CS265 2018
Lab 01 (2.5%)
Getting Started with the Eclipse IDE and TestNG for use in CS265

This exercise is necessary to ensure that we all get Eclipse up and running on our computers – you must complete this exercise regardless if you an Eclipse expert or not. This guide should work for most recent version of Eclipse. However the guide is specifically written for Eclipse Neon.

1. Develop a simple class in Eclipse:

    - Startup Eclipse (**Eclipse Java Neon** in the Eclipse folder on your desktop)

    - If you are prompted for a workspace: enter a new folder name – or browse for it, dedicated to CS265 (for example: "X:\CS265\EclipseWorkspace" – do **not** use a folder on the C: drive). If you want you can choose to check the box "*Use this as default and do not ask again*".

    - If a "Subclipse" prompt appears – clear the "Report Usage…" checkbox and click on OK

    - Be patient – Eclipse takes a little time to start up.

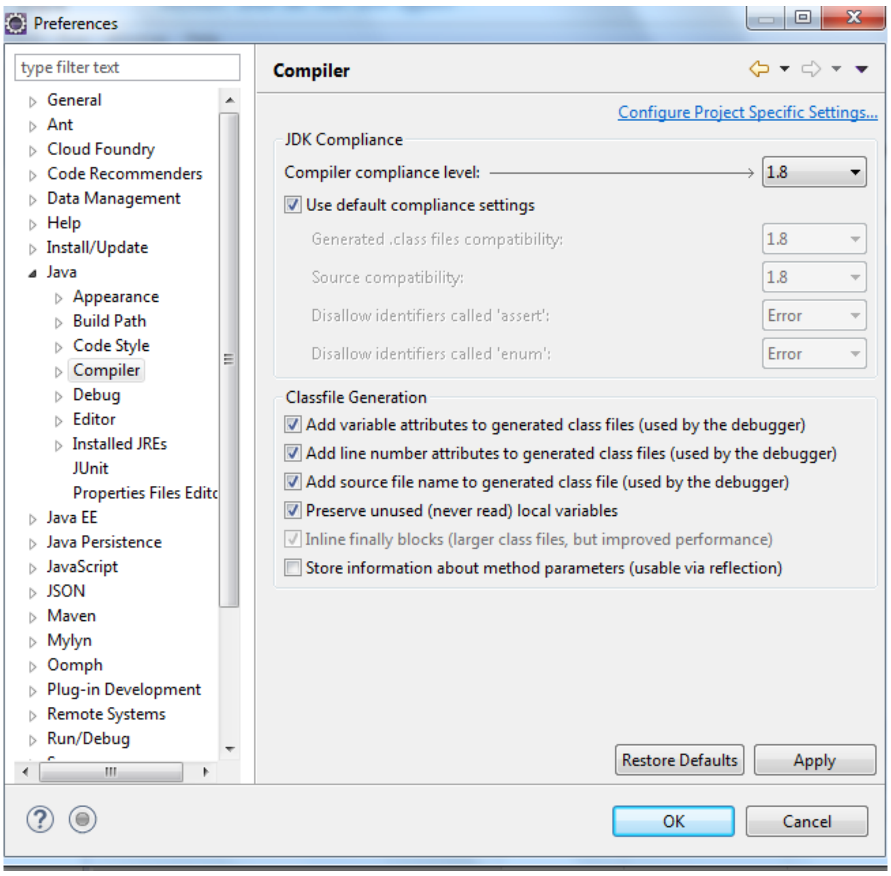
    - If the Welcome tab is visible, close it (click on the X)

    - If you were NOT prompted for a workspace when you started Eclipse (normally because you have run Eclipse before, and thereby set the default workspace), you can use File->Switch Workspace[1] to make sure you are in the right workspace, or to move to the right workspace. You can make a folder on your X-Drive at this point. **DO NOT USE THE C-DRIVE**. If you are using Linux use your home directory.

    - Eclipse may restart itself when you switch workspace in which case you'll go back to the 'Subclipse' message step shown previously.

    - To setup Java, use Window->Preferences->Java->Compiler and make sure the "Compiler Compliance Level" is set to 1.8 – if not, select 1.8 then click on APPLY and then OK. **Ask a demonstrator for help if you have problems**

[1] This means click on the **File** menu and then on the **Switch Workspace** menu item


Preferences

type filter text

- General
- Ant
- Cloud Foundry
- Code Recommenders
- Data Management
- Help
- Install/Update
- Java
  - Appearance
  - Build Path
  - Code Style
  - Compiler
  - Debug
  - Editor
  - Installed JREs
  - JUnit
  - Properties Files Editc
- Java EE
- Java Persistence
- JavaScript
- JSON
- Maven
- Mylyn
- Oomph
- Plug-in Development
- Remote Systems
- Run/Debug

Compiler

Configure Project Specific Settings...

JDK Compliance

Compiler compliance level: 1.8

[✓] Use default compliance settings

Generated .class files compatibility: 1.8

Source compatibility: 1.8

Disallow identifiers called 'assert': Error

Disallow identifiers called 'enum': Error

Classfile Generation

[✓] Add variable attributes to generated class files (used by the debugger)

[✓] Add line number attributes to generated class files (used by the debugger)

[✓] Add source file name to generated class file (used by the debugger)

[✓] Preserve unused (never read) local variables

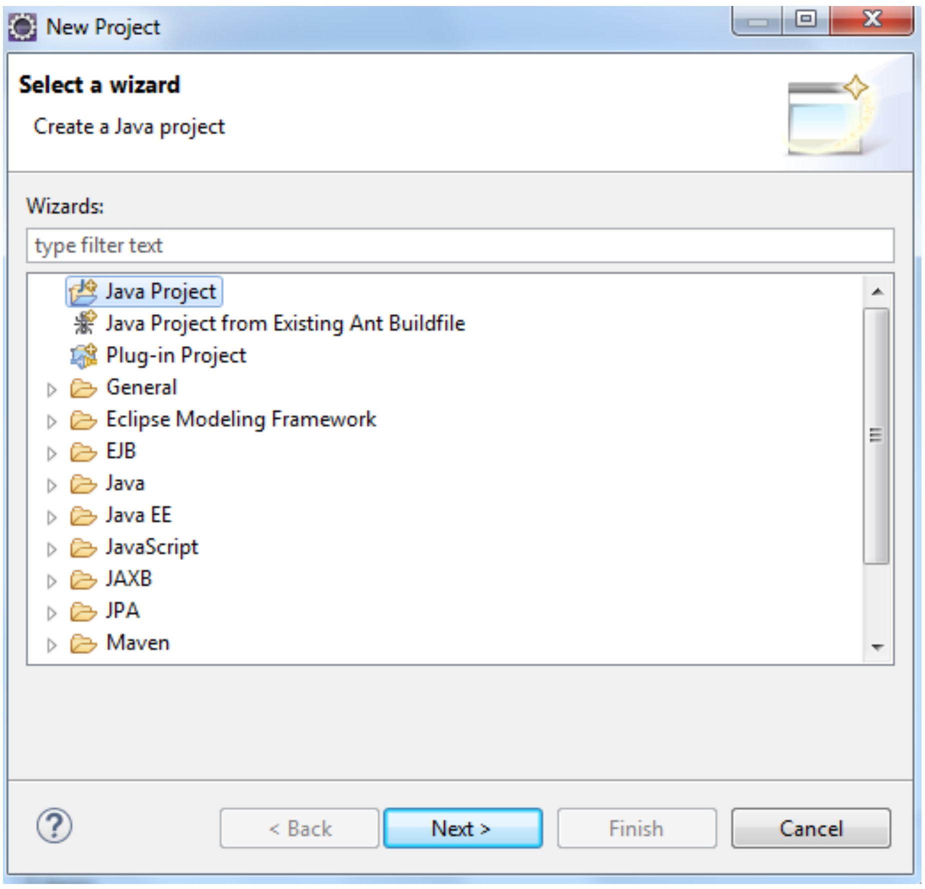
[✓] Inline finally blocks (larger class files, but improved performance)

[ ] Store information about method parameters (usable via reflection)

Restore Defaults | Apply

OK | Cancel

- Create a new Java Project called “CS265Lab1” as follows:
  - select File->New-> Project (just this word on it’s own)
  - select the Java Project Wizard (you may have to expand Java to see this)


New Project

Select a wizard

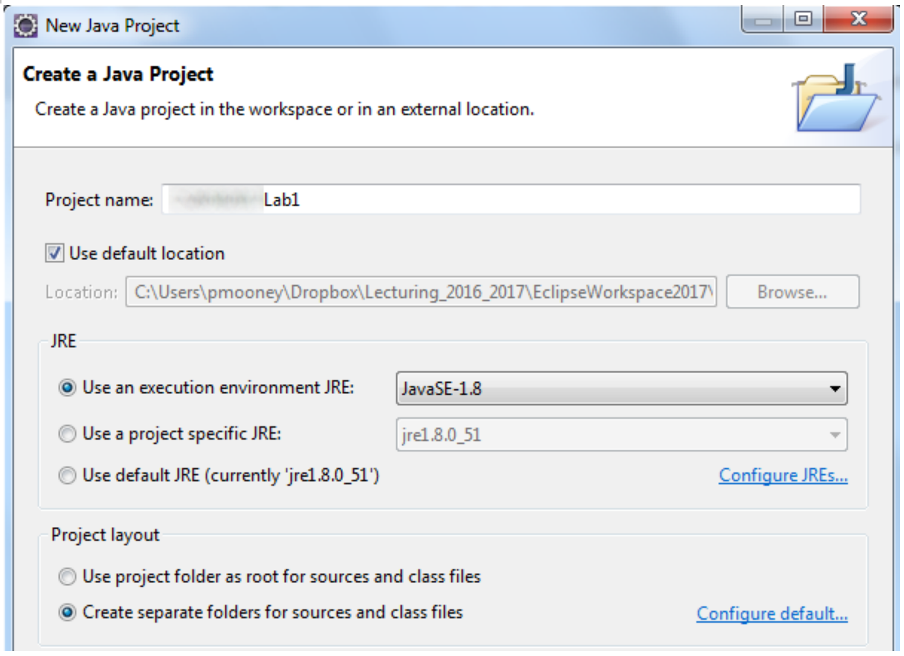
Create a Java project

Wizards:

type filter text

- Java Project
- Java Project from Existing Ant Buildfile
- Plug-in Project
- General
- Eclipse Modeling Framework
- EJB
- Java
- Java EE
- JavaScript
- JAXB
- JPA
- Maven

- click on **Next**
- enter **CS265Lab1** into the project name
- make sure "Create separate source and output folders" is selected
- Make sure the Use an Execution Environment is "JavaSE-1.8"


New Java Project

Create a Java Project

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

Project name: [redacted]Lab1

[✓] Use default location

Location: C:\Users\pmooney\Dropbox\Lecturing_2016_2017\EclipseWorkspace2017\

JRE

[✓ Use an execution environment JRE:] JavaSE-1.8

Use a project specific JRE: jre1.8.0_51

Use default JRE (currently 'jre1.8.0_51')

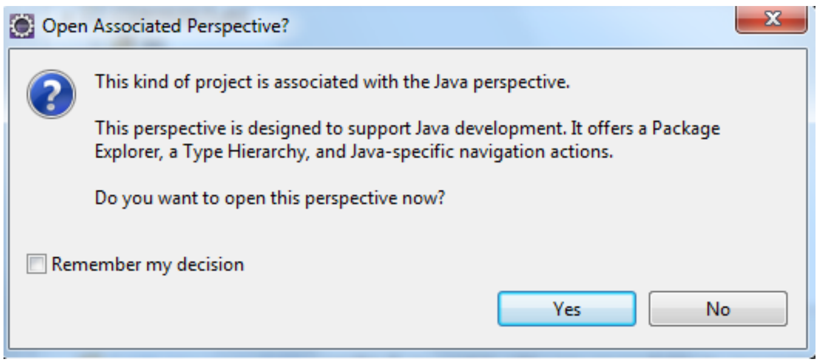
Project layout

Use project folder as root for sources and class files

[✓ Create separate folders for sources and class files]

- click on Finish

If you encounter a window about Java Perspective – click yes



This kind of project is associated with the Java perspective.

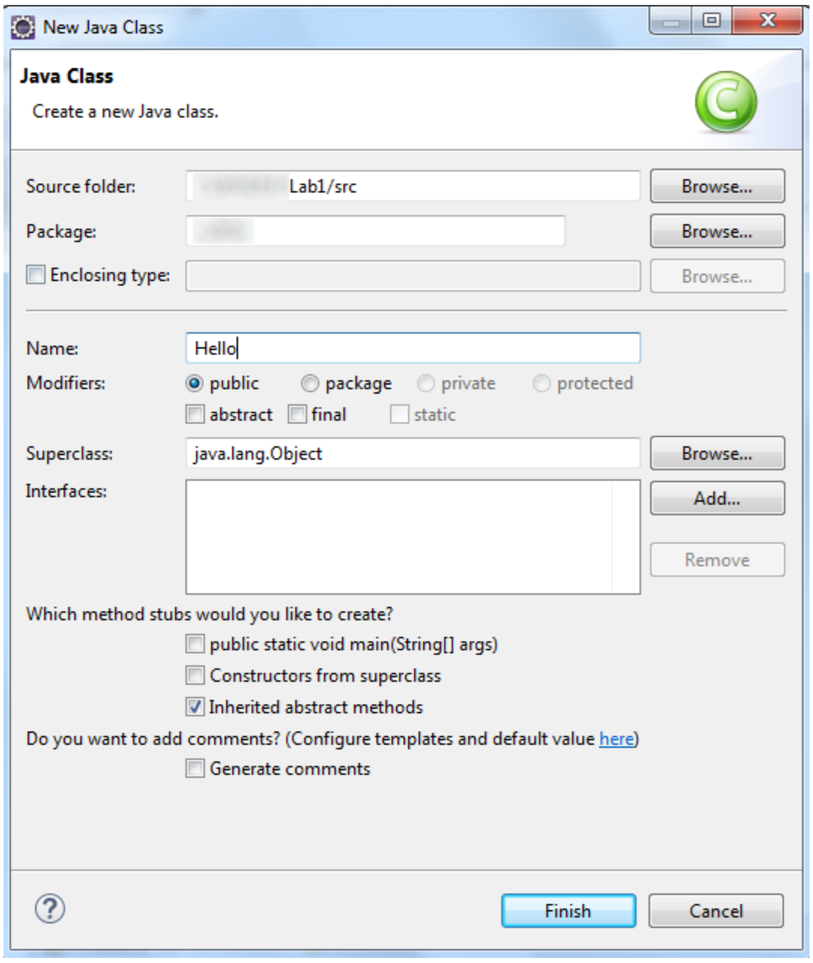
This perspective is designed to support Java development. It offers a Package Explorer, a Type Hierarchy, and Java-specific navigation actions.

Do you want to open this perspective now?

To organise our work better – we will create a new project for every lab – this separates the lab assignments nicely and makes you become more familiar with eclipse.

- Create a new Java package "**`cs265`**" as follows:
    - in the Package Explorer window (click on the tab in the left-hand panel if it is not showing), expand **CS265Lab1** (by clicking on the ▶ icon)
    - right click on **src**, and select New->Package
    - enter **cs265** as the name **NOTE – We will use this as our package name for all our code during this module – this makes it easier for me to provide you with code on Moodle which you can then copy and paste.**
    - and click on Finish

- In the cs265 package, create a new Java Class "Hello" as follows:
    - right click on cs265 (under **src** in the **CS265Lab1** project)
    - select New->Class
    - enter the name **Hello**
    - make sure only the **Public** modifier is selected
    - make sure none of the "would you like to create method stubs" options are selected
    - make sure generate comments is not selected
    - click on Finish


New Java Class

Java Class

Create a new Java class.

Source folder: [redacted]Lab1/src

Package: [redacted]

☐ Enclosing type:

Name: Hello

Modifiers: ● public ○ package ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

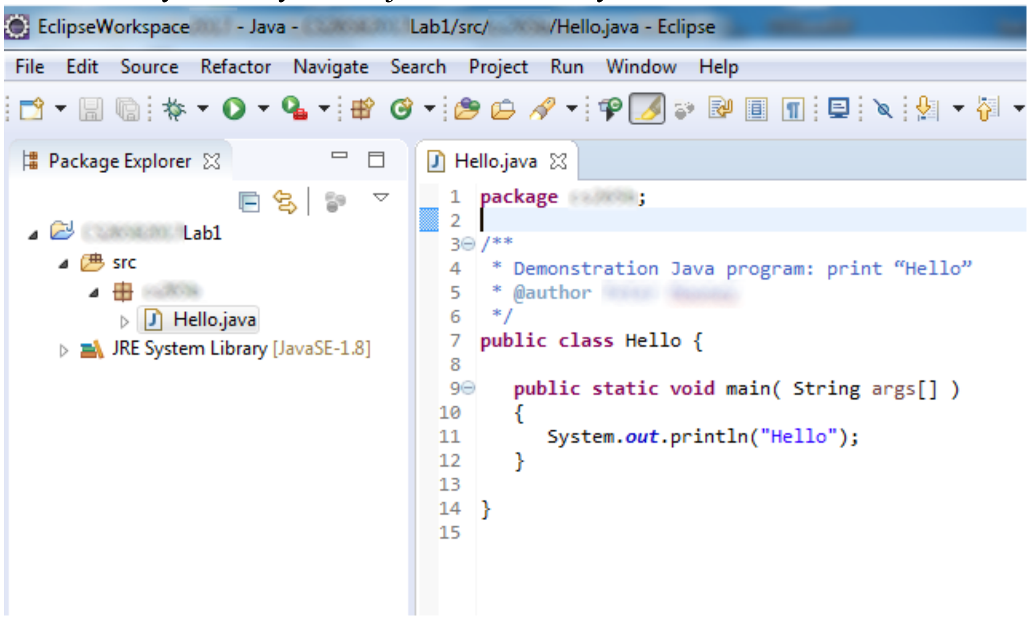
☐ Generate comments

- The source code should appear in a new Code Tab called "Hello.java" – modify it so that it matches the following code (you can cut and paste the source code from below or from Moodle– make sure to put your own name in every program you write):

```
package cs265;

/**
 * Demonstration Java program: print "Hello"
 * @author put your name here
 */
public class Hello {

   public static void main( String args[] )
   {
      System.out.println("Hello");
   }

}
```

- Save your work (use either control-S or File->Save)

- Note: the special comments in the source code are called **Javadoc** (you can view them in Eclipse, under the Javadoc tab, by clicking on the class name Hello in the source code window). Always include your name in every java program, and I recommend you always write javadoc for every method.


EclipseWorkspace[redacted] - Java - [redacted]Lab1/src/[redacted]/Hello.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
[redacted]Lab1
src
[redacted]
Hello.java
JRE System Library [JavaSE-1.8]
Hello.java

```java
package [redacted];

/**
 * Demonstration Java program: print "Hello"
 * @author [redacted]
 */
public class Hello {

    public static void main( String args[] )
    {
        System.out.println("Hello");
    }

}
```

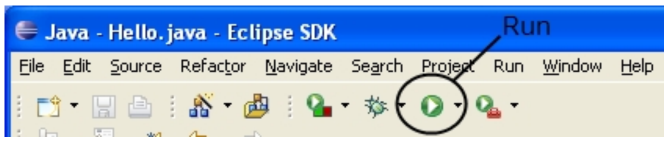
- **Now compile and run your program as follows**:

    - either:

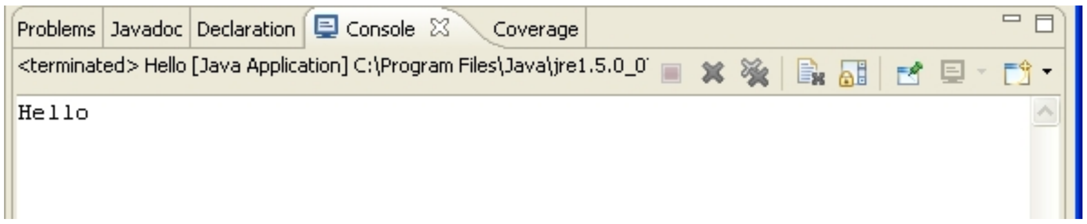
        1. right click on Hello.class in the Package Explorer, and select Run As->Java Application

    - or:

        2. left click on Hello.class in the Package Explorer, and click the Run button in the tope menu bar


Java - Hello.java - Eclipse SDK
Run
File Edit Source Refactor Navigate Search Project Run Window Help

- The Java “console” window should appear – probably at the bottom of your screen – showing the output from the program:



```
<terminated> Hello [Java Application] C:\Program Files\Java\jre1.5.0_0
Hello
```

- **Next – we will copy some more code into our Java code here.**
- Copy and paste these two IMPORT statements into your Java code – just under the package statement on the first line – or you can just type them.

```java
import java.net.InetAddress;
import java.net.UnknownHostException;
```

- Save your code – by pressing the save button again.
- Then inside your `public static void main(String[] args)` method – copy and paste the following code. You can put it before or after the existing System.out.println() statement.
- YOU HAVE TO TYPE THIS CODE IN YOURSELF – OR COPY IT FROM THE PDF here (which does not always work well!).

```java
        InetAddress ip;
         String hostname;
         try {
             ip = InetAddress.getLocalHost();
             hostname = ip.getHostName();
             System.out.println("My current IP 
Address is : " + ip);
             System.out.println("My current 
Hostname is : " + hostname);
             System.out.println("Please make note 
of these for Moodle later");

         } catch (UnknownHostException e) {

             e.printStackTrace();
```

```
    }
```

- Save your code – by pressing the save button again.
- Now – as before RUN your code.
- **Watch for the output in the console window at the bottom of the screen. You should take note of these – as you will need to enter them into Moodle for today's lab.**

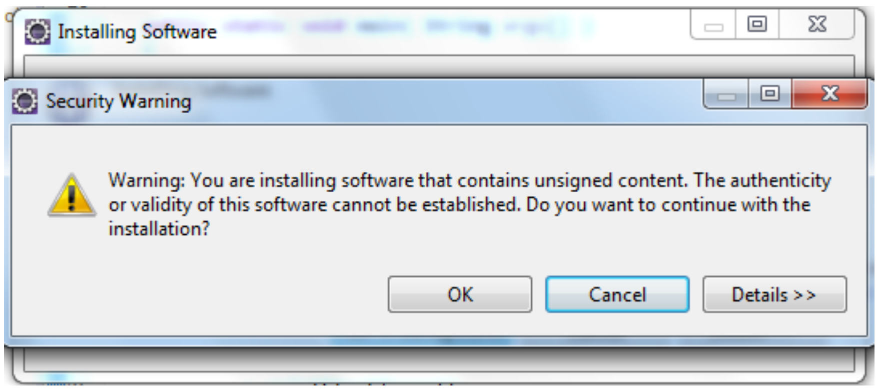
Additional Notes

- If prompted to save your source code, click on yes
- Just exit and restart Eclipse if it doesn't behave properly
- When you start Eclipse it reopens the same workspace you were using on the previous

INSTALLING TestNG in Eclipse.

You need to install the **TestNG** plugin for Eclipse next. You only need to do this **ONCE**. Follow the instructions below. Ask a demonstrator if you have problems.

- If not still running, start Eclipse, making sure to select the same workspace
- If you have code opened – make sure to SAVE your work now.
- Select Help->Install New Software
- Click on the ADD button
- Enter **TestNG** as the name, and http://beust.com/eclipse for the Location
- Click on OK
- Make sure "Show only the latest versions of available software" is selected (ticked)
- Make sure all the other options are not selected
- Select TestNG (click on the checkbox)
- Click NEXT
- Wait for the "Install Details" window to appear. The version should be provided – just pick whichever one is provided as this is usually the most recently available version
- It might be the case that this is already installed – in any case you will still click NEXT
- Click NEXT
- The Review Licenses page will appear. Click on the "I Accept…" option
- And then click on FINISH
- When a Security Warning about unsigned content popup appears, click on OK


Installing Software

Security Warning

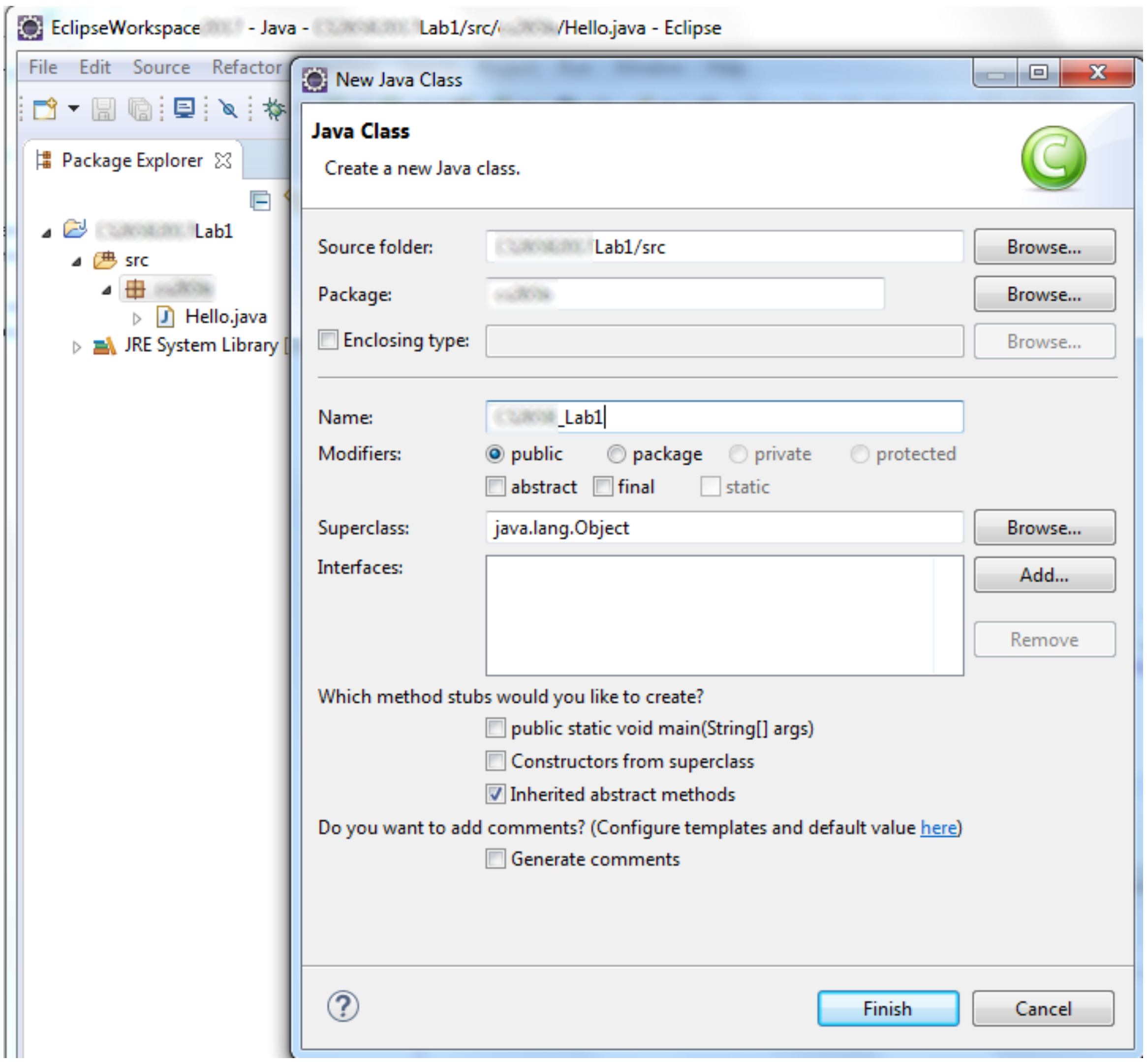
Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?

- If a Windows Security Alert appears, click on CANCEL
- You will next be asked whether to restart Eclipse – click YES
- When Eclipse restarts, you might have to reselect the Workspace (unless you have set the a folder as your default workspace – which you were advised to do)

Lab 1 Part B – Using TestNG for the first time.

On Moodle you will see the Java File **`CS265_Lab1.java`**

In the current project (**CS265Lab1**) create a new Java Class (as outlined above) with the name CS265_Lab1


EclipseWorkspace[redacted] - Java - [redacted]Lab1/src/[redacted]/Hello.java - Eclipse

New Java Class

Java Class

Create a new Java class.

Source folder: [redacted]Lab1/src

Package: [redacted]

[ ] Enclosing type:

Name: [redacted]_Lab1

Modifiers: [✓ public] package private protected

[ ] abstract [ ] final [ ] static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

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

[ ] Constructors from superclass

[✓] Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

[ ] Generate comments

Click Finish

When you have created the new class – you should COPY AND PASTE – EXACTLY AS IT APPEARS ON MOODLE – the contents of the `CS265_Lab1.java` file.
You should delete all of the contents of the class which Eclipse generated and then COPY AND PASTE the code from Moodle into the empty class file. Be careful not to have two package statements.

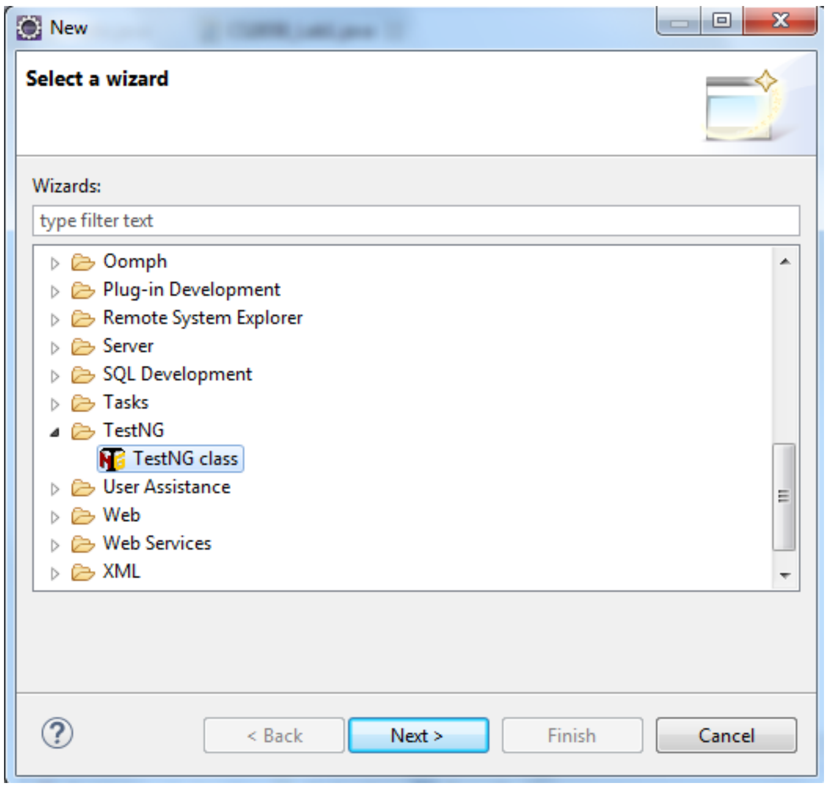
DO NOT MODIFY THIS FILE – this java program is the software you are testing. In Black Box Testing you can only run the code – you cannot change the code or modify it in this class.

`CS265_Lab1.java` is not an application – so this will not run. There are two methods in this class.

Now – we want to develop a TestNG test suite – for this java class – and in particular the two methods which are included within the class. This is where our software tests of the methods in CS265_Lab1 will be written.

So we must create a new TestNG class

You MUST RIGHT CLICK on the CS265_Lab1.java tab in the project explore and choose "New…." And then choose the "OTHER" option…

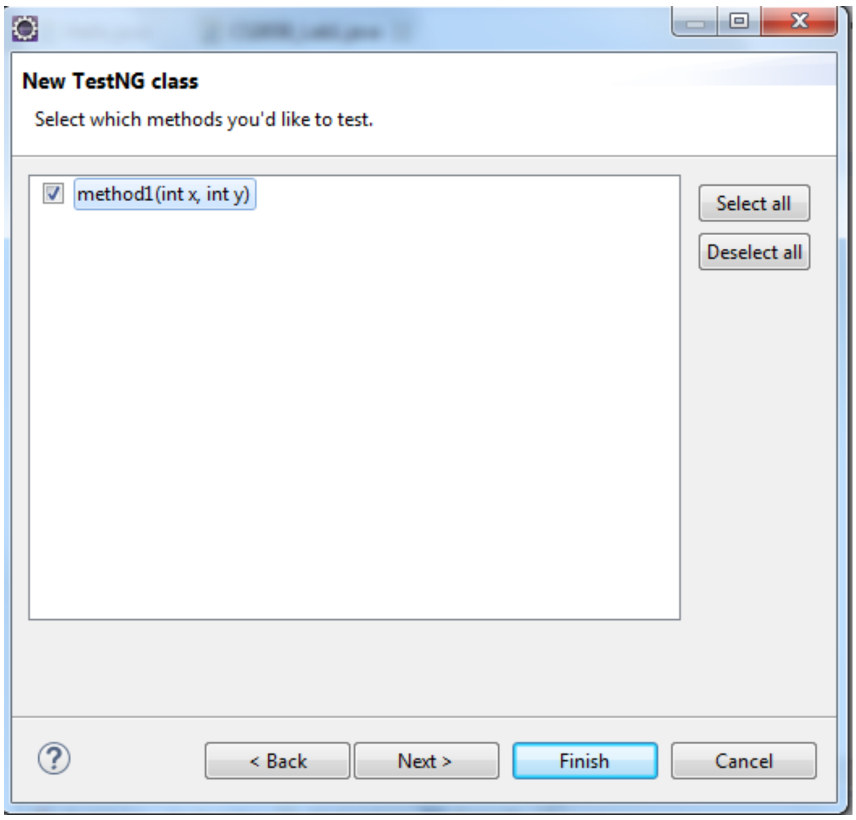


Scroll down to the TestNG tab and expand it – then click on TestNG class.

A new window will popup

You must select the methods (UNITS) that you want to test.

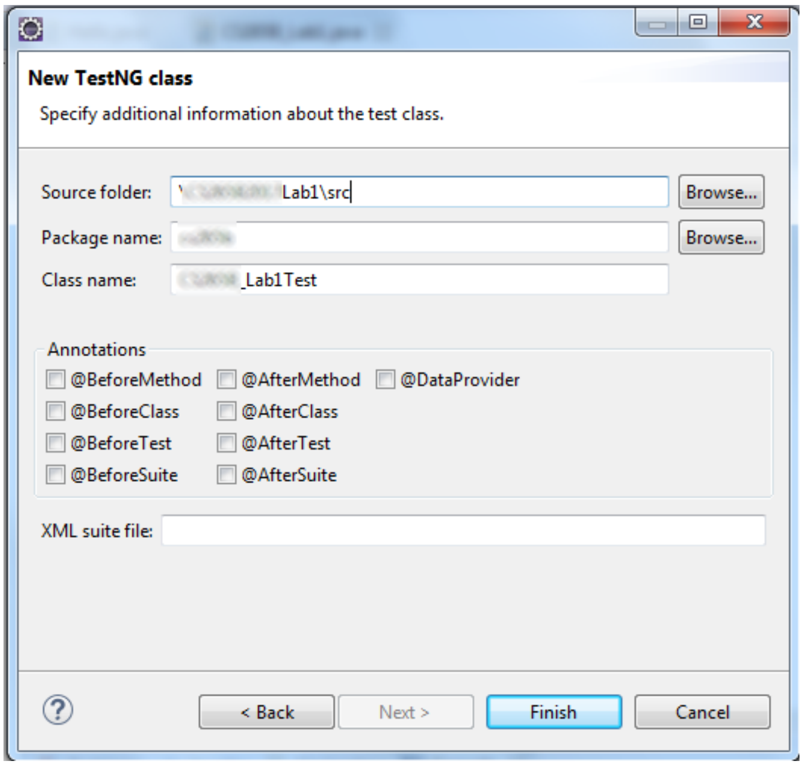
The screen below will have different unit names – Lab 1 select BOTH of the methods as we want to write tests for them both.


New TestNG class

Select which methods you'd like to test.

[✓] method1(int x, int y)

Click NEXT when you have both methods selected.


New TestNG class

Specify additional information about the test class.

Source folder: \[redacted]Lab1\src

Package name: [redacted]

Class name: [redacted]_Lab1Test

Annotations

[ ] @BeforeMethod [ ] @AfterMethod [ ] @DataProvider
[ ] @BeforeClass [ ] @AfterClass
[ ] @BeforeTest [ ] @AfterTest
[ ] @BeforeSuite [ ] @AfterSuite

XML suite file:

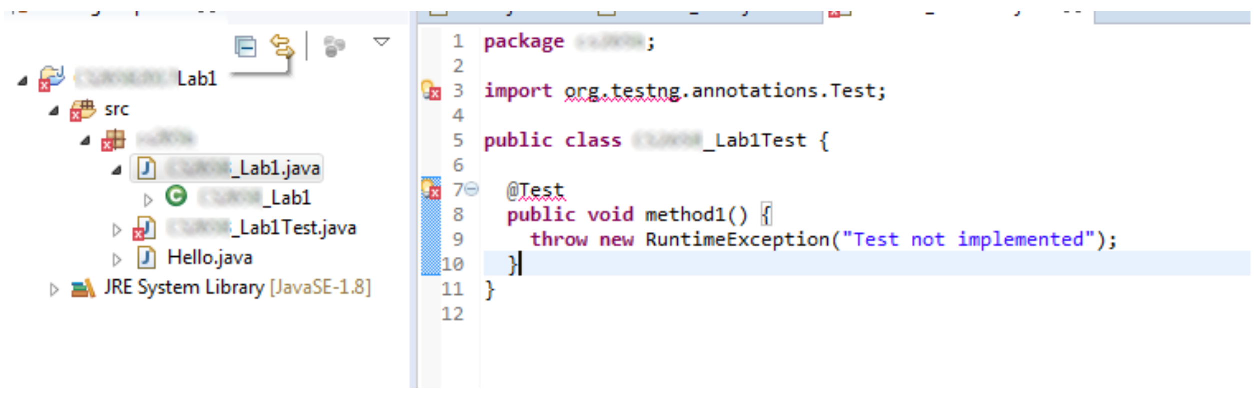
ENSURE that you make sure that the SOURCE FOLDER path ENDS with the name src…..

Do not change the package name.

Do not change the Class name.

Press the FINISH button.

You will see some errors from Eclipse (your errors will be similar to these below).

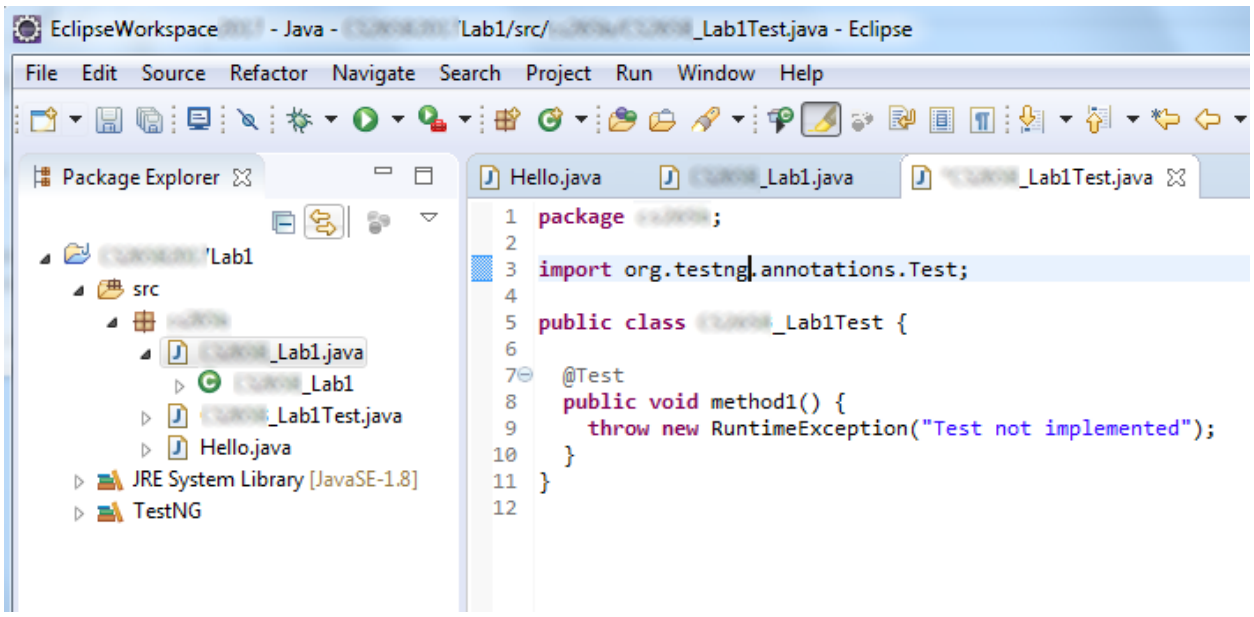


```
[redacted]Lab1
  src
    [redacted]
      [redacted]_Lab1.java
        [redacted]_Lab1
      [redacted]_Lab1Test.java
      Hello.java
  JRE System Library [JavaSE-1.8]
```

```java
package [redacted];

import org.testng.annotations.Test;

public class [redacted]_Lab1Test {

  @Test
  public void method1() {
    throw new RuntimeException("Test not implemented");
  }
}
```

Hover the cursor over the first error X box– you should see “`The import org.testng cannot be resolved`”. Left click on the red box, then double-click on the option to **add the TestNG library**. The errors should disappear.

You should be error free now.



EclipseWorkspace[redacted] - Java - [redacted]Lab1/src/[redacted]_Lab1Test.java - Eclipse

```
[redacted]Lab1
  src
    [redacted]
      [redacted]_Lab1.java
        [redacted]_Lab1
      [redacted]_Lab1Test.java
      Hello.java
  JRE System Library [JavaSE-1.8]
  TestNG
```

```java
package [redacted];

import org.testng.annotations.Test;

public class [redacted]_Lab1Test {

  @Test
  public void method1() {
    throw new RuntimeException("Test not implemented");
  }
}
```

DELETE THE method1 and method2 code as will be automatically generated by Eclipse/TestNG.

REPLACE with this software test below – NOTE the @Test MUST be included.

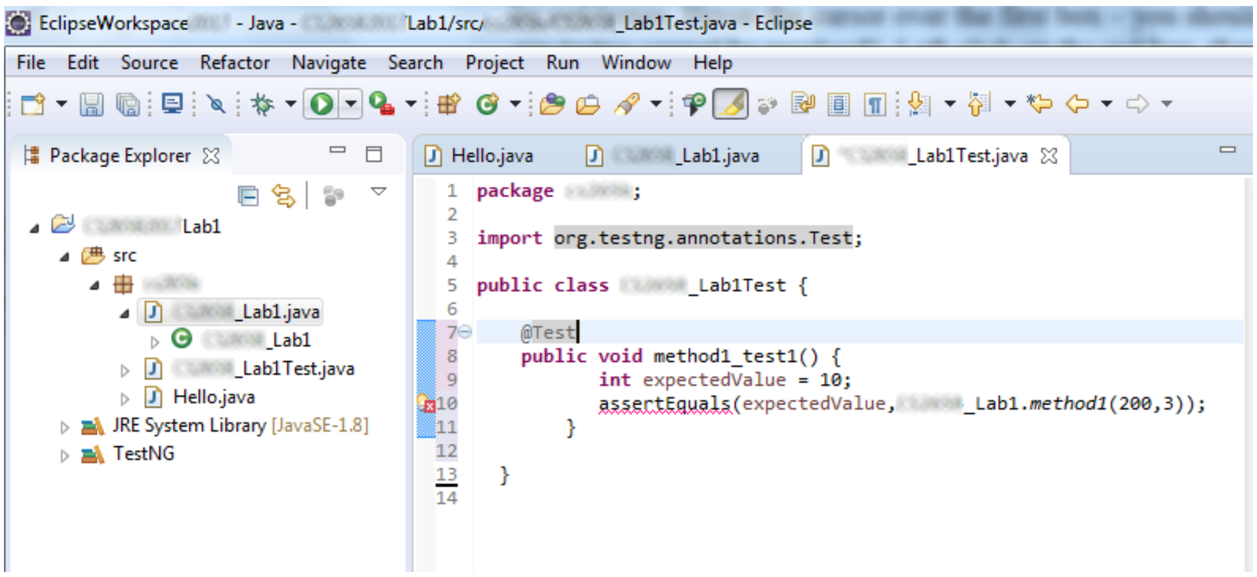
Save the file after these changes.

```java
@Test
public void method1_test1() {
```

```
                  int expectedValue = 70;
assertEquals(expectedValue,CS265_Lab1.method1(35,10));
     }
```

Notice that this TestNG Test (`method1_test1()`) is performing a software test of the Unit called method1 in the CS265_Lab1 class.

When you save the CS265_Lab1Test you will notice that there is an error like the one below (your file contents might look a little different as I will have different methods to test in the Lab 1 example)


EclipseWorkspace[redacted] - Java - [redacted]Lab1/src/[redacted]_Lab1Test.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
[redacted]Lab1
src
[redacted]
[redacted]_Lab1.java
[redacted]_Lab1
[redacted]_Lab1Test.java
Hello.java
JRE System Library [JavaSE-1.8]
TestNG
Hello.java
[redacted]_Lab1.java
*[redacted]_Lab1Test.java

```
package [redacted];

import org.testng.annotations.Test;

public class [redacted]_Lab1Test {

    @Test
    public void method1_test1() {
            int expectedValue = 10;
            assertEquals(expectedValue,[redacted]_Lab1.method1(200,3));
        }

   }
```

There is an error beside assertEquals.

Again click on it, and double-click the **Add static import org.testng.assertJUnit.assertEquals** option. The error should disappear.

IF the import suggestion does not appear/work, enter it manually, so that it reads:

```
org.testng.assertJUnit.assertEquals(expectedValue,CS265_Lab1.method1(35,10));
```

or similar, based on what data you use.

SAVE THE FILE

TASK – Run the Software Test.

Right click on the CS265_Lab1Test.java file in the Eclipse project browser pane on the left hand side and choose “Run As … “ and then choose TestNG Test.

Eclipse will probably pause for a moment or two while it runs the test.

In the background Eclipse is running every method in the `CS265_Lab1Test.java` file where it sees the @Test annotation.

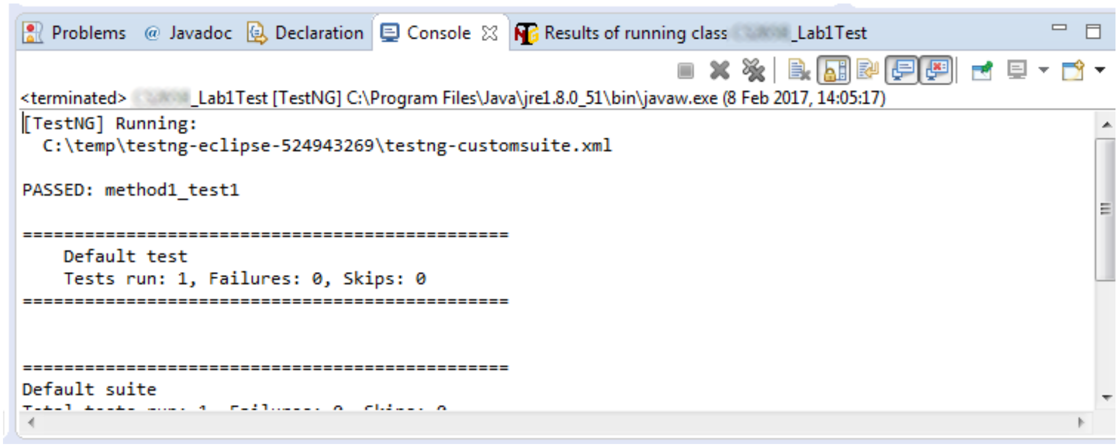
If you get a warning from Windows about the Firewall – choose "ALLOW ACCESS".

When you have run the TestNG test… you should look at the console tab at the bottom of your workspace.

Notice the way that we named our Test methods – we named them based on the test number (just our own numbering) and then the method from the CS265_Lab1 which was the unit under test. Using a simple convention like this makes it much easier for us to document and trace the tests we are writing.

This is the finish of this part of the lab.

Refer to the lab sheet below to complete the remainder of the Assignment for Lab 1.



```
<terminated> [redacted]_Lab1Test [TestNG] C:\Program Files\Java\jre1.8.0_51\bin\javaw.exe (8 Feb 2017, 14:05:17)
[TestNG] Running:
  C:\temp\testng-eclipse-524943269\testng-customsuite.xml

PASSED: method1_test1

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Failures: 0, Skips: 0
```

CS265 Lab 1 Assignment.

In the `CS265_Lab1Test.java` class – there are two methods called method1 and method 2.

You are asked to create at least 6 TestNG tests for each method in this class. <u>You do not need to use equivlance partioning or other methods.</u>

The goal is to "exercise the specification" and to write these software tests to test if the code works based on the specification you are given.

Make sure you also have tests that FAIL.

In the case of these two methods the specification is the text supplied as comments before each method.

Try to look at the specification and write your tests which relate directly to what is written in the specification. Do not be influenced by the code within the methods as in reality in a black box test scenario this would be hidden from you entirely.

To repeat: You should build your test cases based on the specification of both methods only – you should not concern yourself with the source code. You should also have tests that fail.

It is advisable that you plan out your test cases ON PAPER first – and then create them as TestNG tests.

Try to be creative in your tests.

Write your code for each test and be sure to save your file(s) as you are working.

When the tests are executed and ran (see above how to run as TestNG tests) – you should have a look at the bottom of the Eclipse window – click on the "Results of running … " tab – this will allow you to see what information Eclipse and TestNG generate about the tests.

After testing, you have to answer some questions as part of the Moodle Quiz for this lab. You must answer these questions to receive the points.

This lab is worth 2.5% CA for CS265.