CSCI 1101- Computer Science II
Laboratory No. 1

Welcome to your first lab in CSCI 1101. The lab consists of two parts. The first part is meant to get you back on track on essential java concepts that you learned in CSCI 1100. It will also help bring new students in 1101 up to speed in basic java programming.

The second part introduces you to the basics of object-oriented programming that was discussed in the lectures. For each exercise, your task is to first write the class that defines the given object. Compile it and check it for syntax errors. Then write the "demo class" with the main program that creates and uses the object.

You may use JGrasp, JCreator or any other IDE.

Note: All submissions are through Moodle. Log on to https://moodle.cs.dal.ca - use your cs login ID. Submissions are pretty straightforward. Instructions will be also be given in the first lab.
Deadline for submission: Saturday, January 19, 2013 at 12 noon.

PART 1: REVIEW OF JAVA PROGRAMMING (FROM CS 1100 CONCEPTS)

Exercise 1
(a) Write a java program to determine if a number is perfect. Your program must read the given integer and print whether it is perfect or not. A perfect number is defined as a positive integer (>0) such that the sum of its factors (including 1, but not the number itself), add up to the number. For example 6 (= 1+2+3) is a perfect number.

Use the following template:

```
import java.util.Scanner;
public class Ex1a
{
        public static void main(String[] args)
        {
                //rest of the code here
        }
}
```

(b) Modify the above program so that it uses a method isPerfect to determine if a given integer is perfect or not.

```
import java.util.Scanner;
public class Ex1b
{
        public static void main(String[] args)
        {
                //rest of the code here
        }
```

```
	public static boolean isPerfect(int n)
	{
		//code here
	}
}
```

(c) Modify 1(b) to print all perfect numbers from 1 to 10,000.

(d) Determine the execution time for the core part of the program (finding all perfect numbers between 1 and 10,000) in 1(c). You can use the following code template to obtain the execution time:

```
long startTime, endTime, executionTime;
startTime = System.currentTimeMillis();

//code segment

endTime = System.currentTimeMillis();
executionTime = endTime – startTime;
```

The above segment will give the time for executing the code segment in milliseconds.

Exercise 2: Write a program that accepts a String as input from the keyboard and determines whether or not the String is a palindrome. Ignore whitespace and punctuation (space, tab, single quote, double quote, exclamation, comma, question mark, hyphen, semi-colon, colon, period). Ignore case as well. For example, all these inputs are palindromes:

```
A man, a plan, a canal, Panama.
Was it a car or a cat I saw?

"If I had a hi-fi!!"
```

Use the following template:

```
public class Ex3
{
	public static void main(String[] args)
	{
		//code here
	}
	public static boolean isReverse (String line)
	{
		//code here
	}
```

```
        //add other methods if required
}
```

PART 2: INTRODUCTION TO OBJECTS AND CLASSES

Exercise 1 (from class notes)
To help you get started and also to review the basic concepts of object-oriented programming, this exercise takes you through the complete process. We will review the Rectangle class that we discussed in class.

1 (a) Write a java class definition for a rectangle object. The object should be capable of setting its length and width and computing its area. Use this to create two rectangle objects of dimensions 10 X 30 and 20 X 25, respectively. Print their areas.

Here is the Java class file (Rectangle.java). Compile it.

```
public class Rectangle
{
	private int length;
	private int width;

	public Rectangle()
	{
	}
	public void setLength(int l)
	{
		length = l;
	}
	public void setWidth(int w)
	{
		width = w;
	}
	public int getLength()
	{
		return length;
	}
	public int getWidth()
	{
		return width;
	}
	public int findArea()
	{
		return length*width;
	}
}
```

Now you can write a tester program called RectangleDemo.java to create and use Rectangle objects. Compile and run it.
Warning! If you cut and paste this code, it may result in errors (for example, the double quotes will not be compiled properly).

```
public class RectangleDemo
{
	public static void main(String[] args)
```

```
      {
            Rectangle rect1, rect2;

            rect1 = new Rectangle();
            rect2 = new Rectangle();
            rect1.setLength(10);
            rect1.setWidth(30);
            rect2.setLength(20);
            rect2.setWidth(25);

            System.out.println("Area of the first rectangle: " +
rect1.findArea());
            System.out.println("Area of the second rectangle: " +
rect2.findArea());
      }
}
```

1 (b) Now make the following changes:

- **In the Rectangle class, use the keyword this in the setLength and setWidth methods**
- **In the Rectangle class, add another constructor that accepts the length and the width and sets the attributes.**
- **In the Rectangle class, add a toString method to print the length and width of the rectangle like this:**
  `Length:  10      Width:  30`
- **Modify the RectangleDemo program to read user given length and width from the keyboard, create the Rectangle object and print its dimensions (using the toString method) and the area (using the findArea method):**
-

```
Enter length and width: 45 60

Length:  45      Width:     60
Area: 2700
```

Exercise 2: Design a class named Fan to represent a fan. The class contains:

- A String data field named speed that specifies the speed of the fan (SLOW, MEDIUM or FAST).
- A boolean data field named on that specifies whether the fan is on
- A double data field named radius that specifies the radius of the fan.
- A String data field named colour that specifies the colour of the fan.
- A no-arg constructor that creates a fan.
- Get and set methods for all four data fields.
- A toString method that returns a string description of the fan. If the fan is on, the method returns the fan speed, colour, and radius as a nicely represented string. If the fan is not on, the method returns the fan colour and radius along with the string "The fan is off".

Write a test program that creates two Fan objects. Assign maximum speed, radius 10, colour yellow to the first Fan object and turn it on. Assign medium speed, radius 5, colour blue, and off status to the second fan object. Display the objects by invoking their toString methods.

Exercise 3: Your friend who works at the superstore is urgently in need of your help with some basic sales records. Consider a class Sales that keeps track of the sales of one item in the store. An object of this class will have the attributes

- Name of the item
- Cost of the item
- Bulk quantity (to qualify for discount)
- Discount (percentage)
- Number sold
- Total amount
- Total discount

and the following methods

- Constructor that sets the cost, bulk quantity and the discount
- Get and set methods for cost, bulk quantity and discount
- `registerSale(int n)` records the sale of n items. If n is larger than the bulk quantity, the cost will be reduced by the discount.
- `displaySales` displays the name of the item, number sold, the total amount, and total discount.

Ignore sales tax.

Implement the class and test it. For example, for
Item: Shampoo
Cost: 2.50
Bulk quantity (to qualify for discount): 4
Discount: 10 percent
Number sold: 10

The output would be:
Shampoo
Number sold: 10
Total amount: $ 22.50
Total discount: $ 2.50

As another example, for
Item: Toothpaste
Cost: 1.99
Bulk quantity: 20
Discount: 15
Number sold: 10

The output would be:
Toothpaste
Number sold: 10
Total amount: 19.90
Total discount: 0