# assessed Lab 5

# Built-in functions and methods

## ASSESSMENT INFORMATION

**This worksheet *is* one of the seven assessed lab sheets.**

**It can be assessed within the next *5* weeks. Let me know *in advance* when you’d like to be assessed.**

**Do not forget to have it ‘signed off’ after you have been assessed.**

## 1 introduction

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.

## 2 Preliminaries

Create a project in Eclipse called CS1002\_Lab5 and create a corresponding class (say CS1002\_Lab5). Try and organise your work (from the following exercises) into separate methods as we did in the previous worksheet.

## 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;

System.*out*.println(number);

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

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

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

}

Is it showing you an error? Can you fix it?

The DecimalFormat class enables 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 as we will need it later.

<http://docs.oracle.com/javase/tutorial/i18n/format/decimalFormat.html>

To complete the following exercise you will also need a good understanding of the String class:

<http://docs.oracle.com/javase/6/docs/api/java/lang/String.html>

<http://docs.oracle.com/javase/tutorial/java/data/strings.html>

<http://docs.oracle.com/javase/tutorial/java/data/numberformat.html>

### Exercise 1

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

Hint: You may find the split method useful

1. Create a string c that consists of a in reverse concatenated with b, including a space between them
2. Search for the substring ‘ll’ in c and replace it for ‘ppp’

## 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.

### Exercise 2

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

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

{

*RoundingTest*(10.2);

}

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

{

//add your code here to apply floor, round and ceil on x

}

1. The output should look like this:

For 10.200000 ceil=10.000000 floor=10.000000 round=10

For this, you need to use the String class method format in your program. Read up on this method in the JavaDocs and Java tutorials (links above). An example of its use is given below:



1. Now use your roundingTest method 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.

## 5 Random Number Generation

Again, you need a good understanding of the Random class to be able to complete this exercise.

<http://docs.oracle.com/javase/6/docs/api/java/util/Random.html>

### Exercise 4

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

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

## 6 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).

<http://docs.oracle.com/javase/6/docs/api/java/io/File.html>

### Exercise 3

Implement the program and note how it works, run it a few times with different directories and verify that it lists the files correctly. You may need to add something for the code to work!

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

{

String dir\_name = “c:\\temp\\”;

//Or another directory. Be careful with the “” that MSWord uses.

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 it filters out (that is, doesn’t print) text files (.txt).

Test the program to ensure it is working correctly.

## 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.

### Exercise 5: Reading Data

Copy-paste the text in Appendix A, to create a text file called “the art of flying.txt”. Write a program that takes as a parameter the name of the file and returns an integer containing the number of words in that file. The text in Appendix A contains 558 words.

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.

### Exercise 6: Writing Data

1. Complete the following code 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).

i should take all values from 1 to n.

Follow the example below:

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

{

*writefile*(5,"c:\\temp\\ex6.txt" );

}

**public** **static** **void** writefile(**int** n, String filename)**throws** IOException{

FileWriter writehandle = **new** FileWriter(filename);

//complete the rest of the code to write in the file

}

The contents of the file should look like the following after running your code:

1 : 1.0

2 : 1.4142135623730951

3 : 1.7320508075688772

4: 2.0

1. Now modify the writefile method so that the contents of the files will look like that:

That is, the first square root is to one decimal place, the second is to two decimal places and the third is to three decimal places.

1: 1.0

2: 1.41

3: 1.732

4: 2.0000

### Exercise 7: Searching for the Longest Word

Write a Java program to find all of the longest word in a list of words entered by the user assuming that the user finishes entering the words in the list when s/he enters the word “done”.  
**Example-1:**  
{  
"cat",  
"flag",  
"green",  
"country",  
"resource"  
}  
Result: resource

**Example-2:**  
{  
"cat",  
"dog",  
"red",  
"is",  
"am"  
}  
Result: cat, dog, red

* The code in the main method will ask the user to enter the list of words.
* Then, the code responsible about finding the longest word/s in the list should be written in a user-defined method named (Longest\_Word) that is called from the main method and returns the longest word to the main method
* In case there is more than one word has the same length as the longest word (e.g. example 2 above, then the user-defined method will return all the words separated by a comma)

### Appendix A. Sample Word Count Text

There is an art, it says, or rather, a knack to flying. The knack lies in learning how to throw yourself at the ground and miss. Pick a nice day, and try it. The first part is easy. All it requires is simply the ability to throw yourself forward with all your weight, and the willingness not to mind that it's going to hurt. That is, it's going to hurt if you fail to miss the ground. Most people fail to miss the ground, and if they are really trying properly, the likelihood is that they will fail to miss it fairly hard. Clearly, it is the second part, the missing, which presents the difficulties. One problem is that you have to miss the ground accidentally. It's no good deliberately intending to miss the ground because you won't. You have to have your attention suddenly distracted by something else when you're halfway there, so that you are no longer thinking about falling, or about the ground, or about how much it's going to hurt if you fail to miss it. It is notoriously difficult to prize your attention away from these three things during the split second you have at your disposal. Hence most people's failure, and their eventual disillusionment with this exhilarating and spectacular sport. If, however, you are lucky enough to have your attention momentarily distracted at the crucial moment by, say, a gorgeous pair of legs (tentacles, pseudopodia, according to phylum and/or personal inclination) or a bomb going off in your vicinity, or by suddenly spotting an extremely rare species of beetle crawling along a nearby twig, then in your astonishment you will miss the ground completely and remain bobbing just a few inches above it in what might seem to be a slightly foolish manner. This is a moment for superb and delicate concentration. Bob and float, float and bob. Ignore all consideration of your own weight simply let yourself waft higher. Do not listen to what anybody says to you at this point because they are unlikely to say anything helpful. They are most likely to say something along the lines of "Good God, you can't possibly be flying!" It is vitally important not to believe them or they will suddenly be right. Waft higher and higher. Try a few swoops, gentle ones at first, then drift above the treetops breathing regularly. DO NOT WAVE AT ANYBODY. When you have done this a few times you will find the moment of distraction rapidly easier and easier to achieve. You will then learn all sorts of things about how to control your flight, your speed, your maneuverability, and the trick usually lies in not thinking too hard about whatever you want to do, but just allowing it to happen as if it were going to anyway. You will also learn about how to land properly, which is something you will almost certainly screw up, and screw up badly, on your first attempt. There are private clubs you can join which help you achieve the all-important moment of distraction. They hire people with surprising bodies or opinions to leap out from behind bushes and exhibit and/or explain them at the critical moments. Few genuine hitchhikers will be able to afford to join these clubs, but some may be able to get temporary employment at them.