# CSCI 1101 – Winter 2017
# Laboratory No. 5

*This lab is a continuation of the concepts of object-oriented programming. The first question on the lab is a repeat of Exercise 3 from last week. The second question uses 1-d arrays as instance variables. The third question is a revision on 2-d arrays (useful for Assignment 2).*

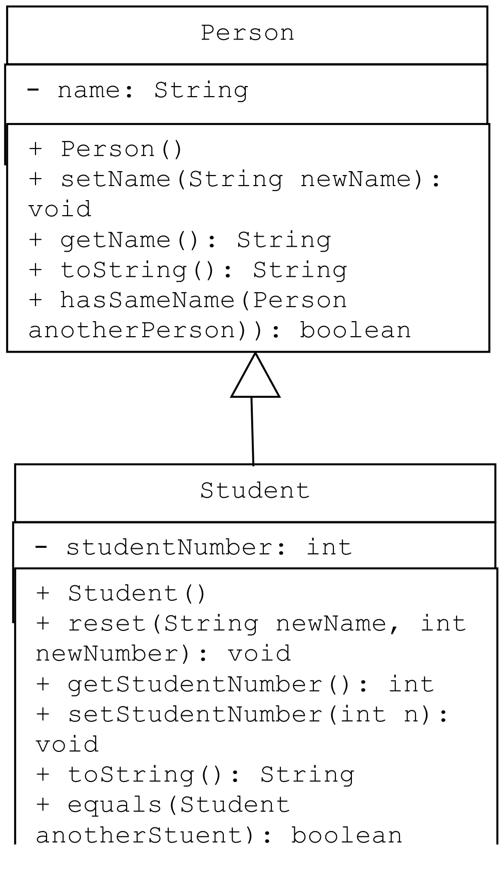
1. *All submissions must be made on Brightspace (dal.ca/brightspace).*
2. ***Submission deadline is 11.55 p.m. (5 minutes to midnight) on Saturday, February 18th, 2017.***
3. *Put the java source code files and the text outputs for each exercise in a folder. Zip the folder into one file and submit the zip file.*

***4. Marking Scheme:***
*Each exercise carries 10 points. Your final score will be scaled down to a value out of 10.*
*Working code, Outputs included, Efficient, Comments included: 10/10*
*No comments: subtract one point*
*Unnecessarily long code and inefficient program, improper use of variables: subtract one point*
*No outputs: subtract two points*
*Code not working: subtract up to six points depending upon the extent to which the program is incorrect.*

***5. Error checking****: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.*

***6. Testing your code and generating the outputs:*** *If the test data is included in the question, use that to test your classes. In addition, test it with two more input sets. Otherwise, create your own test data and run your program for at least 3 input sets such that they cover the range of results expected.*

**Exercise 1 (If you have already submitted the solution to this exercise last week, no need to submit it again):** The following figure shows a UML diagram in which the class `Student` is inherited from the class `Person`


Person
- name: String
+ Person()
+ setName(String newName): void
+ getName(): String
+ toString(): String
+ hasSameName(Person anotherPerson)): boolean
Student
- studentNumber: int
+ Student()
+ reset(String newName, int newNumber): void
+ getStudentNumber(): int
+ setStudentNumber(int n): void
+ toString(): String
+ equals(Student anotherStuent): boolean

Implement the two classes. Write a demo program that tests all the methods in the `Student` class as well as the inherited methods. Note: In the `toString` method in the `Student` class, use the method `getName()` to get the name.

**Exercise 2:** Write a Lottery class that simulates a lottery. The class has the following attributes:

- lotteryNumbers: array of five integers
- userArray: array of five integers
- matches: number of matches

and the following methods:

- Constructor: takes in no arguments. It should use the Random class to generate five random numbers in the range of 1 through 9 and puts them in the lotteryNumbers array
  Note: (int)(Math.random()*9+1) generates random numbers in the range 1 through 9.
- Method that accepts an array of five integers that represent a person's lottery picks.
- Method should compare the corresponding elements in the two arrays and finds the number of matches. For example, the following shows the lotteryNumbers array and the user array with sample numbers stored in each. There are two matching digits (numbers 9 and 3). Note that the digits should be in the same position as well.

lotteryNumbers array:

|  |  |  |  |  |
|---|---|---|---|---|
| 7 | 4 | 9 | 1 | 3 |

user array:

|  |  |  |  |  |
|---|---|---|---|---|
| 4 | 2 | 9 | 7 | 3 |

- Method printPrize that prints the lotteryNumbers array, the user array and the prize. The prize is determined according to the following rules:

|  |  |
|---|---|
| Grand Prize ($1 million): | all five digits match |
| $500,000 prize: | four out of five match |
| $1000 prize: | three out of five match |
| $10 prize: | two out of five match |
| $2 prize: | one out of five match |
| Sorry, nothing for you | none out of five match |

Demonstrate the class in a program that asks the user to enter five numbers. The program should display the lotteryNumbers array and the user's array and the number of digits that match. It should also display a message about the prize.
Note: With random numbers, it is really hard to win even a $10 prize.

**Exercise 3:** This exercise is on 2-d arrays. First read the following brief tutorial. This is similar to the tutorial that has been given in Assignment 2.

**Background on 2-d arrays in java**: 2-d arrays are very similar to 1-d arrays except that they have two subscripts or indices, one representing the row number and the other representing the column number.

For example,

```
int[][] a = new int[5][5];
```

creates a 2-d array with 5 rows and 5 columns.

You can process the 2-d array in the same manner. Instead of one for loop, we use a nested for loop. For example,

```
for(i=0;i<5;i++)
{
      for(j=0;j<5;j++)
            System.out.print(a[i][j] + "\t");
      System.out.println();
}
```
will print the contents of the array as a 5X5 matrix.

The boxes will be numbered as follows:

| | Column 0 | Column 1 | Column 2 | Column 3 | Column 4 |
|---|---|---|---|---|---|
| Row 0 | a[0][0] | a[0][1] | a[0][2] | a[0][3] | a[0][4] |
| Row 1 | a[1][0] | a[1][1] | a[1][2] | a[1][3] | a[1][4] |
| Row 2 | a[2][0] | a[2][1] | a[2][2] | a[2][3] | a[2][4] |
| Row 3 | a[3][0] | a[3][1] | a[3][2] | a[3][3] | a[3][4] |
| Row 4 | a[4][0] | a[4][1] | a[4][2] | a[4][3] | a[4][4] |

If you want to print all the elements in Column 2, for example:
```
for(i=0;i<5;i++)
            System.out.println(a[i][2]);
```

Similarly, the following will print the elements in Row no.3
```
for(j=0;j<5;j++)
            System.out.print(a[3][j] + "\t");
```
The following program creates a 2-d array, reads 25 integers from the keyboard, prints it as a 5X5 matrix and finds the sum of all elements. Note that the 25 integers can be entered all on a single line when they are read from the keyboard. Try it out.
```
import java.util.Scanner;
public class TwoDArray
{
      public static void main(String[] args)
      {
            int[][] a = new int[5][5];
            int i,j, sum=0;
            Scanner keyboard = new Scanner(System.in);
            for(i=0; i<5;i++)
                  for (j=0;j<5; j++)
                        a[i][j]= keyboard.nextInt();

            for(i=0;i<5;i++)
            {
                  for(j=0;j<5;j++)
                        System.out.print(a[i][j] + "\t");
                  System.out.println();
            }

            for(i=0;i<5;i++)
                  for(j=0;j<5;j++)
```

```
                        sum+=a[i][j];
                System.out.println("The sum of all elements is: " + sum);
        }

}
```

**Array initializer expression for 2-d arrays**

A 2-d array can be created using an array initializer expression. For example:

```
int[][] numbers = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

creates the following array

```
1     2     3     4
5     6     7     8
9     10    11    12
```

**Length field in a 2-d array**

A 2-d array has a length field that holds the number of rows, and each row has a length field that holds the number of columns.

The following program uses the length fields of a 2d array to display the number of rows, and the number of columns in each row. Try it out.

```
public class Lengths
{
        public static void main(String[] args)
        {
                int[][] numbers = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
                System.out.println("The number of rows is " + numbers.length);
                for(int index=0; index<numbers.length; index++)
                        System.out.println("The number of columns in row " +
                index + " is " +  numbers[index].length);
        }
}
```

**Now here is exercise 3**

Write a program that randomly fills in 0s and 1s into an 8 X 8 checker board, prints the board, and finds the rows, columns, or diagonal with all 0s or 1s. Use a 2-d array to represent the checker-board. Note: There is no object definition for this exercise. All you need to do is to write the class with a main method.
Here are two sample runs of the program:

```
Checker Board:
11001100
10101010
01010100
00000000
11110010
01010100
00001100

All zeros:
Row 3
Column 7
```

Checker Board:

```
10101000
10100001
11100011
10100001
11100111
10000001
10100111
00100001
```

All zeros:
Sub-diagonal right to left

**What to submit**: Source codes for the above exercises and sample outputs (at least three runs for each exercise).