Set 1

University of Rajshahi
Department of Computer Science and Engineering
1st Year Even Semester Examination-2020
Course: CSE1222-Object Oriented Programming Lab

Full Marks: 35 = 30 + 5(Viva) Time: 04 hours

Answer all five (05) questions

1. Add a method to the `Table` class below that computes the average of the neighbors of a table element in the eight directions shown in Figure (*Neighboring Locations in a Two-Dimensional Array)*

```
public double neighborAverage(int row, int column)
```

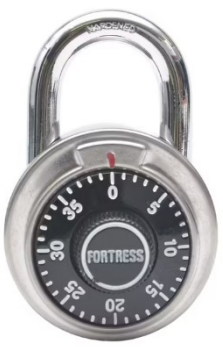
However, if the element is located at the boundary of the array, include only the neighbors that are in the table. For example, if row and column are both 0, there are only three neighbors.

[i - 1][j - 1]	[i - 1][j]	[i - 1][j + 1]
[i][j - 1]	[i][j]	[i][j + 1]
[i + 1][j - 1]	[i + 1][j]	[i + 1][j + 1]

```
public class Table{
     private int[][] values;
     public Table(int rows, int columns) {
          values = new int[rows][columns];
     }
     public void set(int i, int j, int n) {
          values[i][j] = n;
     }
}
```

2. Declare a class ComboLock that works like the combination lock in a gym locker, as shown here. The lock is constructed with a combination of three numbers between 0 and 39. The reset method resets the dial so that it points to 0. The turnLeft and turnRight methods turn the dial by a given number of ticks to the left or right. The open method attempts to open the lock. The lock opens if the user first turns it right to the first number in the combination, then left to the second, and then right to the third.

```
public class ComboLock{
     . . .
     public ComboLock(int secret1, int secret2, int secret3) { . . . }
     public void reset() { . . . }
     public void turnLeft(int ticks) { . . . }
     public void turnRight(int ticks) { . . . }
     public boolean open() { . . . }
}
```



3. Consider an interface

```
public interface NumberFormatter{
     String format(int n);
}
```

Provide four classes that implement this interface. A `DefaultFormatter` formats an integer in the usual way. A `DecimalSeparatorFormatter` formats an integer with decimal separators; for example, one million as 1,000,000. An `AccountingFormatter` formats negative numbers with parentheses; for example, –1 as (1). A `BaseFormatter` formats the number in base n, where n is any number between 2 and 36 that is provided in the constructor.

4. Write a program that reads a file containing text. Read each line and send it to the output file, preceded by line numbers. If the input file is

```
Mary had a little lamb
Whose fleece was white as snow.
And everywhere that Mary went,
The lamb was sure to go!
```

Then the program produces the output file

```
/*1*/ Mary had a little lamb
/*2*/ Whose fleece was white as snow.
/*3*/ And everywhere that Mary went,
/*4*/ The lamb was sure to go!
```

The line numbers are enclosed in /* */ delimiters so that the program can be used for numbering Java source files. Prompt the user for the input and output file names.

5. Write a program **`Find`** that searches all files specified on the command line and prints out all lines containing a reserved word. Start a new thread for each file. For example, if you call

```
java Find Buff report.txt address.txt Homework.java
```

Then the program might print

```
report.txt: Buffet style lunch will be available at the
address.txt: Buffet, Warren|11801 Trenton Court|Dallas|TX
Homework.java: BufferedReader in;
address.txt: Walters, Winnie|59 Timothy Circle|Buffalo|MI
```

Set 2

University of Rajshahi
Department of Computer Science and Engineering
1st Year Even Semester Examination-2020
Course: CSE1222-Object Oriented Programming Lab

Full Marks: 35 = 30 + 5(Viva) Time: 04 hours

Answer all five (05) questions

1. Consider the following class:

```
public class Sequence{
     private ArrayList<Integer> values;
     public Sequence() {
          values = new ArrayList<Integer>();
     }
     public void add(int n) {
          values.add(n);
     }
     public String toString() {
          return values.toString();
     }
}
```

Add a method

```
public Sequence append(Sequence other)
```

That creates a new sequence, appending this and the other sequence, without modifying either sequence.
For example, if `a` is `1 4 9 16`
and `b` is the sequence `9 7 4 9 11`
Then the call `a.append(b)` returns the sequence `1 4 9 16 9 7 4 9 11` without modifying `a` or `b`.

2. Implement a class `Robot` that simulates a robot wandering on an infinite plane. The robot is located at a point with integer coordinates and faces north, east, south, or west. Supply methods

```
public void turnLeft()
public void turnRight()
public void move()
public Point getLocation()
public String getDirection()
```

The turnLeft and turnRight methods change the direction but not the location. The move method moves the robot by one unit in the direction it is facing. The get Direction method returns a string "N" , "E" , "S" , or "W" .

3. The `System.out.printf` method has predefined formats for printing integers, floating point numbers, and other data types. But it is also extensible. If you use the S format, you can print any class that implements the Formattable interface. That interface has a single method:

```
void formatTo(Formatter formatter, int flags, int width, int precision)
```

In this exercise, you should make the BankAccount class implement the Formattable interface. Ignore the flags and precision and simply format the bank balance, using the given width. In order to achieve this task, you need to get an `Appendable` reference like this:

```
Appendable a = formatter.out();
```

`Appendable` is another interface with a method

```
void append(CharSequence sequence)
```

`CharSequence` is yet another interface that is implemented by (among others) the String class. Construct a string by first converting the bank balance into a string and then padding it with spaces so that it has the desired width. Pass that string to the append method.

4. Write a program **`Find`** that searches all files specified on the command line and prints out all lines containing a specified word. For example, if you call

```
java Find ring report.txt address.txt Homework.java
```

Then the program might print

```
report.txt: has broken up an international ring of DVD bootleggers that
address.txt: Kris Kringle, North Pole
address.txt: Homer Simpson, Springfield
Homework.java: String filename;
```

The specified word is always the first command line argument.

5. Write a program `WordCount` that counts the words in one or more files. Start a new thread for each file. For example, if you call

```
java WordCount report.txt address.txt Homework.java
```

Then the program might print

```
address.txt: 1052
Homework.java: 445
report.txt: 2099
```