COMP1006/1406 - Summer 2016 - Tutorial 1

Objectives: Basic Java programming: input, output, arrays, Strings, control flow.

Play Computer [10 minutes]

Each tutorial will have a play computer problem. In this problem, we will give you a snippet of code (or some pseudo-code), and your task is to execute the code on paper (hence, you play the role of the computer) and answer some questions about the final output or final variable values.

The code will not be working Java code. It might not be any computer language at all. However, you should still be able to read and understand what the code is doing (or trying to do). You will be asked to play computer on the midterm and final exam.

You should not spend more than 10 minutes on this problem in the tutorial.

```
function mystery(int n)
----------------------------------------
   integer x = 4
   integer y = 9

   integer count = 0

   for(i=0; i<x; i+=1)
      if( y + 2*i >= n)
         count += 1
         y = y - x
      else
         y = y + x
      end if
   end for

   print count, y
----------------------------------------
```

What is printed when this function is called with input 14? That is, what should be displayed if `mystery(14)` is called?

Bugs [20 minutes]

In each tutorial, we will provide you with one more Java programs that have some bugs in them. They might be compile-time errors, run-time errors, or logical errors (a flaw in the logic of the program). Your task will be to find the bugs and fix the programs.

➨ Download the Java program `Bugs1.java`. Compile and then run this program in DrJava.

The `Bugs1.java` program is supposed to be like the hello world program you are familiar with. The bug in this program is simple, but DrJava does not help very much in telling us what it is. Using an IDE, such a DrJava, can help to make programming easier. But, they are also software themselves and might have their own bugs associated with them or might not always work perfectly.

In this case, in the Interactions window, after compiling the program, trying typing "java Bugs1" instead of "run Bugs1". The use of "run" is just a feature of DrJava to make running programs simpler but it is not exactly the same as using "java Bugs1". The error message when calling the JVM directly (using java instead of run) in this case is more informative than DrJava's run command.

➨ Download `Bugs2.java`. Compile and then run this program.

Where exactly did the program crash? The compiler tells you this information. The error message you get should look like this

```
java.lang.NullPointerException
   at Bugs2.main(Bugs2.java:20)
   at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
   at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
   at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
   ...
```

The first line tells you what kind of error crashed the program and the second line tells you exactly where in the code the program crashed. Thus, we know it crashed in the class `Bugs2`, in the `main` method, on line 9.

Even when you know which line a program crashes on, we might not know exactly where in that line the problem is Try modifying the offending line (line 20) in the program to two lines:

```
   System.out.println("words[i] is " + words[i]);
   System.out.println("length of words[i] is " + words[i].length());
```

What can you determine about the bug in the original code? What was the problem? What is a potential solution?

Coding

1: Basic Command Line Programs

If you look at the `main` method in the HelloWorld program from the first class, you will notice that it takes an array of Strings as input. When we run the HelloWorld program, we can specify what that input is and use it in our program. Just keep in mind that command line arguments are always seen as Strings to Java (and so you may need to convert a string to a number if need be).

Download the `HelloHello.java` program from the course webpage.

`https://www.scs.carleton.ca/sites/default/files/course_page/HelloHello.java`

Read the class and make sure you understand what it is doing. Try running it with different command line arguments (like the examples provided in the header comment section of the file).

➨ Modify the HelloHello program so that it outputs an N by N block of Qs. Here, N is specified as the first command line argument.

For example, running the program with `java HelloHello 10` will output

```
QQQQQQQQQQ
QQQQQQQQQQ
QQQQQQQQQQ
QQQQQQQQQQ
QQQQQQQQQQ
QQQQQQQQQQ
QQQQQQQQQQ
QQQQQQQQQQ
QQQQQQQQQQ
QQQQQQQQQQ
```

and running `java HelloHello 2` will output

```
QQ
QQ
```

➨ Create new program called `Args.java` that does the following:

- If there there are no command line arguments when the program is executed, simply output to standard out (`System.out`) a message saying there were no command line arguments.

- If there were one or more command line arguments, the program should output to standard out a message saying how many command line arguments there are, and then on a single line, display each command line argument with a single space between each word.

For example, running `java Args cat dog eel` should output

```
There are 3 command line arguments
cat dog eel
```

The last line should consist of "cat", a space, "dog", a space, "eel" and a newline character. You might find using a combination of `System.out.print()` and `System.out.println()` useful.

Note: In Java, if you try to print an array like you would print a list in Python, the output will most likely not be what you expect. Try printing `args` (with something like `System.out.println(args);` and see what the output is.

➨ Modify your Args program to print to standard out, again on a single line ending with a newline, all the command line arguments in reverse order. For example, running `java Args cat dog eel` should output

```
eel dog cat
```

➨ Modify the `Args` program to do the following: output to standard out, on a single line ending with a newline, all the command line arguments in uppercase, followed by another single line with all the command line arguments in lowercase. For example, running `java Args Cat dOG EEl` should output

```
CAT DOG EEL
cat dog eel
```

Have a look at the String class API for help with creating an uppercase or lowercase version of a given string.

`https://docs.oracle.com/javase/8/docs/api/java/lang/String.html`

➨ Modify the `Args` program to do the following: output to standard out, on a single line ending with a newline, all the command line arguments that length greater than 2 but less than 5.For example, running `java Args doggy Cat kitttty dOG EEl 12 a abc` should output

```
Cat dOG EEl abc
```

2: Basic Input with Scanner

There are several ways in which we can get user input during the execution of our programs in Java. One simple way is to use a Scanner object.

https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html

➨ Download the HelloInput program. Read and run the code. Be sure you understand what is happening. Then, modify the code so that the user is prompted for their firts name, last name and age (in years. After the user enters this information, the program displays some information about that user (their name and how many years before they can retire, at 65). For example, input and output might look like

```
Enter your first name : Cat
Enter your last name : Kittenish
Enter your age : 3
Hi Cat, you can retire in 62 years
```

Extra Coding

➨ The Math class provides many useful static functions for us to use.

https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

The random method returns a double between 0.0 (inclusive) and 1.0 (exclusive). We can use this to simulate a coin flip as follows:

```
if( Math.random() < 0.5){
   // heads
}else{
   // tails
}
```

Write a program that prompts the user for three inputs: two integers and a double (denote them n, m and p). Your program should create a 2-dimensional array (n X m) of chars. Populate your array with either 'x' or 'o', where an 'x' is placed with probability p and an 'o' is placed otherwise. You can assume $0 \leq p < 1$.

For example, running your program with input 10, 11 and 0.5 will create an array (10 rows and 11 columns) where each element is an 'x' with probability 1/2.

Your program should create the array and then display it nicely to the screen.