

Lab1

Getting Started with Java Using Eclipse

Objectives:

At the end of this lab, you should be able to

- Create a new project, package and Java class in Eclipse.
- Edit, correct, and run a simple Java program within Eclipse.

Part 1: Getting Started (Instructor-lead) (Time: 50 minutes)

Introduction to Eclipse


During this course, you will be using Eclipse to develop java programs. Eclipse is an open-source, Java-based, extensible development platform designed for nothing in particular but everything in general. Because of its roots, it is currently most popular as a Java integrated development environment (IDE). Eclipse ships with plugins for writing and debugging Java code. Additional plugins for more advanced Java development, such as JavaFX, JSP/servlets, are also available.



Preparing your Java Computing Environment

1. Install Java by downloading the Java JDK (Java Development Kit) from the Oracle website:
<https://www.oracle.com/technetwork/java/javase/downloads/index.html>
2. Install the Eclipse IDE platform as follows:
 - Download the Eclipse IDE (2019-12) installer from this link:
<https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2019-12/R/eclipse-inst-win64.exe>
 - Run the installer and choose **Eclipse IDE for Java Developers**. Make the installation folder as C:\java-2019-12, then click next, and wait for the installation to finish.
 - Locate the **eclipse.exe** file which has the icon  eclipse (found in C:\java-2019-12\eclipse)
 - Create a shortcut and save it in the desktop.
3. Create your own workspace. Create a new folder CMPS251Name; personalize it by replacing Name with your own name.

Getting Started

After installing eclipse on your computer, you can start the program by running the executable file eclipse by double-clicking on the following icon.  eclipse

Before the eclipse starts, a dialog box is displayed to specify the workspace which is the folder where your work will be saved. You can use the browse button to browse your disk.

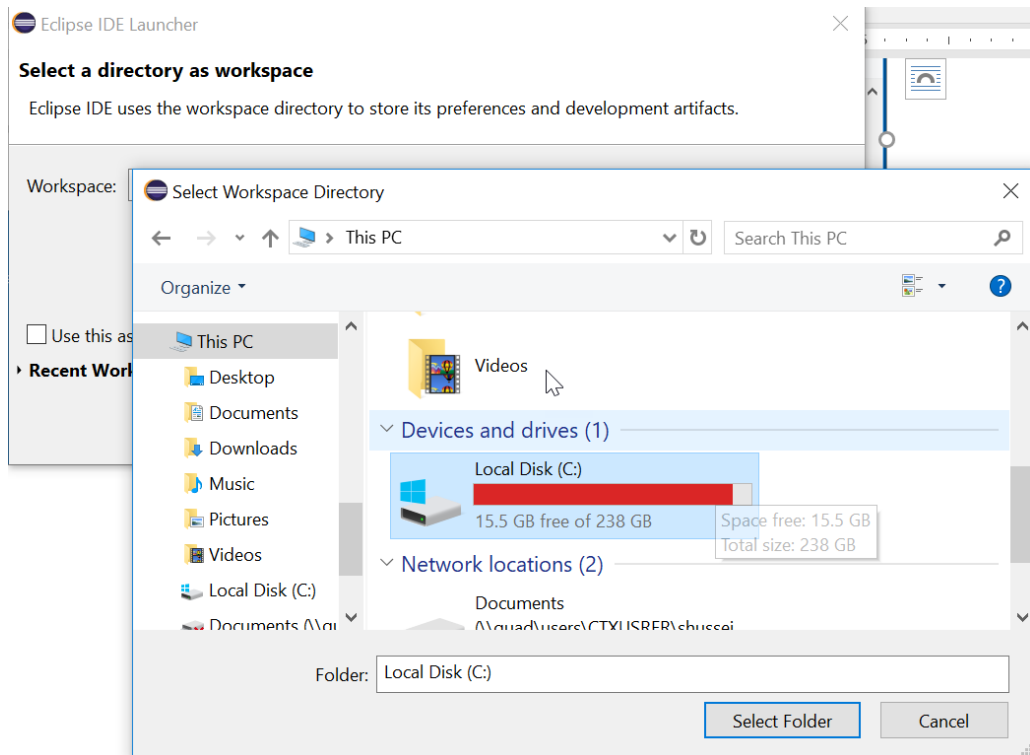


Figure 1

For the first time, you will get a welcome page as in Figure 2.

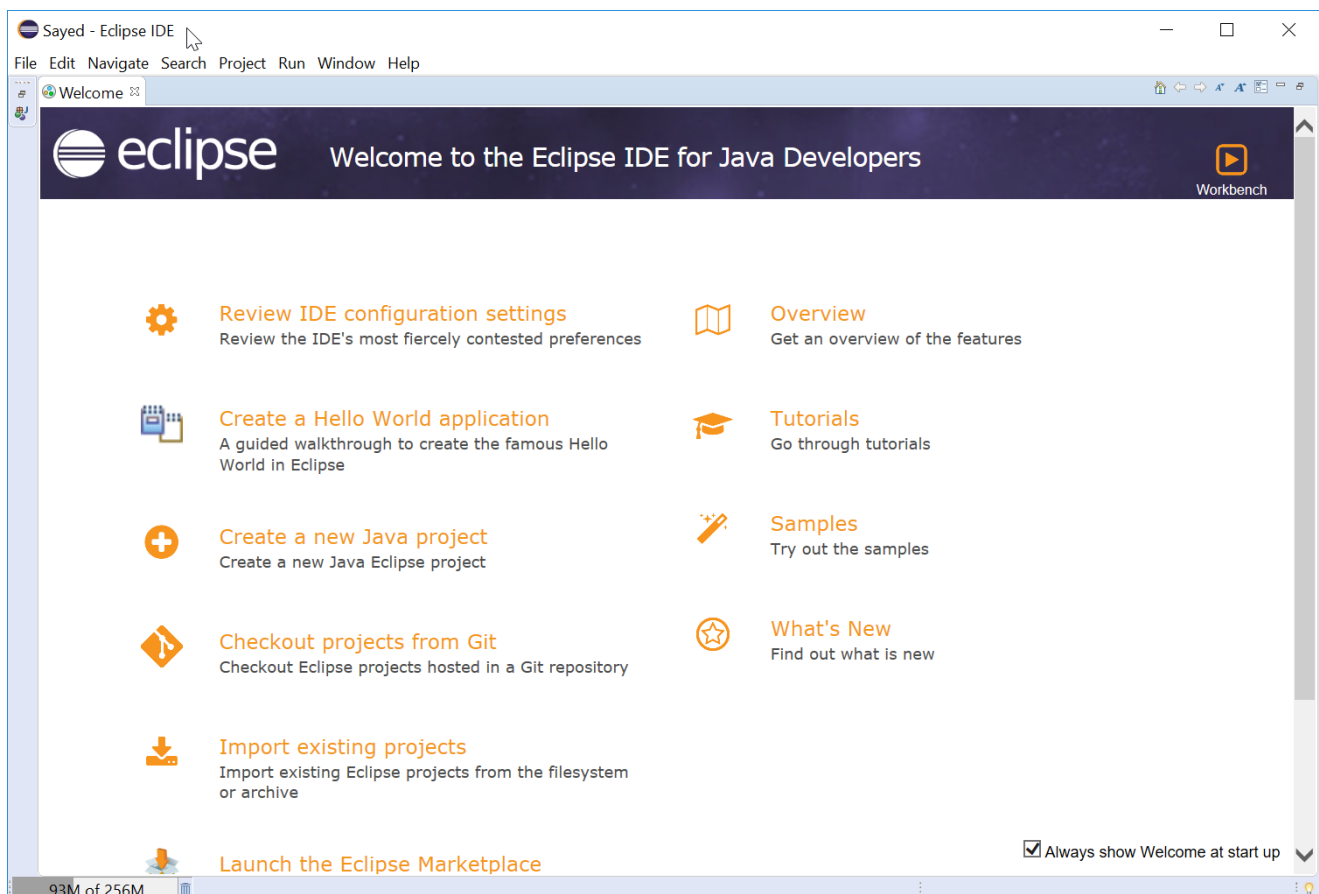


Figure 2

You can explore eclipse features from this page, or you can simply close it to start your first project and move to the main window of eclipse.

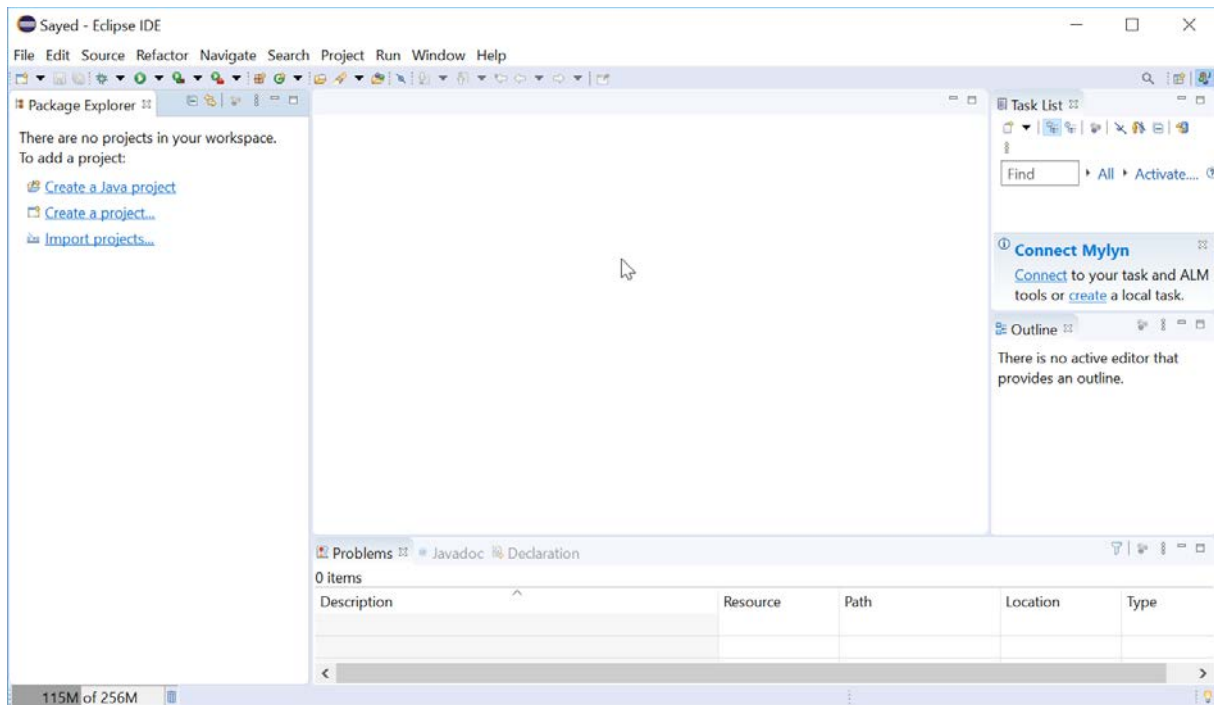


Figure 3

As you can see from figure 3 that main window divides into smaller windows called Views. Each view displays different information about your work. You can close any view and you can open a specific view by selecting Window menu-> Show View-> and select the desired view.

Your First Cup of Java in Eclipse

The following steps explain how to use eclipse to write a simple java program that prints the message “Welcome to Java Language”.



Can you guess why the logo of Java is a cup of coffee!!!!

1. Creating Project.

Eclipse allows you to organize your work into Projects. By default, Eclipse stores a new Project in a subdirectory under the workspace directory. In order to create a new project, select File->New-> Java Project.

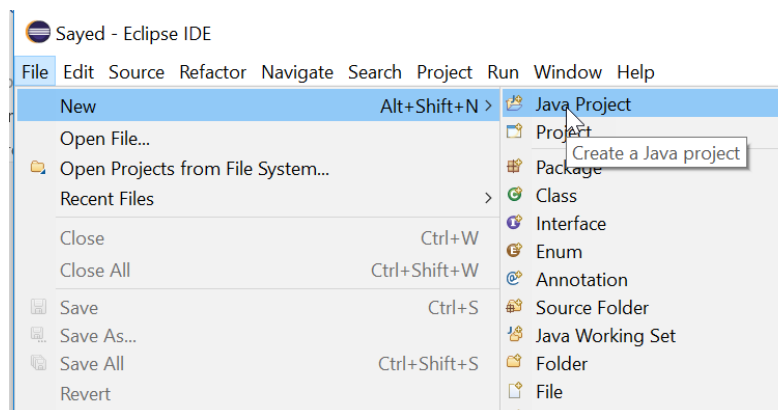


Figure 4

Alternatively, you can right click in either the Package Explorer View or the Navigator View, and also select New -> Java Project.

A new window like the one in figure 5 will be opened.

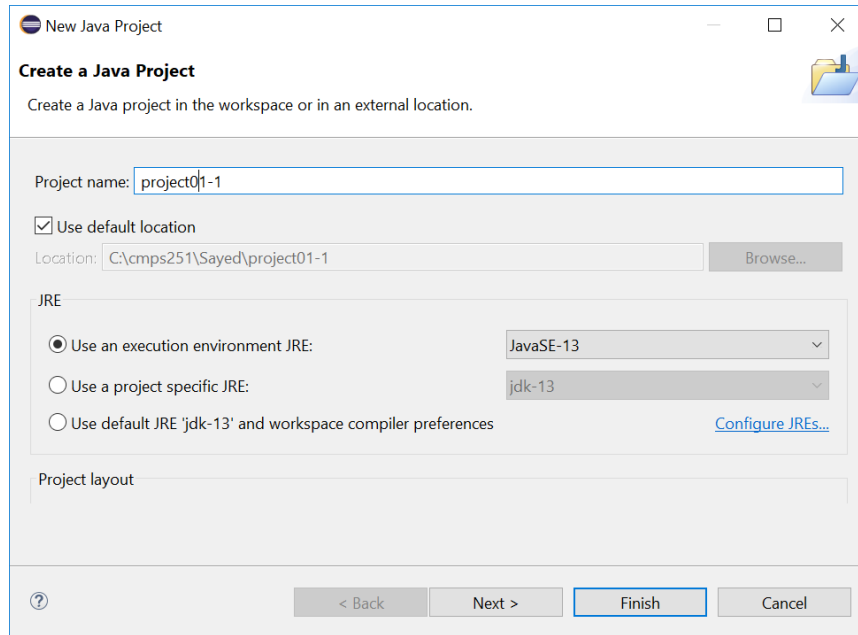


Figure 5

In the “New Java Project” dialog, you can specify the name for the project. Let’s call it “project01-1”, then click Finish button. On the dialog box called “**New module-info.java**” click the “**Don't Create**” button.

2. Creating Package

A Package in Java is a group of classes, which are often closely or logically related in some way. Each new project has a default package. Programs of the default package are stored directly in the project's main directory. When you have many classes (programs) in your project, it is recommended to use packages to organize your work.

To create a new package in “project01-1” project:

1. In the Package Explorer View, select the “project01-1” Project.
2. Select File -> New -> Package, or right click on **project01-1** and select New -> Package.

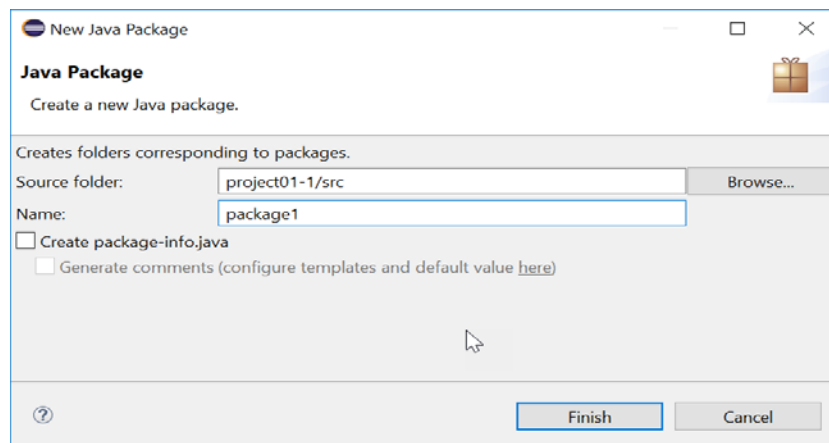


Figure 6

3. The Source Folder field indicates the project in which this package will be created. It should say **project01-1/src** ; if not then type it in, or click Browse and select it from the resultant list.
4. Specify the name of the package in the Name field. Enter “package1” as the name for the package.
5. Click Finish to create the package. A new package is created.

3. Creating Class

Creating a class is similar to creating package.

1. In the Package Explorer, select package1, which is the package in which the class will be created.
2. Select File -> New -> Class, or right click on package1 and select New -> Class.

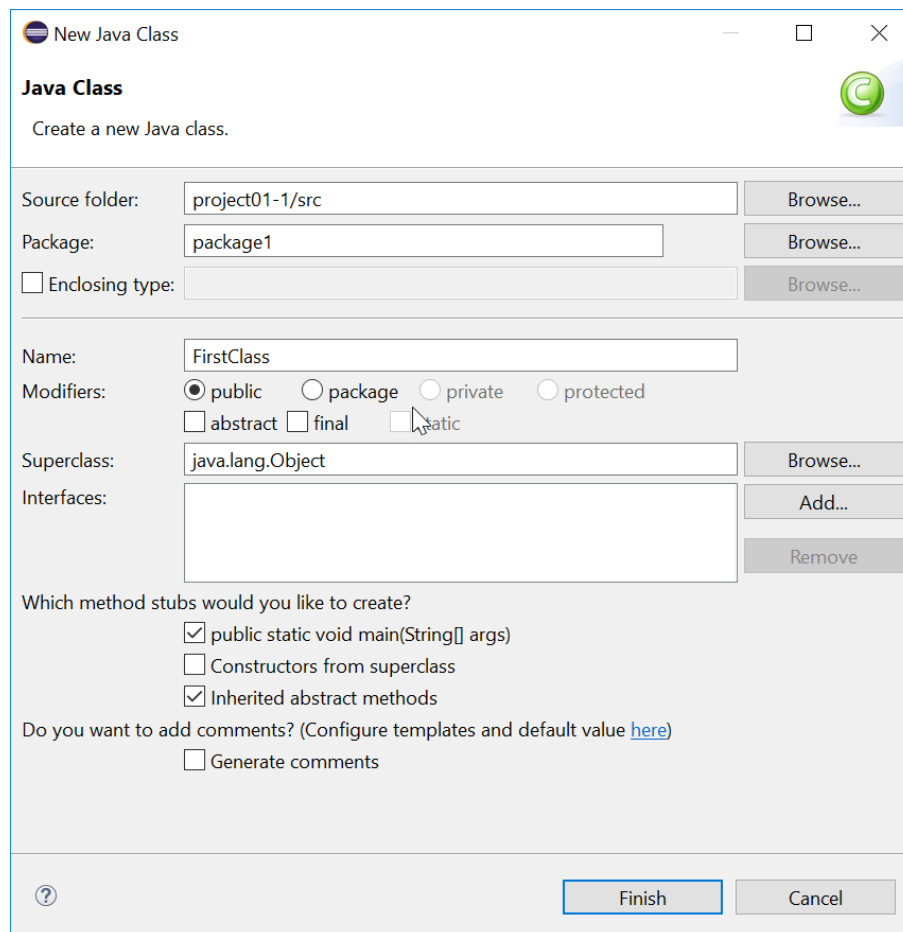


Figure 7

3. In the Name field enter the class name “FirstClass”
4. Eclipse provides templates which greatly simplify class creation. These templates can generate skeletons for the basic code required by the new class. Therefore, we need to specify the options for this template as following :
 - Under Modifiers select public, which specifies this will be a public class.
 - The Superclass field should read **java.lang.Object**. If we wanted to extend an existing class, we could specify it here, and have Eclipse generate stubs for all super class constructors and inherited abstract methods.
 - The Interfaces field should be empty.
 - Select only the public static void main(String[] args) checkbox.
5. Click Finish so the class is created.

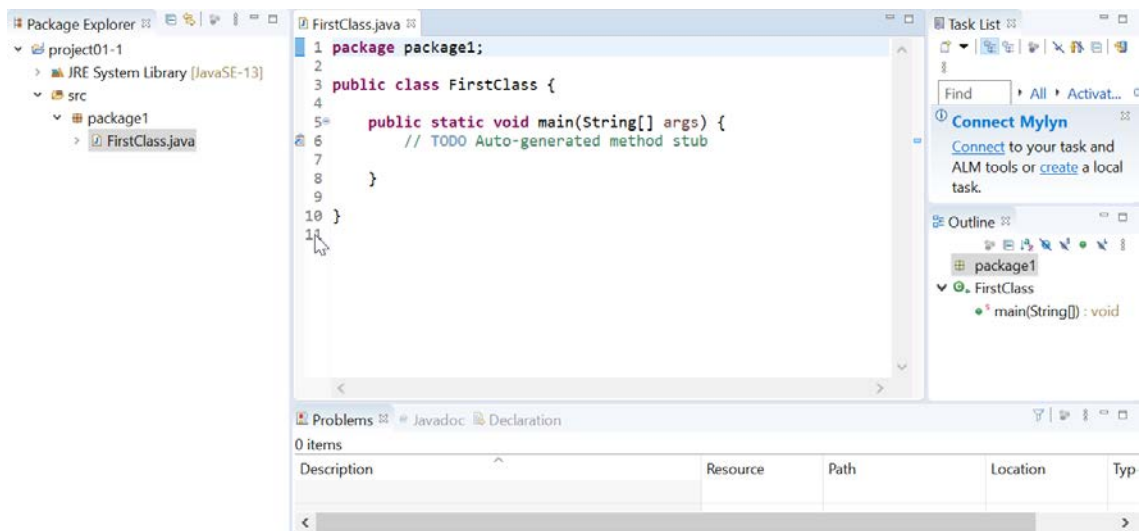


Figure 8

As you can see, Eclipse created a class with a template of code. The code is for the FirstClass, which has a method called “main”. This method is the start point of execution for any project.

The generated code also contains some comments.

6. Inside the main method add the following line

System.out.println(“Welcome to Java Language”);

Note that when you type the statement, Eclipse provides many features to help you write your code.

Each time you type “.”, it displays many options to choose.

If there is a syntax error, red line appear to point to the error. If you point to the sign you get suggestions to correct this error.

Running the Program using Eclipse IDE.

You can run your program from Run menu, select “Run As”->Java Application.

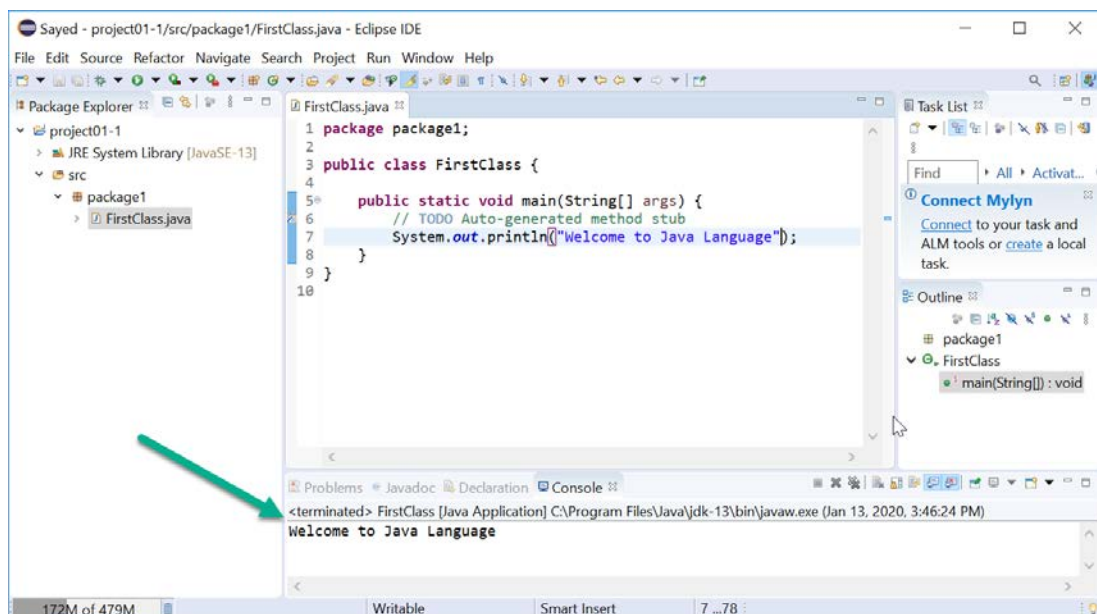


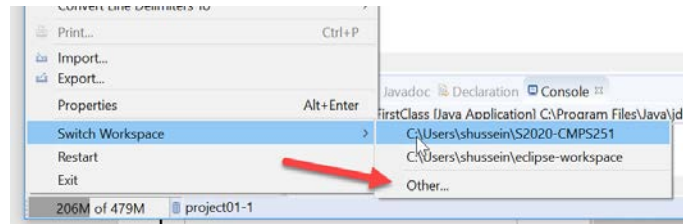
Figure 9

If you did not save the file yet, you will be asked to do so. If your program is correct, the output will be displayed in Console view as shown in Figure 9 above.

Notes:

1. Eclipse saves your files in the specified workspace. To see the folder of current workspace, point to your project->right click and select **Properties**, select **Resource**.
2. To change the workspace to another folder:

Select File -> Switch workspace
choose an existing workspace, or
choose Other and create a new one.



3. For each project, Eclipse creates a corresponding folder in the workspace. This folder contains two subfolders:
 - **src**: this folder contains the source files (.java files).
 - **bin**: this folder contains object files (.class files).



4. To regenerate .class file, you need to resave your java files in Eclipse.

Part 2: Practice Exercises (Time: 70 minutes)

Exercise 1:

Change the above program so that it outputs your name repeated 10 times, each in a new line (hint: use a for-loop).

Exercise 2: Keyboard Input: To use the keyboard for input:

- 1) Import **java.util.Scanner**.
 - 2) Create an input object, let us call it **cin** for example, using the following statement:
`Scanner cin = new Scanner(System.in);`
 - 3) To read an **integer** from the keyboard use the **nextInt()** method of the **cin** object:
`integer=cin.nextInt();`
 - 4) In a similar fashion use **nextFloat()**, **nextDouble()** for reading float numbers and double numbers. For string input use the **next()** method.
 - 5) When you finish reading all your input make sure you **close** the input stream (**cin.close()**).
- Write a Java application that reads two strings representing the user's first name and last name and then concatenate the two strings in a new string called `fullName` (insert a single space to separate the first name and the last name). Display the full name.
Hint: use the **next()** method of the input object as described in above.
 - Now, run the application and enter your name, your father's name and your last name all in one line. What was the output? _____
Can the **next()** method read a String that includes spaces? _____
Replace the **next()** method with **nextLine()** and run the application and give it your full name separated with spaces. What was the output? _____
What is your conclusion? _____

Exercise 3:

Write a Java application that reads three integer numbers. Calculate the sum and average for these numbers. Display the three numbers and their sum and average.

Additional Practice Problems

Exercise 4: Concatenating Text and Numbers

- Create a Java project and let the *class wizard* create a class called `NumbersTest` that contains a main method.
- Perform the following in the `main()` method:
 - Add two 'double' variables, `var1` and `var2` and initialize them with `3.33` and `6.67` respectively.
 - Store the sum of the two numbers in a variable called `sum`.
 - Output the values of the two variables and their sum, properly labelled, each on a separate line. (see the hint given below)

Hint: To output `var1=3.3` we concatenate (i.e. join) the label "`var1=`" with the value of `var1` as in the following statement: [Java's concatenation operator is '+']

```
System.out.println("var1="+var1);
```

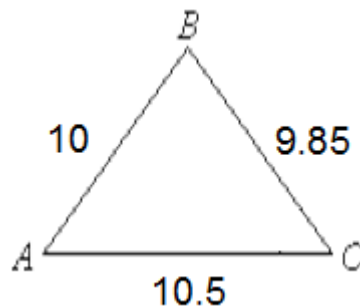
Exercise 5:

Write a Java application for the calculation of the *area* of a *scalene* triangle whose sides are *a*, *b* and *c*. Use the following formula:

$$area = \sqrt{s(s-a)(s-b)(s-c)}$$

where $s = (a + b + c) / 2$. [Hint. The square root function is `Math.sqrt()`]

Use your program to calculate the area of the following triangle:



Exercise 6:

Write a Java application to read three numbers representing the percent scores of a student in three subjects: Math, Computer, and English, and then compute and display their average. If the average is less than 60 then print an asterisk 'i.e. a *' next to the average. Test your program once with data that produces an average greater than 60 and once with a set of data that produces an average which is less than 60.