Lab 7 --- Exercises

Create a project **Lab07**

Q1) The triangle inequality principle states that the sum of the lengths of any two sides of a triangle always exceeds the length of the third side.

Pythagoras' theorem states that the square of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the other two sides

Write a java class Triangle that has the three sides of the triangle as attributes. It also has the following methods:

- **isItTriangle**(): boolean. Returns true if the three sides accurately represents a triangle. (Any side cannot exceed the sum of both other sides). Otherwise returns false.
- **isItRight**(): boolean. Returns true if the all three sides represent a triangle and it is a right angle tiangle satisfying Pythagoras theorm. otherwise returns false.
- **longest**(): double. Returns the longest side.

Triangle
+ side1 : double + side2 : double + side3 : double
+ isItTriangle(): boolean + isItRight(): boolean + longets():double

Write a class **testTriangle** that:

Declare a Triangle object **t** and prompts the user to enter the lengths of three sides of a triangle.

displays a message indicating whether the lengths correctly represent a triangle, and if it is,

displays a message indicating whether the triangle is a right triangle.

Sample Run 1

```
Enter the three sides 4 6 8
It is a triangle
The longest side is 8.0
```

Sample Run 2

```
Enter the three sides 1 2 8
It is not a triangle
```

Q2) Write a Java class that represent a building. The building has a number of appartments for rent, some are normal and some are delux. The delux appartments are %20 more expensive than the normal ones. Here is the UML diagram:

Building
- apts: integer - normal: integer - delux: integer + rent: double
+ setApts(apts : int):void + howManyRented():int + rentApt(n: int, type: String): boolean + printInfo():void

As shown in the UML diagram, write the class Building that has the attributes:

- **apts**: Total number of appartments in the building.
- **normal**: The number of rented normal appartments in the building.
- **delux**: The number of rented delux appartments in the building.
- **rent**: The monthly rent of a normal appartment.

The methods of this class are:

- **setApts**(int **apts**): void. Sets the value of the attribute apts (since its private)
- **howManyRented**():int. Returns how many appartments are rented.
- **rentApt**(int **n**, String **type**):boolean. This method receives an integer n which is the number of appartments we like to rent and their type. Based on the value of type you should increase occupied normal or delux appartments accordingly. But first, you must check if you have n appartments available to rent, If you do then rent the appartments and reurn true, else, return false.
- **printInfo**():void. print all information about the building in an organized manner.

After writing class building, write a java class testBuilding to test your building class. In this class you should:

1. Declare an object b of class Building.
2. Read two building attributes: **apts** and **rent** from the user. No need to read normal and delux appartments numbers because they start as zero.
3. Ask the user how many **delux** appartments he would like to rent. If there is availability rent them and display a message stating that n appartments were rented. If not you should display the message "**Requested number of appartments exceeds availability**"
4. Ask the user how many **normal** appartments he would like to rent. If there is availability rent them and display a message stating that n appartments were rented. If not you should display the message "**Requested number of appartments exceeds availability**"
5. Print the building Information.
6. Feel free to try creating additional building objects, read their data and manipulate them.
7. exit.

SAMPLE RUN

```
Please enter Number of Appartments and rent amount :50 2000
How many delux appartments would you like to rent? 20
20 delux Appartments have been rented
How many normal appartments would you like to rent? 15
15 normal Appartments have been rented

================= Building Info ==========================
The Building has 50 appartment.
Only 35 have been rented.
15 normal appartments with rent = 2000.0 SR per month.
And 20 delux appartments with rent 2400.0 SR per month
==========================================================
```

SAMPLE RUN2

```
Enter number of appartments and rent amount :100 10000
How many delux appartments would you like to rent? 70
70 delux Appartments have been rented
How many normal appartments would you like to rent? 40
Requested number of appartments exceeds availability

================= Building Info ==========================
The Building has 100 appartment.
Only 70 have been rented.
0 normal appartments with rent = 10000.0 SR per month.
And 70 delux appartments with rent 12000.0 SR per month
==========================================================
```

Q3) Create the class **TV** with the following UML.

TV
-volumeLevel: int -channel: int -on: boolean
+turnOn(): void +turnOff():void + volumeLevelUp(int vol):void + volumeLevelDown(int vol):void +channelUp(int ch):void +channelDown(int ch):void +toString():String

Class **TV** contains the following data members:

- private **channel**: **int** between 0 and 99
- private **volumeLevel**: **int** between 0 and 8
- private **on**: **boolean** if on is true then TV is On, if on is false then its Off

Class **TV** contains the following methods:

- public void **turnOn**(); Sets the variable **on** to true.
- public void **turnOff**(); Sets the variable **on** to false.
- public void **volumeLevelUp**(int vol); Increases volumeLevel by vol amount. However, volumeLevel must always be between 0 -8 for example, if volumeLevel is 5 and you called volumeLevelUp(3) then new volumeLevel Should be 5+3 = 8. (But volumeLevel must not exceed 8)
- public void **volumeLevelDown**(int vol); Decreases volumeLevel by vol amount. However, volumeLevel must always be between 0 -8. for example, if volumeLevel is 5 and you called volumeLevelDown(3) then new volumeLevel Should be 5-3 = 2. (But volumeLevel must not go below zero)
- public void **channelUp**(int ch); Increases channel number by ch. However, channel number must be between 0 – 99 (i.e. if channel is 90 and you go up 15 channels your new channel should be 5)
- public void **channelDown**(int ch); Decreases channel number by ch. However, channel number must be between 0 – 99 (i.e. if channel is 10 and you go down 15 channels your new channel should be 95)
- public String **toString**(); Returns all current information of the TV as a String. IF the TV is on it should return a string indicating TV is on in addition to volume and channel information. IF TV is off, it should return a message saying that. An example of the returned string is given below.

```
TV is on and current channel is 5 and current volume
level is 8.
```

Create a class **testTV** that will do the following:

- Declares object **tv1** of class TV.
- Turn the tv on
- Set volume to 5
- Set channel to 20
- Print TV status
- Increase volume by 6
- increase channel by 90
- Print TV status
- Turn off TV
- Print TV Status
- Turn TV back on
- Lower volume by 7
- Decrease channel by 15
- Print TV status

Feel free to add your own changes to the TV volume and channels, print status after each change. (Note that java always initializes your int data members to 0, so the value of the channel and volumeLevel when you create the TV object is 0)

Sample Run

```
TV is On and current channel is 20 and volume level is 5
TV is On and current channel is 10 and volume level is 8
TV is Off
TV is On and current channel is 95 and volume level is 1
```

Q4) Design a class named **LinearEquation** for a 2 x 2 system of linear equations:

$$ax + by = e$$
$$cx + dy = f$$

A system of linear equations can be solved using Cramer's rule as following:

$$x = \frac{ed-bf}{ad-bc} \quad y = \frac{af-ec}{ad-bc}$$

The class contains:

- Data fields **a, b, c, d, e**, and **f**.
- A method named **isSolvable()** that **returns** true if ad - bc is not 0.
- Methods **solveX()** and **solveY()** that **return** the solution for the equation.

Draw the UML diagram for the class and then implement the class. Write a test program that prompts the user to enter **a, b, c, d, e**, and **f** and displays the solution. If **ad - bc** is **0**, report that **"The system has no solution."**

Sample Run 1

Enter a, b, c, d, e, f: 9 4 3 -5 -6 -21 ↵

x is -2.0 and y is 3.0

Sample Run 2

Enter a, b, c, d, e, f: 1 2 2 4 5 5 ↵

The system has no solution

UML

LinearEquation
+ a: double + b: double + c: double + d: double + e: double + f: double
isSolvable(): boolean solveX(): double solveY(): double

TestLinearEquation
main(): void