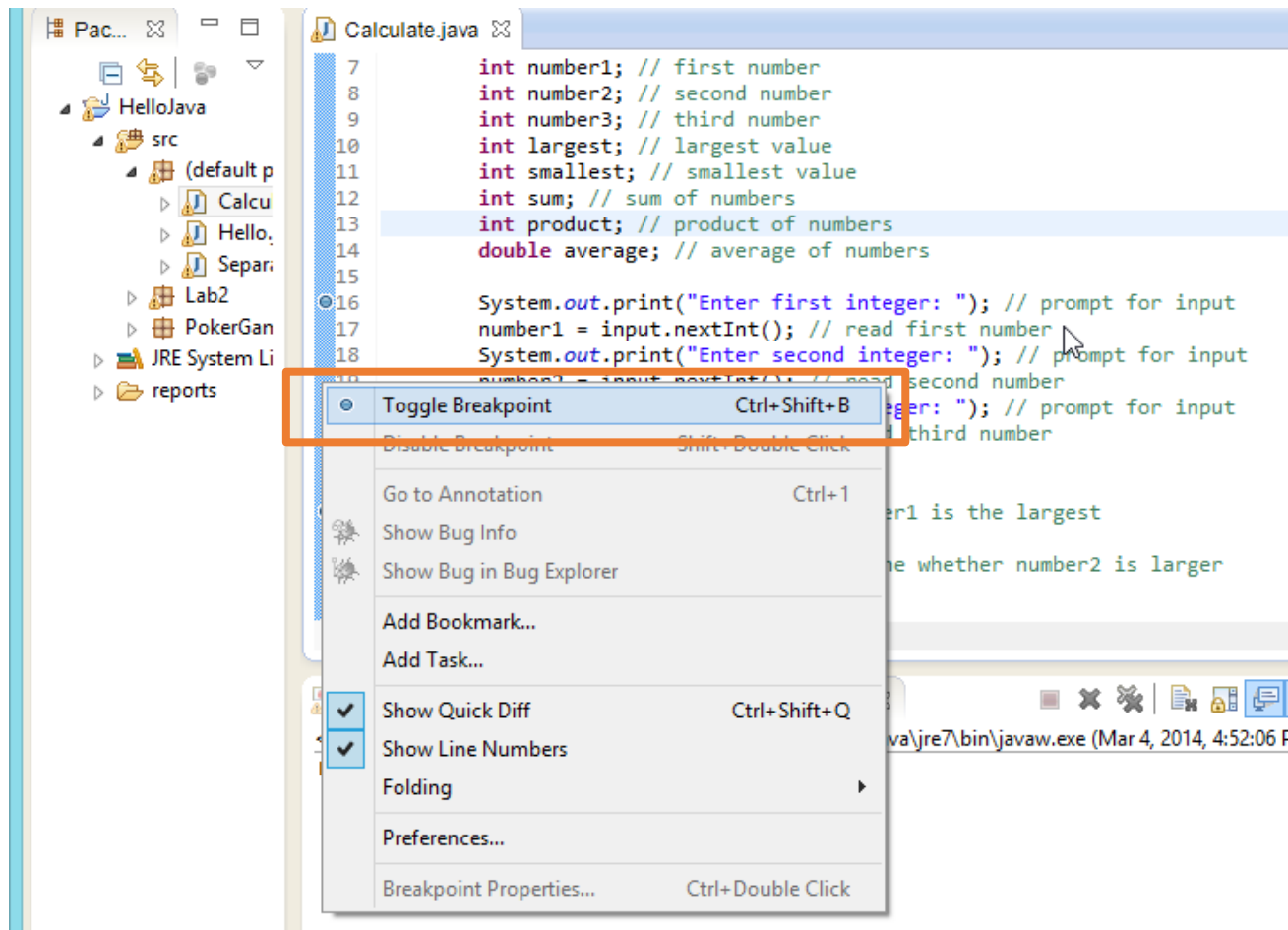


- Right click on the class
Select Run As -> Java
Application
- Menu Run -> Run
- Note:
 - Show console log:
Menu Window-> Show
view -> Console



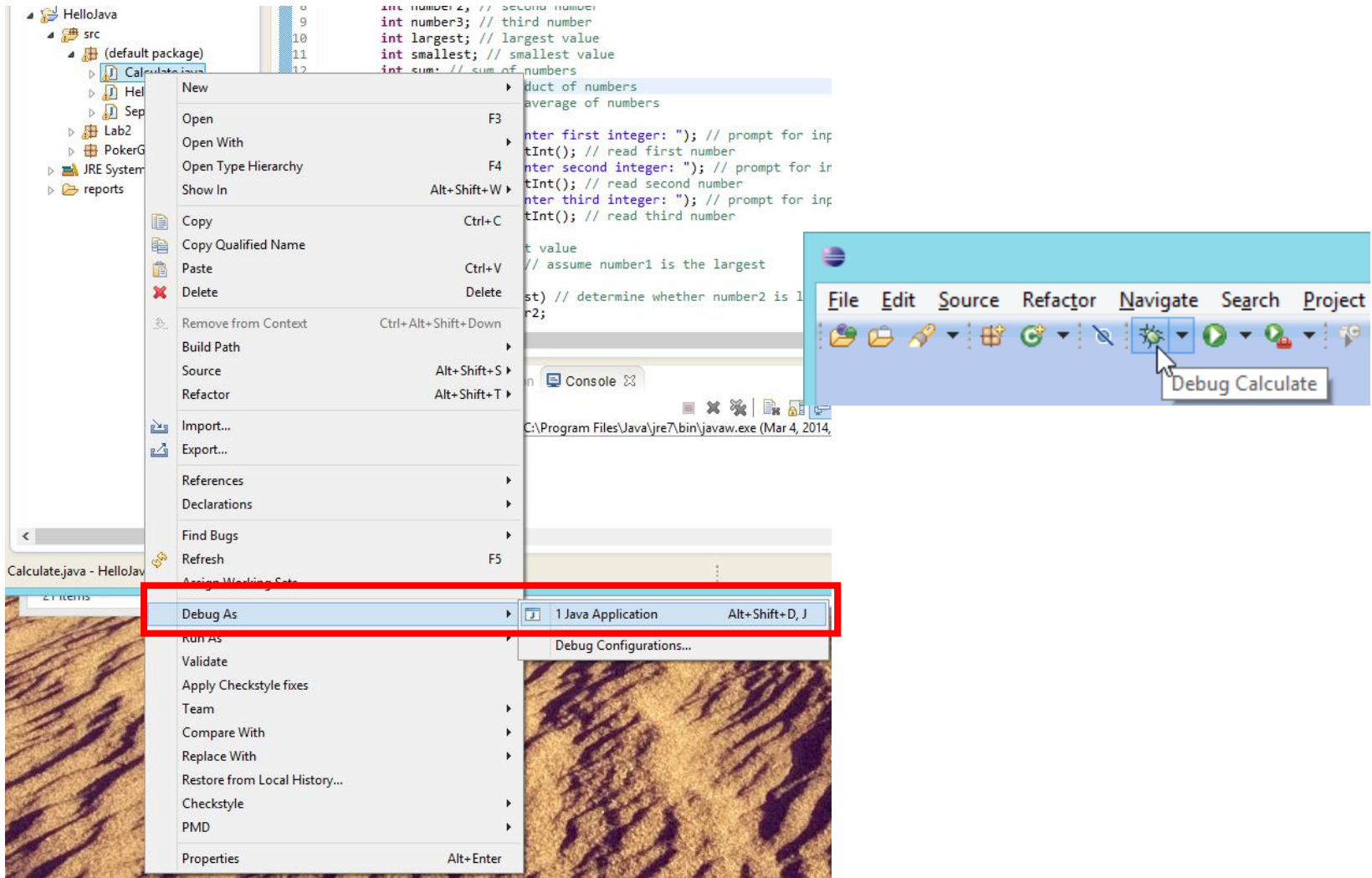
5. Debugging

Setting breakpoints



5. Debugging

Starting debugging



• 2.

5. Debugging

Debug perspective



Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying thread management.

Do you want to open this perspective?

☐ Remember my decision

Debug - HelloJava/src/Calculate.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Java PMD Debug

Debug

- Calculate [Java Application]
 - Calculate at localhost:50306
 - Thread [main] (Suspended (breakpoint at line 16 in Calculate))
 - Calculate.main(String[]) line: 16
- C:\Program Files\Java\jre7\bin\javaw.exe (Mar 4, 2014, 5:07:35 PM)

Variables

| Name | Value |
|-------|------------------|
| args | String[] (id=16) |
| input | Scanner (id=18) |

Calculate.java

```
7 int number1; // first number
8 int number2; // second number
9 int number3; // third number
10 int largest; // largest value
11 int smallest; // smallest value
12 int sum; // sum of numbers
13 int product; // product of numbers
14 double average; // average of numbers
15
16 System.out.print("Enter first integer: "); // prompt for input
17 number1 = input.nextInt(); // read first number
18 System.out.print("Enter second integer: "); // prompt for input
```

Outline

- Calculate
 - main(String[]) : void

Console

Calculate [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Mar 4, 2014, 5:07:35 PM)

```
|
```

III. PRACTICE



Exercise 1: Name



- **Objective:**

- Using `System.out.printf` to output text and characters
- Using Scanner class to get input text
- Compiling and executing Java applications.
- Debug your program, add breakpoint and watch your variable.

- **Problem:**

- Write an application that **input your name**, and **displays** the welcome "*Hello <your name>.*"

- **Sample output:**

Input your name: Java

Hello Java.

Exercise 2: Numbers



- **Objective:**

- Using the *Scanner* class to obtain input from the user.
- Using *System.out.printf* to output information to the user.
- Using arithmetic operators to perform calculations and operators to compare variable values.
- Using *if* statements to make decisions based on the truth or falsity of a condition.

- **Problem:**

- Write an application that **inputs three integers** from the user and displays the **sum, average, product, smallest** and **largest** of the numbers.
- *Note: The calculation of the average will result in a float with precision is 3 number. i.e. 3.333*

- **Sample output:**

```
Enter first integer: 10
Enter second integer: 20
Enter third integer: 30
For the numbers 10, 20 and 30
Largest is 30
Smallest is 10
Sum is 60
Product is 6000
Average is 20.000
```