***Practical List For B.Sc. (Hons) Comp Sc II Sem Java Prog. 2018***

1. Consider a dieter's problem of wanting to "walk off" the calories consumed from eating one or more *Pizzas*. If you were the dieter and you really wanted to eat ***three*** Pizzas, how many miles would you need to walk to burn off the equivalent calories?

The formula to be used are as follows:

1. Calories consumed = (number of Pizzas) × (number of calories per Pizza
2. Miles to burn it off = (calories consumed) ÷ (calories burned off per mile)

Here are two **constants** to work from

1. Calories in one Pizza = **590** calories
2. Calories burned in one mile of walking = **93.5** calories per mile

Calculate the values for the four test cases given below.

|  |  |  |
| --- | --- | --- |
| **Test Cases** | | |
| **Big Macs Eaten** | **Calories Consumed** | **Miles to Walk It Off** |
| 0 |  |  |
| 1 |  |  |
| 3 |  |  |
| 9 |  |  |

Create a class Dieter which has three instance variables nPizzas ( number of Pizzas one has eaten), calorie\_gained and calorie\_burned. It has one method Calculate() which returns the number of miles one need to burn given the value for three instance variables. Create DieterDemo class to test Dieter class.

2. Create an array that can hold ten integers, and fill each slot with a different random value from 1-50 from keyboard. Display those values on the screen, and then prompt the user for an integer. Search through the array, and count the number of times the item is found. It should print zero if the number is not there. Use proper validation of input values. Implement Exception Handling if you can think of any reasonable exception that this program can throw.

3. Create a class point with two co-ordinate values x and y, with a parameterized constructor and a method that takes a point as its parameter and returns distance between the point passed as parameter and a point with which this method is invoked. The class also implements getX, getY, and setX and setY methods also. Implement Exception Handling if you can think of any reasonable exception that this program can throw.

4. Write a Console Program that reads in a list of integers, one per line, until a sentinel value of 0 (which you should be able to change easily to some other value). When the sentinel is read, your program should display the smallest and largest values in the list, as illustrated in this sample run: This program finds the smallest and largest integers in a list. Enter values, one per line, using a 0 to signal the end of the list.

? 17

? 42

? 11

? 19

? 35

? 0

The smallest value is 11 The largest value is 42. Your program should handle the following special cases:

• If the user enters only one value before the sentinel, the program should report that value as both the largest and smallest.

• If the user enters the sentinel on the very first input line, then no values have been entered, and your program should display a message to that effect.

* Implement Exception Handling if you can think of any reasonable exception that this program may throw.

5. Write a Console Program that reads in a number from the user and then displays the Hailstone sequence for that number. For example Hailstone sequence for number 17 looks like this:

This program computes Hailstone sequences.

Enter a number: 17

17 is odd, so I make 3n+1 = 52

52 is even, so I take half = 26

26 is even, so I take half = 13

13 is odd, so I make 3n+1 = 40

40 is even, so I take half = 20

20 is even, so I take half = 10

10 is even, so I take half = 5

5 is odd, so I make 3n+1 = 16

16 is even, so I take half = 8

8 is even, so I take half = 4

4 is even, so I take half = 2

2 is even, so I take half = 1

The process took 12 steps to reach 1. Implement Exception Handling if you can think of any reasonable exception that this program may throw.

6. Create a Date class with three integer instance variables named day, month and year. It has a parameterized constructor with three parameters to initialize the instance variables and a method daysSinceJan1(). It computes and returns the number of days since January 1 of the same year, including Jan 1 and the Day in the Date object ( Don’t forget leap years). Implement Exception Handling if you can think of any reasonable exception that this program may throw.

7. Create a Die class with one integer instance variable called sideUp. Give it a constructor and a getSideUp() method that returns the value of sideUp and a void roll() method that changes sideUp to a random value from 1 to 6.Then create a DieDemo class with a main method that creates two Die objects, rolls them and prints the sum of two sides up.

Use min + (int)(Math.random() \* ((max - min) + 1)) to generate random integer between [min, max] or 1 + randomNumbers.nextInt(6);

8. Write a Java program that can serve as a ***ending time*** calculator. The user enters the starting time in hours and minutes, a duration in total minutes, and your program will calculate and display the ending time (as hours:minutes).

For example, if an event starts at **2 30** and lasts **125** minutes, it will end at **4 35**

**Note**: To simplify the problem, assume military time (0..23) rather than standard time, in which you would need to worry about a.m. and p.m.

Begin by developing **test cases** for this program. Ending time should be expressed in military time (0..23)   
For example: 1:30 p.m. is expressed as 13:30 military time

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** All times are in military time | | | | |
| **Start Hours** | **Start Minutes** | **Duration  (in minutes)** | **Ending Hour (military time 0..23)** | **Ending Minute  (0..59)** |
| 1 | 30 | 120 |  |  |
| 11 | 45 | 60 |  |  |
| 12 | 40 | 90 |  |  |
| 10 | 0 | 240 |  |  |

Implement Exception Handling if you can think of any reasonable exception that this program may throw.

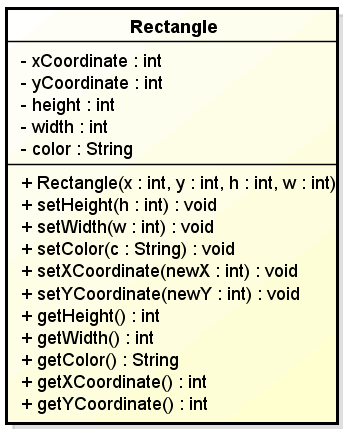
9. Write a menu driven program, which displays the options for calculating factorial of a given integer, to check if the given number is prime or not and to convert it into binary. It then takes user’s choice and acts accordingly.

10. Write a menu driven program to perform matrix addition, subtraction, multiplication, trace calculation, find if a given matrix is symmetric or not.

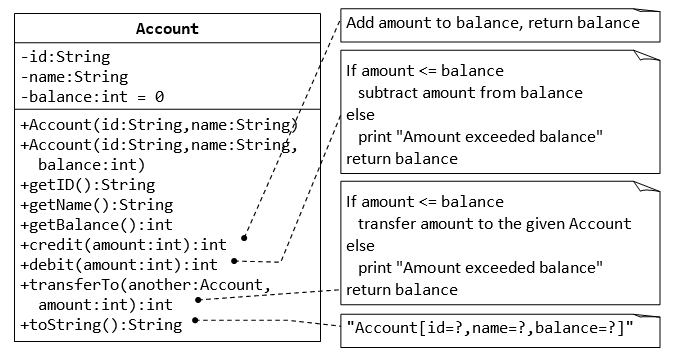
11. The UML class diagram for the **Rectangle** **class** is shown at right.

* There are **five fields** (instance variables), four of which are integers and one of which is a string
  + Always declare fields as **private**
* There is a **constructor** method that accepts four arguments:
  + the (x, y) coordinates, the height, and width
  + the constructor method should always assign a rectangle the default **color** black as "#000000"
  + constructor methods should always be declared as **public** and return no type
* ***Setter*** and ***getter*** methods for height, width, and color
  + Setter methods delared as **public** and return **void** (nothing)
  + Getter methods declared as **public** and return the **appropriate** type
* ***Setter*** and ***Getter*** methods for the xCoordinate and yCoordinate
* ***Calculate method calculates and returns the the length of the main diagonal of the rectangle.***

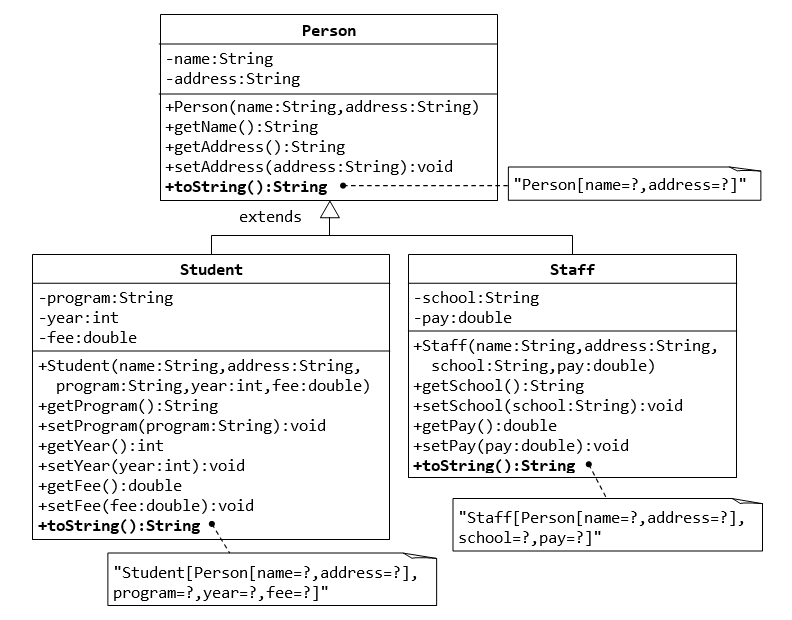
1. **Create** a new Java **class** named **Rectangle** and save as Rectangle.java
2. **Type** in the Java code to implement the **public** **Rectangle** **class** shown in the UML diagram below.



12. Implement the Account Class described below.



13. Implement the Superclass Person and its subclasses as described below:



14. Write a program that creates two threads and start them in main, where threads print integer between 4 to 0 in descending order and sleep between two consecutive prints for 100 ms. Main waits for the first thread to complete before starting the second thread. Use any of the method to create thread (runnable or extending the thread class).

15. Write an applet that creates a frame and implement mouseEvents on frame.

16 Write an applet that creates a frame and implement mouseEvents on applet window.

17. Write an applet that implements a keyEvents on its window.

18. Write a program to take an array of Strings and sort it in the ascending order. For example if the array consists of strings-

Ram, Apple, Beta, Gama, Theta then after sorting it will have-

Apple, Beta, Gama, Ram, Theta. Test your program for the case where an string appear more than once in the array.