Laboratory 6 – Week 8

## Built in Functions and Methods

### 7.1 Introduction

**Firstly, this worksheet *is* one of the worksheets from which your laboratory worksheets portfolio of work will be assessed.**

This laboratory worksheet covers the use of built in functions, methods and classes within the Java programming environment. This laboratory involves the creation of a number of Java programs. Make sure that you save any code you write. Also make sure you save any results or notes that you observe about your work.

Note that you are unlikely to complete this worksheet in just one laboratory session.

### 7.2 Preliminaries

Create a project in Eclipse called CS1702\_Lab6 and create a corresponding class (say CS1702\_Lab6). Try and organise your work (from the following exercises) into separate methods as we did in the previous worksheet.

### 7.3 Strings

Copy the following code into your project and run the program.

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

{

**double** number = 1.0/3.0;

DecimalFormat number\_format = **new** DecimalFormat(“#.##”);

System.*out*.println(number);

String formatted\_string = number\_format.format(number);

System.*out*.println(formatted\_string);

}

The DecimalFormat class enable us to format numbers (and other classes) in a variety of ways. In the above example we are formatting the number to two decimal places. Read up on this class in the **JavaDocs** since we will need it later.

Using the following variables (where specified):

x = 123.456, y = 17/3, z = √2, a = “Hello”, b = “World”

Write a program that performs the following programming tasks:

1. Display y to three decimal places
2. Display how many digits come before the decimal point and how many comes after a number. Test this program on x, y and z
3. Create a string c that consists of a in reverse concatenated with b, including a space between them
4. Search for the substring ‘ll’ in c and replace it for ‘ppp’

### 7.4 Mathematical Functions

In this module we will not be using or need many of the mathematical functions that Java supports. However the functions floor, round and ceil are very useful.

Write a method in a manner similar to the code example below that applies these three functions to a given parameter.

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

{

*RoundingTest*(10.2);

}

**private** **static** **void** RoundingTest(**double** x)

{

...

}

The output would be along the lines of:

For 10.200000 ceil=11.000000 floor=10.000000 round=10

Use the String class method format in your program. Read up on this method in the JavaDocs. An example of its use is given below:



Test your program on the following numbers: -100.1, ‑100.49, ‑100.5, -100.51, -100.9, 0, 100.1, 100.49, 100.5, 100.51, 100.9.

### 7.5 System Commands

The following snippet of code lists all of the files in a directory. The program uses the File class and the listFiles method of the File class. Read up on these programming constructs in the **JavaDocs** (as usual).

Implement the program and note how it works, run it a few times with different directories and verify that it lists the files correctly.

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

{

String dir\_name = “c:\\temp\\”; //Or another directory

File dir = **new** File(dir\_name);

File[] dir\_list = dir.listFiles();

**for**(**int** i=0;i<dir\_list.length;++i)

{

System.*out*.println(dir\_list[i].getName());

}

}

Modify the program in the following ways:

1. List whether each file is a directory or not.
2. In addition to the above, modify the program so that you can specify a filter on the type of file, e.g. “\*.txt”.

Test the program to ensure it is working correctly.

### 7.6 Random Number Generation

Implement and run the following code snippet. This program generates ten random integers. Note that the numbers range between a very large negative number and very large positive number. This might not be that useful for many applications. Often we want to generate uniformly distributed random numbers between two limits/bounds.

Random rand = **new** Random();

**for**(**int** i=0;i<10;++i)

{

System.*out*.println(rand.nextInt());

}

Modify, run and test the program as follows:

1. Generate random integers between -100 and +100.
2. Generate random integers between limits (pre-specified) *a* and *b*.
3. Generate random doubles.
4. Generate random doubles between zero and one.
5. Generate random doubles between limits (pre-specified) *a* and *b*.

Make sure your programs cater for any error conditions, e.g. where *b* > *a*.

### 7.7 File Handling

Implement the two programs that read and write to a text file from the lecture notes. Test them to see if they work. Base your solutions to the following two exercises on these programs.

**7.7.1 Reading Data**

From the text in Appendix A, create a text file called “Billy Goats.txt”. Write a program that takes as a parameter the name of a file and returns an integer containing the number of words in that file. Test your program on several of your own test files and the file you have created from the Appendix. Note that you can use *Microsoft Word* to count the number of words in a text file. The text in Appendix A contains 348 words (I think…).

### 7.7.2 Writing Data

Write a program that takes as input a number *n* and a text string filename and writes *n* lines to the file where each line is of the form: i: sqrt(i) sqrt(i) sqrt(i). The first column ranges from 1..*n* whilst the first square root is to one decimal place, the second is to two decimal places and the third is to three decimal places.

### 7.8 Creating a JAR File and Command Line Java

Create a separate project called JARExample. Add the following code (and class):

**public** **class** JARExample

{

**public** JARExample()

{

System.*out*.println("This Class does very little");

}

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

{

**for**(**int** i=0;i<args.length;++i)

{

System.*out*.println(args[i]);

}

}

}

**DO NOT ADD THIS CODE TO THE EXISTING PROJECT - CS1702\_Lab6.**

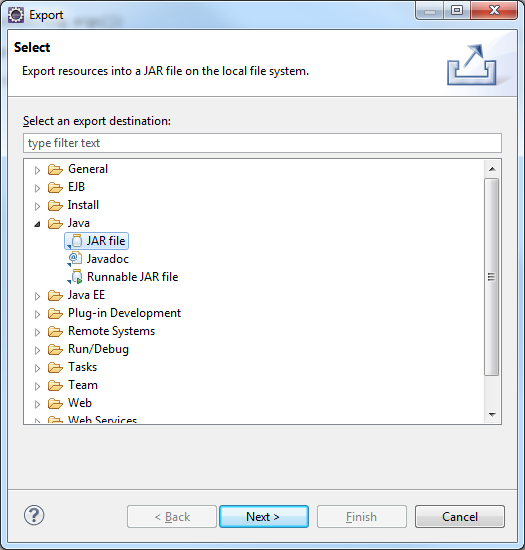
Run the program. You should find that it does absolutely nothing!

**7.8.1 Creating a JAR File**

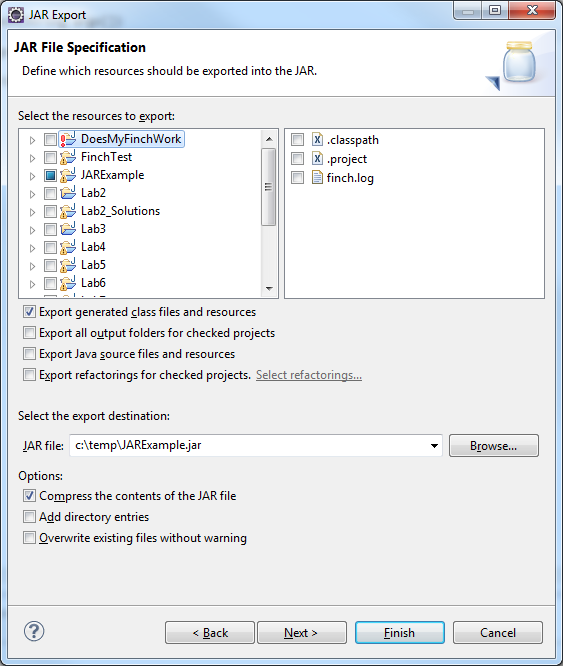
We are now going to create our own JAR file as described in the lectures.

Choose the menu options *File* and then *Export*. When the dialog box appears, expand the *Java* option and then select *JAR file*.

You should get to the point shown in the figure below:

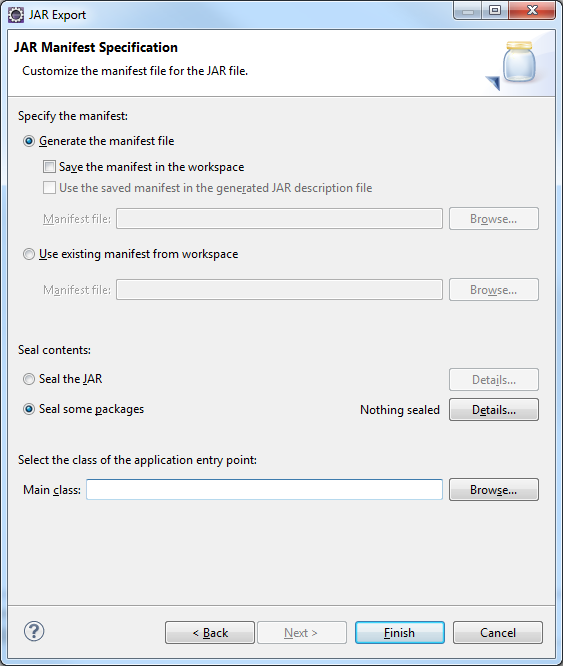


Click *Next>*, the following dialog box should appear.



In the section of the dialog box that says *Select the resource to export:*, choose the JARExample project. In the section of the dialog box that says *Select the export destination:* enter a location for the JAR file to be created in, for example: c:\temp\JARExample.jar. Do not change any of the other settings.

Click *Next*> and then *Next*> again, you should get the following dialog box. In the section of the dialog box that says *Select the class of the application entry point:* enter the name of the class we created, JARExample. Click the *Finish* button and verify that a JAR file has now been created in the expected location.

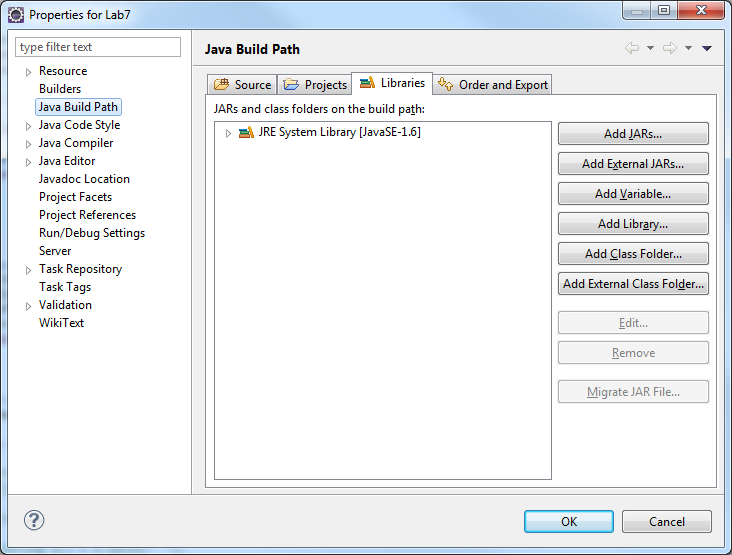


Go back to the CS1702\_Lab6 project and add the following line of code:

JARExample je = **new** JARExample();

You should get an error indicating that Eclipse cannot find the JARExample class. This is because we did not include the source code for in the JARExample project. However we are now going to tell Eclipse where to find the JAR file that we created.

Select the menu options *Project*->*Properties* and select *Java Build Path*. You should get the following dialog box. Choose *Add External JARs…* and select the JAR file that we created.



Click OK once the JAR file has been selected. Run the program and note the output. Modify the JARExample program so that it will produce some other text and repeat the process described above.

**7.8.2 Command Line Java**

Note that our JARExample program has a main method. This is for demonstrating the running of Java from the command line.

Start up a *Windows Command* Window. Type the following command:

java –jar c:\temp\JARExample.jar

Make sure you use the correct location of **your** JAR file.

Nothing should happen. Now type the following:

java –jar c:\temp\JARExample.jar A B C D

Note the output. The extra text after the JAR filename is put into a String array and passed into the main method. Hence the parameter in the definition of main. This is how command line programs that take parameters can be created.

Modify the JARExample program so that it produces output of the form:

Parameter 1 = A

Parameter 2 = B

Parameter 3 = C

Parameter 4 = D

when run from the *Command Window*.

### Appendix A. Sample Word Count Text

Once upon a time there were three billy goats, who were to go up to the hillside to make themselves fat, and the name of all three was “Gruff”. On the way up was a bridge over a cascading stream they had to cross; and under the bridge lived a great ugly troll, with eyes as big as saucers, and a nose as long as a poker. So first of all came the youngest Billy Goat Gruff to cross the bridge. “Trip, trap, trip, trap!” went the bridge. “Who's that tripping over my bridge?” roared the troll. “Oh, it is only I, the tiniest Billy Goat Gruff , and I'm going up to the hillside to make myself fat,” said the billy goat, with such a small voice. “Now, I'm coming to gobble you up,” said the troll. “Oh, no! pray don't take me. I'm too little, that I am,” said the billy goat. “Wait a bit till the second Billy Goat Gruff comes. He's much bigger”. “Well, be off with you,” said the troll. A little while after came the second Billy Goat Gruff to cross the bridge. Trip, trap, trip, trap, trip, trap, went the bridge. “Who's that tripping over my bridge?” roared the troll. “Oh, it's the second Billy Goat Gruff, and I'm going up to the hillside to make myself fat,” said the billy goat, who hadn't such a small voice. “Now I'm coming to gobble you up,” said the troll. “Oh, no! Don't take me. Wait a little till the big Billy Goat Gruff comes. He's much bigger”. “Very well! Be off with you,” said the troll. But just then up came the big Billy Goat Gruff. Trip, trap, trip, trap, trip, trap! went the bridge, for the billy goat was so heavy that the bridge creaked and groaned under him. “Who's that tramping over my bridge?” roared the troll. “It's I! The big Billy Goat Gruff,” said the billy goat, who had an ugly hoarse voice of his own. “Now I'm coming to gobble you up,” roared the troll. What happened next?