

Karnataka Law Society's
GOGTEINSTITUTE OF TECHNOLOGY

Udyambag Belagavi-590008

Karnataka, India.



JOURNAL SUBMISSION

OBJECT ORIENTED PROGRAMMING WITH JAVA LAB

Submitted for the requirements of 3rd semester B.E. in CSE

For **“OBJECT ORIENTED PROGRAMMING
WITH JAVA LAB (18CSL38)”**

Submitted by

NAME	USN
Venkatesh G Dhongadi	2GI19CS175

Academic Year 2020-2021 (Odd semester)

Karnataka Law Society's

GOGTE INSTITUTE OF TECHNOLOGY

Udyambag Belagavi -590008
Karnataka, India.

Department of Computer Science and Engineering



Certificate

This is to certify that the Journal work titled **“OBJECT ORIENTED PROGRAMMING WITH JAVA LAB”** carried out by **Venkatesh G Dhongadi** bearing 2GI19CS175 is submitted in partial fulfilment of the requirements for 3rd semester B.E. in COMPUTER SCIENCE AND ENGINEERING, Visvesvaraya Technological University, Belagavi. It is certified that all corrections/ suggestions indicated have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of research work prescribed for the said degree.

Signature of Faculty:

Signature of HOD:

Date:

INDEX

Sl.No	Experiment title	Sign
1	Write a program to demonstrate the implementation of 2-dimension array.	
2	Write a program to demonstrate the implementation of class and its member methods.	
3	Write a program to demonstrate the implementation of parameterized: a. Methods. b. Constructor	
4	Write a program to demonstrate the implementation of inheritance.	
5	Write a program to demonstrate the implementation of method: a. Overloading. b. Overriding.	
6	Write a program to demonstrate the implementation of run-time polymorphism.	
7	Write a program to demonstrate the implementation of interface.	
8	Write a program to demonstrate the implementation of customized exception handling.	
9	Write a program to demonstrate the implementation of string handling.	
10	Write a program to demonstrate the implementation of JAVA swings.	

Title of the Experiment: Implementation of 2-dimension array

Experiment No. 1

Date: 22-09-2020

Problem Statement:

Write a Java program to accept IA marks obtained by five students in three subjects. The program should accept marks obtained by each student and display the total marks and the average marks. The average marks are computed as the average of best two marks obtained.

Objectives of the Experiment:

1. Learn declaration and initialization of variables and 2-D array in Java
2. Understand the use of 2-D array in a real-life application
3. Learn the usage of Looping constructs and control statements
4. Learn to Display the result in a readable/proper format

Program Source Code:

```
package tw1;

import java.util.Arrays;

import java.util.Scanner;

public class Tw1 {

    public static void main(String[] args) {

        Scanner in=new

        Scanner(System.in);

        //declaring arrays and variables
```

```

int sum=0;

float avg;

int array[][]=new int[5][3];

//reading input from user
for(int i=0;i<5;i++) {

    System.out.println("Enter the marks for student "+(i+1)+" : ");

    for(int j=0;j<3;j++) {

        array[i][j]=in.nextInt();

    }

}

//sorting to get best 2
marksfor (int i = 0; i < 5; i++)
Arrays.sort(array[i]);

//calculating average and printing
averagefor(int i=0;i<5;i++) {

    for(int j=1;j<3;j++) {

        sum+=array[i][j];

    }

    avg=sum/2.0f;

    System.out.println("\nAverage of best of two marks of student"+(i+1)+" :

    "+avg);sum=0;

}

}
}

```

Input: Marks entered for students are (28,26,30), (21,23,20),(25,20,23),(30,30,30),(15,17,15) respectively.

Output:

```
<terminated> Tw1 [Java Application] C:\Program Files\Java\jdk-14.0.2
Enter the marks for student 1:
28 26 30
Enter the marks for student 2:
21 23 20
Enter the marks for student 3:
25 20 23
Enter the marks for student 4:
30 30 30
Enter the marks for student 5:
15 17 15

Average of best of two marks of student1: 29.0

Average of best of two marks of student2: 22.0

Average of best of two marks of student3: 24.0

Average of best of two marks of student4: 30.0

Average of best of two marks of student5: 16.0
```

Outcomes of the Experiment: At the end of the laboratory sessions the students should be able to

1. Demonstrate the use of 2-D array in solving real-life problems.
2. Identify appropriate variables and their types
3. Identify appropriate looping constructs (for or do while)
4. Check if one loop will suffice or use nesting
5. Identify the control statements needed to meet the problem requirements.

Conclusion: From the given problem statement, we could identify the necessary variables of appropriate type, and looping/control statements and the necessary program logic. The program was written in Eclipse IDE by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of built-in class System and its method println to display the result. The program was executed for two-three sets of input and result obtained were verified to be correct and recorded.

Practice Program Statement:

It is required to store and analyse data about 6 car manufacturer's sales data in all the 12 months of a year. Demonstrate how you would store the data in a two-dimensional matrix and do the following

- a. Write a method to find for a given car manufacturer, the month in which, maximum no. of cars is sold.
- b. Write a method to find the average number of cars sold for each car manufacturer
- c. Write a method to find the total number of cars sold for each car manufacturer.

Assume – row index 0 – 'Maruti Suzuki', 1 – 'Hyundai' 2 – 'Tata Motors' 3- 'KIA' 4 – 'BMW' 5 – 'Renault' col index 0 – 'Jan', 1- 'Feb' 11 – 'Dec'

Demonstrate the working of the program with appropriate values for each car manufacturer and the months.

Source Code:

```
package tw1pract;

import

java.util.Scanner;public

class Tw1pract {

    public static void main(String[] args) {

        Scanner in=new

        Scanner(System.in);int

        carSale[][]=new int[6][12];

        int n;

        //reading the sales

        valuefor(int i=0;i<6;i++)

        {

            System.out.println("Enter the Sale per month for manufacturer "+(i+1)+":

            ");for(int j=0;j<12;j++) {

                carSale[i][j]=in.nextInt();

            }

        }

        System.out.println("Enter the manufacturer no. to find the maximum car sold:");

        n=in.nextInt();
```

```

        max(carSale,n);        //calling max method

        System.out.println("\nThe average cars sold by

        ");average(carSale);    //calling average

        method System.out.println("\nThe total cars

        sold by "); total(carSale);    //calling total

        method

    }

//method to find the month in which max cars are sold

public static void max(int[][] carSale,int n) {

    int count=carSale[n-1][0],month=1;

    for(int j=0;j<12;j++) {

        if(carSale[n-1][j]>count){

            count=carSale[n-1][j];

            month=j+1;

        }

    }

    switch(month) {

        case 1:System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in

        January.");

            break;

        case 2:System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in

        February.");

            break;

        case 3:System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in

        March.");

            break;

        case 4:System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in

        April.");

```



```
        break;

        case 5: System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in
May.");

        break;

        case 6: System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in
June.");

        break;

        case 7: System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in
July.");

        break;

        case 8: System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in
August.");

        break;

        case 9: System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in
September.");

        break;

        case 10: System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in
October.");

        break;

        case 11: System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in
November.");

        break;

        case 12: System.out.println("\nThe maximum number of cars sold by manufacturer "+n+" is in
December.");

        break;

        default: break;

    }

}

//method to find average cars sold
```

```

public static void average(int[][] carSale) {

    float avg;

    int sum=0;

    for(int i=0;i<6;i++) {

        for(int j=0;j<12;j++) {

            sum+=carSale[i][j];

        }

        avg=sum/12f;

        System.out.println("Manufacturer "+(i+1)+" :"+avg);

        sum=0;

    }

}

//method to find total cars sold

public static void total(int[][] carSale)

    {int sum=0;

    for(int i=0;i<6;i++) {

        for(int j=0;j<12;j++) {

            sum+=carSale[i][j];

        }

        System.out.println("Manufacturer "+(i+1)+" :"+sum);

        sum=0;

    }

}

}

```

Output:

```
<terminated> Tw1pract [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (
```

```
Enter the Sale per month for manufacturer 1:
```

```
35 12 19 29 55 43 68 11 29 68 16 12
```

```
Enter the Sale per month for manufacturer 2:
```

```
88 62 12 36 48 48 50 12 31 33 56 45
```

```
Enter the Sale per month for manufacturer 3:
```

```
19 76 55 19 22 55 11 29 15 87 90 12
```

```
Enter the Sale per month for manufacturer 4:
```

```
76 55 12 19 78 57 12 36 45 55 15 25
```

```
Enter the Sale per month for manufacturer 5:
```

```
21 12 45 64 50 45 80 35 25 51 25 48
```

```
Enter the Sale per month for manufacturer 6:
```

```
65 72 36 18 29 45 36 18 17 12 50 58
```

```
Enter the manufacturer no. to find the maximum car sold:
```

```
3
```

```
|
```

```
The maximum number of cars sold by manufacturer 3 is in November.
```

```
The average cars sold by
```

```
Manufacturer 1 :33.083332
```

```
Manufacturer 2 :43.416668
```

```
Manufacturer 3 :40.833332
```

```
Manufacturer 4 :40.416668
```

```
Manufacturer 5 :41.75
```

```
Manufacturer 6 :38.0
```

```
The total cars sold by
```

```
Manufacturer 1 :397
```

```
Manufacturer 2 :521
```

```
Manufacturer 3 :490
```

```
Manufacturer 4 :485
```

```
Manufacturer 5 :501
```

```
Manufacturer 6 :456
```

Title of the Experiment: Implementation of class and its member method**Experiment No.2****Date:** 02/10/2020**Problem Statement:**

Design a class called Rectangle having two methods. First method named as setDim() takes length and breadth of rectangle as parameters and the second method named as getArea() returns the area of the rectangle. Length and breadth of rectangle are entered through keyboard.

Objectives of the Experiment:

1. Learn declaration and initialization of class and methods in Java
2. Learn how to declare objects of a class and access methods
3. Understand the use of class and methods in a real-life application
4. Learn how to pass parameters to a method
5. Learn to Display the result in a readable/proper format

Program Source Code:

```
/**
Design a class called Rectangle having two methods. First method named as setDim() takes length and br
eadth of rectangle
as parameters and the second method named as getArea() returns the area of the rectangle. Length an
d
breadth of rectangle are entered through keyboard.
*/
import java.io.*;
import java.lang.*;
import java.util.*;

///Class to initialize methods
class Rectangle
{
    //data members
    int length,breadth;
    float area;
```

```

//member functions
void setDim(int length,int breadth){
    this.length = length;
    this.breadth = breadth;
}

float getArea(){
    return length*breadth;
}
}

//main class
class RectArea
{
    public static void main(String[]args){
        Scanner venki = new Scanner(System.in);
        Rectangle r = new Rectangle();    //object instantiation
        System.out.println("Enter the length of the rectanlge : ");
        int l = venki.nextInt();
        System.out.println("Enter the breadth of the rectanlge : ");
        int b = venki.nextInt();
        r.setDim(l,b);
        float a = r.getArea();
        System.out.println("Area is : "+a);
    }
}

```

CASE 1:

Input:

```

Enter the length of the rectanlge :
10
Enter the breadth of the rectanlge :
20

```

Output:

```
The area of the rectangle is : 200.0
```

CASE 2:

INPUT:

```
Enter the length of the rectanlge :  
23  
Enter the breadth of the rectanlge :  
89
```

OUTPUT:

```
The area of the rectangle is : 2047.0
```

Outcomes of the Experiment: At the end of the laboratory sessions the students should be able to

1. Demonstrate the use of class and methods in solving real-life problems.
2. Identify appropriate method to be used for a particular scenario
3. Learn how to return values from a method
4. Initialize variables of a class using this keyword
5. Identify how to make a call for the method

Conclusions: From the given problem statement, we could identify the necessary variables and use the appropriate class and methods and the necessary program logic. We understood how to calculate the area of rectangle using methods. The program was written in Visual Studio Code by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of the built-in class System and its method println to display the result. The program was executed for two-three sets of input and results obtained were verified to be correct and recorded.

PRACTICE PROBLEM

Design a class called Circle having the following methods:

getRadius() to read the radius

computeArea() to compute the area of the circle

computePerimeter() to compute the perimeter of the circle

Declare the required instance variables appropriately.

Program Source Code:

```
/**
Design a class called Circle having the following methods:
getRadius() to read the radius
computeArea() to compute the area of the circle
computePerimeter() to compute the perimeter of the circle
Declare the required instance variables appropriately. */

import java.io.*;
import java.util.*;
import java.lang.*;

class Circle{
    float radius,area,perimeter;
    static float pi = (float)22/7;
    void getRadius(float r){
        this.radius = r;
    }
    float computeArea(){
        return ((pi)*radius*radius);
    }
    float computePerimeter(){
        return (2*(pi)*radius);
    }
}
```

```

}

class CircleArea{
    public static void main(String[]args){
        Scanner venki = new Scanner(System.in);
        Circle c= new Circle();
        System.out.println("\nEnter the radius of circle : ");
        float r = venki.nextFloat();
        c.getRadius(r);
        System.out.println("The Area of circle is : "+c.computeArea());
        System.out.println("The Perimeter of circle is : "+c.computePerimeter
    ());
    }
}

```

INPUT:

```

Enter the radius of circle :
10

```

OUTPUT:

```

The Area of circle is : 314.2857
The Perimeter of circle is : 62.85714

```

CASE 2:

INPUT:

```

Enter the radius of circle :
15.5

```

OUTPUT:

```

The Area of circle is : 755.0714
The Perimeter of circle is : 97.428566

```


Title of the Experiment: Implementation of parameterized methods and constructors**Experiment No.3****Date:** 24/10/2020**Problem Statement:**

A certain small bank intends to automate few of its banking operations for its customers.

Design a class by name myBankAccount to store the customer data having following details:

1.accountNumber 2. acctType 3. Name 4. Address 5. accountBalance

The class must have both default and parameterized constructors. Write appropriate method to compute interest accrued on accountBalance based on accountType and time in years. Assume 5% for S/B account 6.5% for RD account and 7.65 for FD account. Further, add two methods withdrawAmount/depositAmount with amount as input to withdraw and deposit respectively.

The withdrawAmount method must report insufficient balance if accountBalance falls below Rs. 1000.

TASK 1: Build the class with appropriate member variables, constructors and methods.

TASK 2: Instantiate three objects of above type and perform different operations on the same.

TASK 3: Write a function to display all the three customer details in a tabular form with appropriate column headings.

Objectives of the Experiment:

1. Learn declaration and definition of constructors and methods in Java
2. Learn how to declare objects of a class and access methods
3. Learn how constructors are executed and its different types
4. Understand the use of constructors and methods in a real-life application
5. Learn how to pass parameters to a method and constructors
6. Learn to Display the result in a readable/proper format

Program Source Code:

```
import java.util.*;
class myBankAccount{
    int accNo;
    String name,address,acctType;
    double balance;
    static int count=0;
```

```

myBankAccount(){
    Scanner in= new Scanner(System.in);
    accNo=++count;
    System.out.println("Enter the type of account(SB | FD | RD): ");
    accType=in.nextLine();
    System.out.println("Enter customer name: ");
    name=in.nextLine();
    System.out.println("Enter customer address: ");
    address=in.nextLine();
    System.out.println("Enter Initial Amount: ");
    balance=in.nextDouble();
}

myBankAccount(String type, String name, String address, double balance){
    accNo=++count;
    accType=type;
    this.name=name;
    this.address=address;
    this.balance=balance;
}

void computeInterest(){
    double interest=0;
    switch(accType){
        case "SB" :{
            interest=balance*0.05;
            break;
        }
        case "RD" :{
            interest=balance*0.065;
            break;
        }
        case "FD" :{
            interest=balance*0.0765;
            break;
        }
    }
    System.out.println("Interest Earned: " + interest);
}

void withdrawAmount(double amt){
    if(accType.equals("SB")){
        if(balance-amt<1000)
            System.out.println("Denied...Insufficient Balance");
        else
            balance=balance-amt;
    }
}

```

```

        else
            System.out.println("Withdrawal not permitted on these types of account");
    }
    void depositAmount(double amt){
        balance=balance+amt;
    }
    void dispDetails(){
        System.out.println(String.format("%-10s|%-10d|%5s|%-6.2f",name,accNo,accType,balance));
    }
}

public class Bank{
    public static void main(String[] args){
        myBankAccount a1= new myBankAccount("SB","Venkatesh","Bhagyanagar Belagavi",348732.00);
        myBankAccount a2= new myBankAccount("FD","Rohit","Samrudhi ColonyBelagavi",254321.00);
        myBankAccount a3= new myBankAccount("RD","Ajay","Bhagyanagar Belagavi",2354234.00);

        System.out.println(String.format("%-10s|%-10s|%5s|%-6s","Name","AccNumber","Type","Balance"));
        a1.dispDetails();
        a2.dispDetails();
        a3.dispDetails();

        Scanner in= new Scanner(System.in);
        myBankAccount a4= new myBankAccount();
        a4.computeInterest();
        double amnt;
        System.out.println("Enter the amount to be deposited: ");
        amnt=in.nextDouble();
        a4.depositAmount(amnt);

        System.out.println("Enter the amount to be withdrawn: ");
        amnt=in.nextDouble();
        a4.withdrawAmount(amnt);

        System.out.println(String.format("%-10s|%-10s|%5s|%-6s","Name","AccNumber","Type","Balance"));
        a4.dispDetails();
    }
}

```

OUTPUT:

```
Name      |AccNumber | Type|Balance
Venkatesh |1          | SB|348732.00
Rohit     |2          | FD|254321.00
Ajay      |3          | RD|2354234.00
Enter the type of account(SB | FD | RD):
SB
Enter customer name:
Sanket
Enter customer address:
Mahantesh Nagar
Enter Initial Amount:
10000
Interest Earned: 500.0
Enter the amount to be deposited:
400
Enter the amount to be withdrawn:
100
Name      |AccNumber | Type|Balance
Sanket    |4          | SB|10300.00
PS D:\Programming\JAVA\Java\jdk\bin> █
```

```
Name      |AccNumber | Type|Balance
Venkatesh |1          | SB|348732.00
Rohit     |2          | FD|254321.00
Ajay      |3          | RD|2354234.00
Enter the type of account(SB | FD | RD):
FD
Enter customer name:
Ayan
Enter customer address:
Shri Nagar
Enter Initial Amount:
200000
Interest Earned: 15300.0
Enter the amount to be deposited:
1000
Enter the amount to be withdrawn:
500
Withdrawal not permitted on these types of account
Name      |AccNumber | Type|Balance
Ayan      |4          | FD|201000.00
PS D:\Programming\JAVA\Java\jdk\bin> █
```

```
PS D:\Programming\JAVA\Java\jdk\bin> java app
Name      |AccNumber | Type|Balance
Venkatesh |1          | SB|348732.00
Rohit     |2          | FD|254321.00
Ajay      |3          | RD|2354234.00
Enter the type of account(SB | FD | RD):
RD
Enter customer name:
Sejal
Enter customer address:
Anjaneya Nagar
Enter Initial Amount:
900000
Interest Earned: 58500.0
Enter the amount to be deposited:
50000
Enter the amount to be withdrawn:
10
Withdrawal not permitted on these types of account
Name      |AccNumber | Type|Balance
Sejal     |4          | RD|950000.00
█
```

Outcomes of the Experiment: At the end of the laboratory sessions the students should be able to

1. Demonstrate the use of constructors and methods in solving real-life problems.
2. Identify when to use methods and when to use constructors
3. Identify appropriate method or constructor to be used for a particular scenario
4. Learn how to return values from a method
5. Differentiate between default and parameterized constructors
6. Identify how to make a call for the method

Conclusions: From the given problem statement, we could identify the necessary variables and use the appropriate constructors and methods and the necessary program logic. We understood how the banking system works and the way in which transactions are handled. The program was written in Visual Studio Code by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of the built-in class System and its method println to display the result. The program was executed for two-three sets of input and results obtained were verified to be correct and recorded.

PRACTICE PROBLEM

Define a class to represent a rectangle in which constructors and parameterized methods are used. It also has a method to compute area of rectangle.

- i. First make a class rectangle in which we declare the parameterized constructor.
- ii. Then demonstrate the use of parameterized method.
- iii. Use a method to compute area of rectangle.
- iv. Create a class to demonstrate the call of the methods in previously created class rectangle holding constructors, parameterized methods and method to compute area of rectangle.

Program Source Code:

```
import java.util.Scanner;

class Rectangle{
    int length,breadth;
    Rectangle(int l,int b){
        length=l;
        breadth=b;
    }
    Rectangle(){
        //initialize an object
    }
    void changeDim(int length,int breadth) {
        this.length=length;
        this.breadth=breadth;
    }
    int area() {
        return length*breadth;
    }
}

public class tw3 {
```

```

public static void main(String[] args) {
    Rectangle r1=new Rectangle(20,40);
    System.out.println("Area of rectangle for object r1 :"+r1.area());
    Scanner s=new Scanner(System.in);
    Rectangle r2=new Rectangle();
    System.out.println("Enter the length and breadth for rectangle: ");
    int length=s.nextInt();
    int breadth=s.nextInt();
    r2.changeDim(length, breadth);
    System.out.println("Area of rectangle for object r2 :"+r2.area());
}
}

```

OUTPUT:

```

PS D:\Programming\JAVA\Java\jdk\bin> java app
Area of rectangle for object r1 :800
Enter the length and breadth for rectangle:
90
50
Area of rectangle for object r2 :4500

```

```

Area of rectangle for object r1 :800
Enter the length and breadth for rectangle:
35 45
Area of rectangle for object r2 :1575
PS D:\Programming\JAVA\Java\jdk\bin>

```

Title of the Experiment: Implementation of class and its member methods**Experiment No.4****Date:** 25/10/2020**Problem Statement:**

A company has two types of employees – FullTime and Partime. The company records for each employee his/her name, age, address, salary and gender. Given the basic salary of the FullTime employee the components of his/her gross salary are: Dearness allowance – 75% of basic salary, HRA – 7.5% of basic salary, IT – 10% of basic. The salary of a Partime employee is dependent on the qualification, experience, number of working hours and the rate per hour, as below:

	Qualification		
Experience		Per hour	Per day
0 years	0 Rs.	0 Rs.	0 Rs.
0 years	0 Rs.	0 Rs.	00 Rs.
0 years	0 Rs.	00 Rs.	00 Rs.

Model this as a problem of hierarchical inheritance by:

- 1) Identifying the super class with its data members and member functions.
- 2) Identify the sub-class/sub-classes and their associated data members and member functions.

Test the program by creating objects of the classes that are so identified.

Objectives of the Experiment:

1. Learn how inheritance works in Java
2. Learn how to create base and derived classes
3. Understand the use of extends keyword
4. Understand the use of inheritance in a real-life application
5. Learn how to base class constructors in derived class
6. Learn about abstract classes and method overriding
7. Learn to Display the result in a readable/proper format

Program Source Code:

```
abstract class Employee
{
    String name;
    int age;
    String address;
    char gender;
    double salary;
    Employee(String name,int age,String address, char gender)
    {
        this.name = name;
        this.age = age;
        this.address = address;
        this.gender = gender;
    }
    void showDetails()
    {
        System.out.println("Name : "+name);
        System.out.println("Age : "+age);
        System.out.println("Address : "+address);
        System.out.println("Gender : "+gender);
        System.out.println("Salary : "+salary);
    }
    abstract void computeSalary();
}
class FTEmployee extends Employee
{
    double basic;
    FTEmployee(String name,int age,String address, char gender,double basic)
    {
        super(name,age,address,gender);
        this.basic = basic;
    }
    @Override
    void computeSalary()
    {
        double da,hra,tax;
        da = basic * 0.75;
        hra = basic * 0.075;
        tax = basic * 0.1;
        salary = basic + da + hra - tax;
    }
}
```



```

class PTEmployee extends Employee
{
    String qual;
    int exp,hrsWorked;
    PTEmployee(String name,int age,String address, char gender,String qual,int exp,int hrsWork
ed)
    {
        super(name,age,address,gender);
        this.qual = qual;
        this.exp = exp;
        this.hrsWorked = hrsWorked;
    }
    @Override
    void computeSalary()
    {
        switch(qual)
        {
            case "BE":
                if(exp>=1 && exp<=5)
                    salary = hrsWorked * 300;
                else if(exp<=10)
                    salary = hrsWorked * 400;
                else
                    salary = hrsWorked * 500;
                break;
            case "Mtech":
                if(exp>=1 && exp<=5)
                    salary = hrsWorked * 500;
                else if(exp<=10)
                    salary = hrsWorked * 700;
                else
                    salary = hrsWorked * 1000;
                break;
            case "phD":
                if(exp>=1 && exp<=5)
                    salary = hrsWorked * 800;
                else if(exp<=10)
                    salary = hrsWorked * 1200;
                else
                    salary = hrsWorked * 1500;
                break;
        }
    }
}

```

```

public class app {
    public static void main(String[] args) {
        FTEmployee f1 = new FTEmployee("Rohit M Borse",007,"Bhagya Nagar, Belagavi",'M',59003)
        ;
        PTEmployee p1 = new PTEmployee("Sanket M Mungarwadi",23,"Mahantesh Nagar",'M',"BE",12,
1000);
        f1.computeSalary();
        f1.showDetails();
        p1.computeSalary();
        p1.showDetails();
    }
}

```

OUTPUT:

```

Name : Rohit M Borse
Age :7
Address :Bhagya Nagar, Belagavi
Gender : M
Salary :101780.175
Name : Sanket M Mungarwadi
Age :23
Address :Mahantesh Nagar
Gender : M
Salary :500000.0

Name : Venkatesh G Dhongadi
Age :23
Address :Bhagya Nagar, Belagavi
Gender : M
Salary :101780.175
Name : Ajay P Betagiri
Age :46
Address :Tilak Nagar
Gender : M
Salary :40000.0
PS D:\Programming\JAVA\Java\jdk\bin>

```

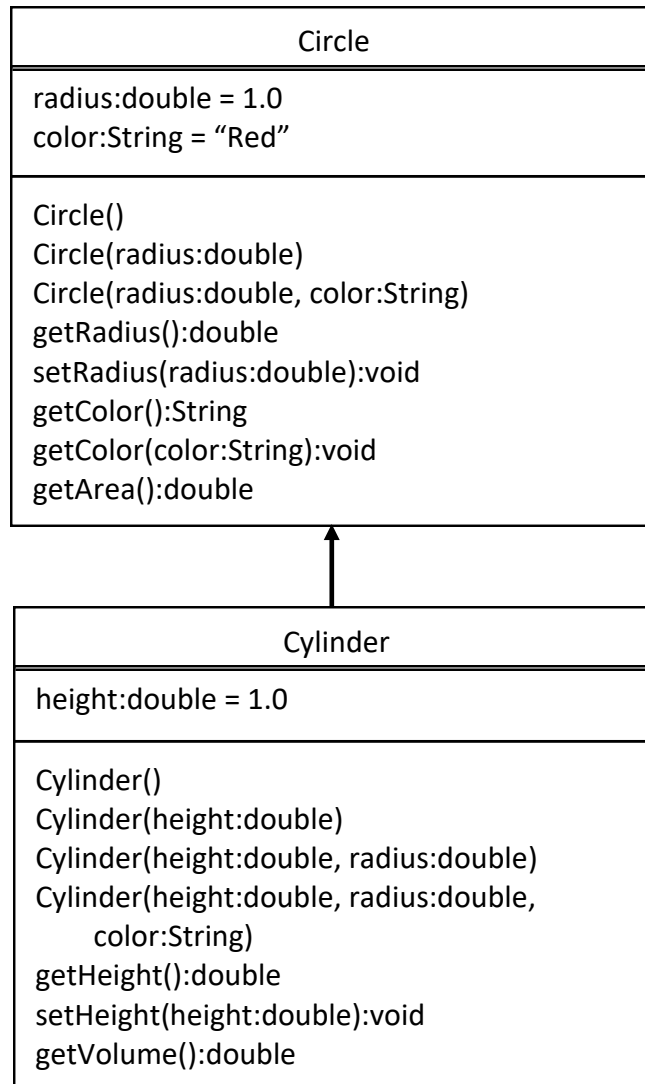
Outcomes of the Experiment: At the end of the laboratory sessions the students should be able to

1. Demonstrate the use inheritance in solving real-life problems.
2. Identify appropriate type of inheritance to use for a particular scenario
3. Creating derived classes using extends keyword
4. Learn how to call super class constructors
5. Identify how to implement inheritance in real life applications

Conclusions: From the given problem statement, we could identify the necessary methods and use the appropriate type of inheritance and the necessary program logic. We understood how to calculate the salary of different types of employees based on their qualification. The program was written in Visual Studio Code by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of the built-in class System and its method println to display the result. The program was executed for two-three sets of input and results obtained were verified to be correct and recorded.

PRACTICE PROBLEM

The class Cylinder inherits all the instance variables (radius and color) and methods (getRadius(), getArea(), among others) from its superclass Circle. It further defines a variable called height, three methods getHeight(), setHeight() and getVolume() and its own constructors. Implement the hierarchy as shown below:



Program Source Code:

```
import java.lang.*;
import java.io.*;
import java.util.*;
class Circle{
    double radius;
    String color;

    Circle(){
        radius = 1.0;
        color="Aqua";
    }

    Circle(double radius){
        this.radius=radius;
        color="Grey";
    }

    Circle(double radius, String color){
        this.radius=radius;
        this.color=color;
    }

    double getRadius() {
        return radius;
    }

    void setRadius(double radius) {
        this.radius=radius;
    }

    String getColor() {
        return color;
    }

    void setColor(String color) {
        this.color=color;
    }

    double getArea() {
        return (Math.PI*radius*radius);
    }
}

//subclass
class Cylinder extends Circle{
    double height;

    Cylinder(){
        super();
        height=1.0;
    }
}
```

```

    }
    Cylinder(double height){
        super();
        this.height=height;
    }
    Cylinder(double height, double radius){
        super(radius);
        this.height=height;
    }
    Cylinder(double height, double radius, String color){
        super(radius,color);
        this.height=height;
    }
    double getHeight() {
        return height;
    }
    void setHeight(double height) {
        this.height=height;
    }
    double getVolume() {
        return (Math.PI*radius*radius*height);
    }
}

public class app {

    public static void main(String[] args) {

        Circle c=new Circle(3.0,"Aqua");
        System.out.println("Radius of circle = "+c.getRadius()+"\nColor of Circle
= "+c.getColor());
        c.setColor("Blue");
        System.out.println("Changed Color of the circlce : "+c.getColor()+(String.
format("\nThe Area of the circle : %.2f",c.getArea())));

        Cylinder c1=new Cylinder(3.0,4.0,"Black");
        System.out.println("Radius of the Cylinder : "+c1.getRadius()+"\nHeight o
f the Cylinder : "+c1.getHeight()+"\nColor of the Cylinder : "+c1.getColor()+Stri
ng.format("\nVolume of the Cylinder : %.2f",c1.getVolume()));
    }
}

```

OUTPUT:

```
PS D:\Programming\JAVA\Java\jdk\bin> java app
Radius of circle = 3.0
Color of Circle = Violet
Changed Color of the circle : Blue
The Area of the circle : 28.27
Radius of the Cylinder : 4.0
Height of the Cylinder : 3.0
Color of the Cylinder : Green
Volume of the Cylinder : 150.80
PS D:\Programming\JAVA\Java\jdk\bin> |
```

```
PS D:\Programming\JAVA\Java\jdk\bin> java app
Radius of circle = 3.0
Color of Circle = Aqua
Changed Color of the circle : Blue
The Area of the circle : 28.27
Radius of the Cylinder : 4.0
Height of the Cylinder : 3.0
Color of the Cylinder : Black
Volume of the Cylinder : 150.80
PS D:\Programming\JAVA\Java\jdk\bin> |
```

USN : 2GI19CS175

Student Name : Venkatesh G Dhongadi

Title of the Experiment(implementation of method overloading.)

Experiment No. 5

Date : _24/12/2020_____

Problem Statement :

5.1) Create a Stack class having an integer array say elem and top_of_stack as instance variables. Define three overloaded methods having the following signatures:

- a. initStack(int size) to create an array of specified size and initialize the top_of_stack
- b. initStack(Stack another) to initialize the Stack object with state of the Stack object "another"
- c. initStack(int [] a) to initialize contents of a[] to the instance variable elem.

Write following methods:

- a. push(): Pushes the element onto the stack,
- b. pop(): Returns the element on the top of the stack, removing it in the process, and
- c. peek(): Returns the element on the top of the stack, but does not remove it.

Also write methods that check whether stack is full and stack is empty and return boolean value true or false appropriately.

Objectives of the Experiment :

- 1. Learn declaration and initialization of variables
- 2. Learn necessary of method overloading
- 3. Learn advantages of method overloading
- 4. Understand the overloaded methods in a real-life application
- 5. Learn the usage of Looping constructs and control statements
- 6. Learn to Display the result in a readable/proper format

Program Source code:

```
package termWork5a;
```

```

class MyStack{
    int []elem;
    int top;
    void initStack(int size) {
        elem=new int[size];
        top=-1;
    }
    void initStack(MyStack another) {
        elem=new int[another.elem.length];
        top=-1;
        for(int e:another.elem) {
            push(e);
        }
    }
    void initStack(int []a) {
        elem=new int[a.length];
        top=-1;
        for(int e:a) {
            push(e);
        }
    }
    void push(int e) {
        if(isfull()) {
            System.out.println("stack overflow");
        }
        else
        {
            elem[++top]=e;
            System.out.println("pushed into the stack:"+e);
        }
    }
    void pop() {
        if(isEmpty())
            System.out.println("Stack underFlow ");

        else
            System.out.println("element is popped is :"+ elem[top--]);

    }
    void peek()
    { if(isEmpty())
        System.out.println("nothing is on the top ");
    else
        System.out.println("element on top is:"+elem[top]);
    }
}

```



```

    }
    boolean isfull()
    {
        return top==elem.length-1 ? true:false;

    }
    boolean isEmpty() {
        return top== -1 ?true:false;

    }
}
public class MyClass {

    public static void main(String[] args) {
        MyStack s1= new MyStack();
        s1.initStack(5);
s1.push(10);
s1.push(20);
s1.push(30);
s1.push(40);
s1.push(50);
s1.push(60);

MyStack s2=new MyStack();
s2.initStack(s1);
s2.pop();
s2.pop();
s2.pop();
s2.pop();
s2.pop();
s2.pop();
s2.pop();
s2.peek();
int[]a = {1,2,3,4,5,6};
MyStack s3=new MyStack();
s3.initStack(a);
s3.peek();
s3.pop();
s3.pop();
s3.pop();

    }

```

}

Input & Output:

Case1:

```
<terminated> MyClass (17) [Java Application] C:\P
pushed into the stack:10
pushed into the stack:20
pushed into the stack:30
pushed into the stack:40
pushed into the stack:50
stack overflow
pushed into the stack:10
pushed into the stack:20
pushed into the stack:30
pushed into the stack:40
pushed into the stack:50
element is popped is :50
element is popped is :40
element is popped is :30
element is popped is :20
element is popped is :10
Stack underFlow
nothing is on the top
pushed into the stack:1
pushed into the stack:2
pushed into the stack:3
pushed into the stack:4
pushed into the stack:5
pushed into the stack:6
element on top is:6
element is popped is :6
element is popped is :5
element is popped is :4
```

Case 2:

```
<terminated> MyClass (17) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.e
pushed into the stack:100
pushed into the stack:20
pushed into the stack:30
pushed into the stack:40
pushed into the stack:50
stack overflow
element is popped is :50
element is popped is :40
element on top is:30
element is popped is :30
pushed into the stack:100
pushed into the stack:20
pushed into the stack:30
pushed into the stack:40
pushed into the stack:50
element is popped is :50
element is popped is :40
element is popped is :30
element on top is:20
element is popped is :20
element is popped is :100
Stack underFlow
nothing is on the top
pushed into the stack:1
pushed into the stack:2
pushed into the stack:3
pushed into the stack:4
pushed into the stack:5
pushed into the stack:6
element on top is:6
element is popped is :6
pushed into the stack:60
element is popped is :5
element is popped is :4
```

Outcomes of the Experiment :

At the end of the laboratory sessions the students should be able to

1. Demonstrate the use of Classes and their member methods to solve real life problems.
2. Identify appropriate variables and their types
3. Learn how to create class, methods and object of class
4. Learn how to pass parameter to the class
5. Learn method overloading
6. Learn how data validation is important

Conclusions :

From the given problem statement, we could identify the necessary variables of appropriate type, and conditional statements and the necessary program logic. The program was written in Eclipse IDE by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of built-in class System and its method println to display the result. The program was executed for two-three sets of input and result obtained were verified to be correct and recorded

2)(practice) Implement a linear search function by using method overloading concept for an array of integers, double and character elements

Program Source code:

```
package termWork5;
import java.util.Scanner;

class MyLinearSearch{

    int linearSearch(int[] c,int key2) {
        for(int i=0;i<c.length;i++) {
            if(c[i]==key2)
            {
                return i;
            }

        }

        return -1;

    }

}
```

```

double linearSearch(double []a,double key) {

    for(int i=0;i<a.length;i++)
        if(a[i]==key)
            return i;

    return -1;

}

int linearSearch(char []a,char key) {

    for(int i=0;i<a.length;i++)
        if(a[i]==key)
            return i;

    return -1;

}

}

public class MyClass {

    public static void main(String[] args) {

        MyLinearSearch m=new MyLinearSearch();
        Scanner in=new Scanner(System.in);
        System.out.println("what you want to search");
        System.out.println("1:integer \t 2:double \t 3:char");
        int choice=in.nextInt();
        System.out.println("how many elements you want to enter");
        int n=in.nextInt();
        System.out.println("enter "+ n+" elements ");
        switch(choice) {

            case 1: int[]a=new int[n];
                    for(int i=0;i<n;i++)
                        a[i]=in.nextInt();
                    System.out.println("entered elements to be searched");
                    int key=in.nextInt();
                    int position=m.linearSearch(a, key);
                    check(position);

```

```

        break;
        case 2:
            double[] a1=new double[n];
            for(int i=0;i<n;i++)
                a1[i]=in.nextDouble();
            System.out.println("entered elements to be searched");
            double key1=in.nextInt();
            int position1=(int)m.linearSearch(a1, key1);
            check(position1);
            break;
        case 3:
            char[] c=new char[n];
            for(int i=0;i<n;i++)
                c[i]=in.next().charAt(0);
            System.out.println("entered elements to be searched");
            char key2= in.next().charAt(0);
            int position2=(int)m.linearSearch(c, key2);
            check(position2);
            break;
    }

}

private static void check(int position1) {
    // TODO Auto-generated method stub
    if(position1>=0)
        System.out.println("element is found at "+(position1+1)+" position");
    else
        System.out.println("element is not found\n");
}

}

```

Input&Output:

Case1:

```
<terminated> MyClass (15) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (D
what you want to search
1:integer      2:double      3:char
1
how many elements you want to enter
7
enter 7 elements
12
43
23
12
45
23
23
entered elements to be searched
12
element is found at 1 position
```

Case2

```
<terminated> MyClass (15) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe
what you want to search
1:integer      2:double      3:char
2
how many elements you want to enter
10
enter 10 elements
122
3242
42124
423232
43523
12342
234532
1234532
124325
523612
entered elements to be searched
12342
element is found at 6 position
```

Case3:

```
<terminated> MyClass (15) [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\ja
what you want to search
1:integer      2:double      3:char
3
how many elements you want to enter
7
enter 7 elements
aditya
suraj
rohit
pooja
akash
anita
sachin
entered elements to be searched
pooja
element is found at 4 position
```

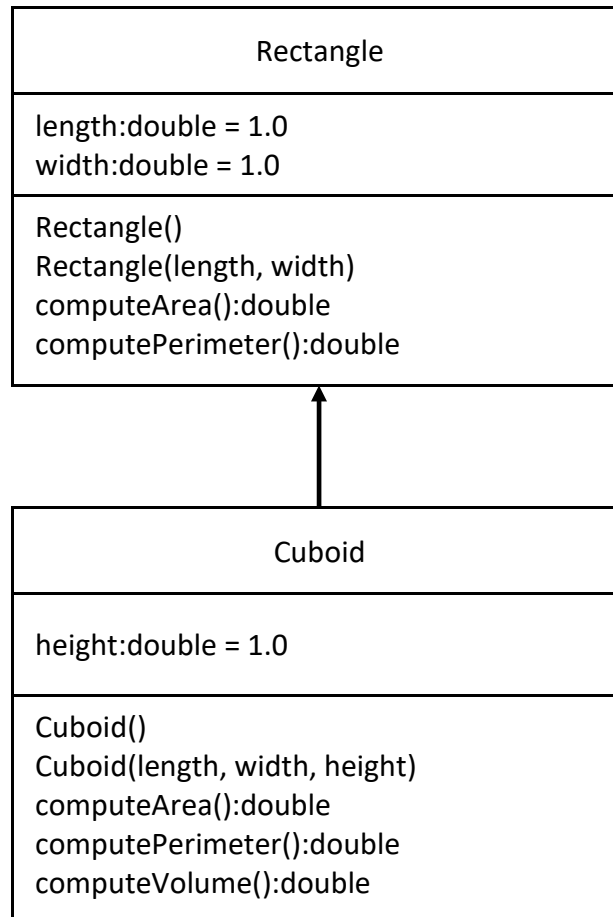

Case4

```
<terminated> MyClass (15) [Java Application] C:\Program Files\Java\jdk-14.0.2\
what you want to search
1:integer      2:double      3:char
3
how many elements you want to enter
5
enter 5 elements
aditya
pooja
anita
rohit
ajay
entered elements to be searched
vishal
element is not found
```

5 b) Overriding.

5 b.1)Implement the following class hierarchy. In the Cuboid class, override the method computeArea() and computePerimeter() of Rectangle class to compute the surface area and perimeter of a rectangle cuboid. Add a method computeVolume() in Cuboid class to compute volume of the cuboid. Assuming length, width and height as l, w and h respectively,

- formula to find the surface area = $2(lw) + 2(hl) + 2(hw)$
- formula to find the perimeter = $2l + 2w$
- formula to find the volume = $l \times w \times h$



Program source code:

```
import java.io.*;
import java.lang.*;
import java.util.*;

class Rectangle
{
```

```

protected double width,length;

public Rectangle(double width, double length){
    this.width = width;
    this.length=length;
}

public double computeArea()
{
    return(length*width);
}

public double computePeri()
{
    return(2*(length+width));
}
}

class Cuboid extends Rectangle
{
    double height;
    public Cuboid(double height,double width, double length)
    {
        super(width,length);
        this.height=height;
    }
    public double computeArea()
    {
        return(2*(length*width+height*length+height*width));
    }
    public double computePeri()
    {
        return(2*(length+height+width));
    }
    public double computeVolume()
    {
        return(length*height*width);
    }
}

public class MethodOverRiding
{
    public static void main(String[]args)
    {
        Rectangle r1=new Rectangle(5,6);
        System.out.println("Area of Rectangle : "+r1.computeArea());
        System.out.println("Perimeter of Rectangle : "+r1.computePeri());

        Cuboid c1= new Cuboid(1,2,3);
        System.out.println("Area of Cuboid : "+c1.computeArea());
        System.out.println("Perimeter of Cuboid : "+c1.computePeri());
        System.out.println("Volume of Cuboid : "+c1.computeVolume());
    }
}

```

```
}}
```

INPUT&OUTPUT:

CASE1:

```
<terminated> MethodOverRiding [Java Application] C:\Progra  
Area of Rectangle : 30.0  
Perimeter of Rectangle : 22.0  
Area of Cuboid : 22.0  
Perimeter of Cuboid : 12.0  
Volume of Cuboid : 6.0  
|
```

Title of the Experiment (Abstract Classes)**Experiment No.**____06_____**Date :** __24/12/20____**Problem Statement:**

Design an abstract class Car to have carName, chassiNum, modelName as member variables and add two abstract methods, startCar and operateSteering . Inherit MarutiCar and BmwCar from Car class and override the two abstract methods in their own unique way. Design a driver class to have driver name, gender and age as data members and add a method driveCar with abstract class reference variable as argument and invoke the two basic operations namely, startCar and operateSteering and demonstrate run-time polymorphism.

Objectives of the Experiment:

1. Learn declaration and initialization of variables and implementation of Run-time Polymorphism in Java.
2. Understand the use of Run-time Polymorphism in a real-life application.
3. Learn the usage of Looping constructs and control statements.
4. Learn to Display the result in a readable/proper format.

Problem Source Code:

```
package termwork_6;
```

```
public class termwork_6 {
```

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

```
        MarutiCar c=new MarutiCar(22,"Maruti","Maruti 800",10001023);
        BMWCar c1=new BMWCar(125,"BMW-AS 6","BMW",1233422);
        Driver d=new Driver("Ajay","Male",24);
        d.driveCar(c);
        d.driveCar(c1);
    }
```

```
}
```

```
abstract class Car {
```

```
    String carName, model;
```

```
    long chassNumber;
```

```
    public Car(String carName, String model, long chassNumber) {
```

```
        this.carName = carName;
```

```
        this.model = model;
```

```
        this.chassNumber = chassNumber;
```

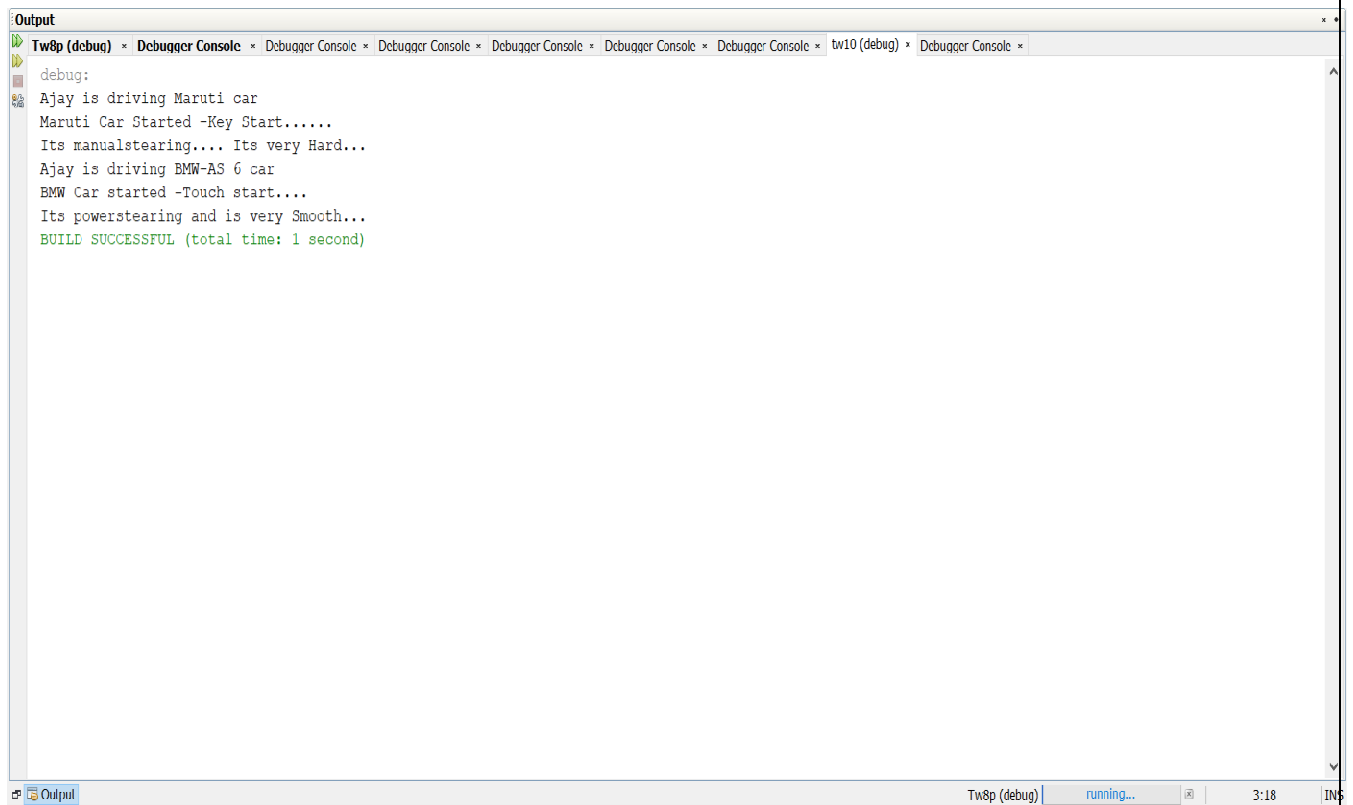
```
    }
```

```

        public abstract void startCar();
        public abstract void operateSteering();
    }
    class MarutiCar extends Car {
        float mileage;
        public MarutiCar(float mileage, String carName, String model, long chassNumber) {
            super(carName, model, chassNumber);
            this.mileage = mileage;
        }
        public void startCar(){ System.out.println("Maruti Car Started -Key Start. ...."); }
        public void operateSteering() { System.out.println("Its manualsteering.... Its very Hard..."); }
    }
    class BMWCar extends Car{
        float horsepower;
        public BMWCar(float horsepower, String carName, String model, long chassNumber) {
            super(carName, model, chassNumber);
            this.horsePower = horsepower;
        }
        public void startCar(){ System.out.println("BMW Car started -Touch start...."); }
        public void operateSteering(){ System.out.println("Its powersteering and is very Smooth. "); }
    }
}
class Driver {
    String name, gender;
    int age;
    public Driver(String name, String gender, int age) {
        this.name = name;
        this.gender = gender;
        this.age = age;
    }
    public void driveCar(Car c){
        System.out.println(name + " is driving "+c.carName+" car ");
        c.startCar();
        c.operateSteering();
    }
}

```

Output:



The screenshot shows the Eclipse IDE's Output window. The title bar indicates the active tab is 'Tw8p (debug)'. The window contains the following text:

```
debug:
Ajay is driving Maruti car
Maruti Car Started -Key Start.....
Its manualsteering.... Its very Hard...
Ajay is driving BMW-AS 6 car
BMW Car started -Touch start....
Its powersteering and is very Smooth...
BUILD SUCCESSFUL (total time: 1 second)
```

The status bar at the bottom shows 'Tw8p (debug)' is 'running...' at 3:18.

Outcomes of the Experiment:

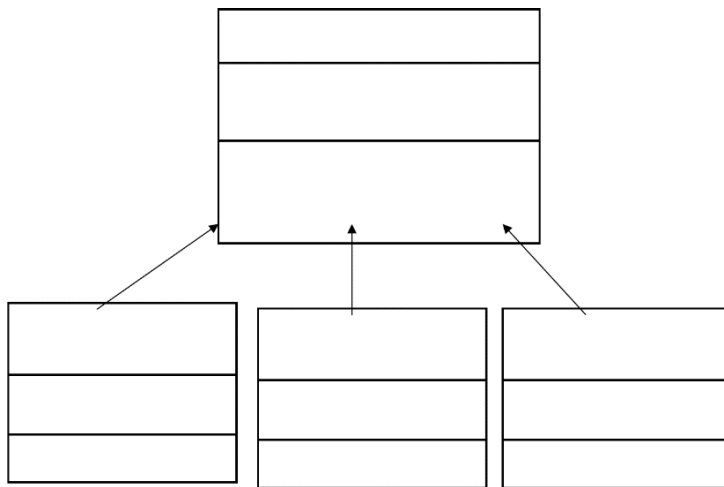
1. Able to Demonstrate the use of Run-time Polymorphism in solving real-life problems.
2. Identify appropriate variables and their types
3. Identify appropriate looping constructs (for)
4. Check if one loop will suffice or use nesting
5. Identify the control statements needed to meet the problem requirements.

Conclusion:

From the given problem statement, we could identify the necessary variables of appropriate type, and looping/control statements and the necessary program logic. The program was written in Eclipse IDE by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of built-in class System and its method println to display the result. The program was executed for two sets of input and result obtained were verified to be correct and recorded.

Practice Problem Statement:

Implement the following inheritance hierarchy.



Problem Source Code:

```
package termwork_6_pp1;
import java.util.Scanner;
```

```
abstractclass Shape{
    double area;
    double perimeter;
    String type;

    abstractvoid computeArea();
    abstractvoid computePerimeter();
}
```

```
class Rectangle extends Shape{
    double length, width;

    Rectangle(){
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the length and width: ");
        length = in.nextDouble();
        width = in.nextDouble();
    }

    void computeArea() {
        area = length * width;
        System.out.println("Area of the rectangle is: " + area);
    }

    void computePerimeter() {
```



```

        perimeter = 2 * (length + width);
        System.out.println("Perimeter of the rectangle is: " + perimeter);
    }
}

```

```

class Circle extends Shape{
    double radius;

    Circle(){
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the radius: ");
        radius = in.nextDouble();
    }

    void computeArea() {
        area = Math.PI * radius * radius;
        System.out.println("Area of the circle is: " + area);
    }

    void computePerimeter() {
        perimeter = 2 * Math.PI * radius;
        System.out.println("Perimeter of the circle is: " + perimeter);
    }
}

```

```

class Triangle extends Shape{
    double a, b, c, s;

    Triangle(){
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the 3 sides: ");
        a = in.nextDouble();
        b = in.nextDouble();
        c = in.nextDouble();
    }

    void computeArea() {
        s = (a + b + c) / 2.0;
        area = Math.sqrt(s * (s-a) * (s-b) * (s-c));
        System.out.println("Area of the triangle is: " + area);
    }

    void computePerimeter() {
        perimeter = a + b + c;
        System.out.println("Perimeter of the triangle is: " + perimeter);
    }
}

```

```

publicclass Tw6_pp1 {

    publicstaticvoid main(String[] args) {
        // TODO Auto-generated method stub
        Rectangle r = new Rectangle();
        r.computeArea();
        r.computePerimeter();

        Circle c = new Circle();
        c.computeArea();
        c.computePerimeter();

        Triangle t = new Triangle();
        t.computeArea();
        t.computePerimeter();
    }
}

```

Output:

Case 1:

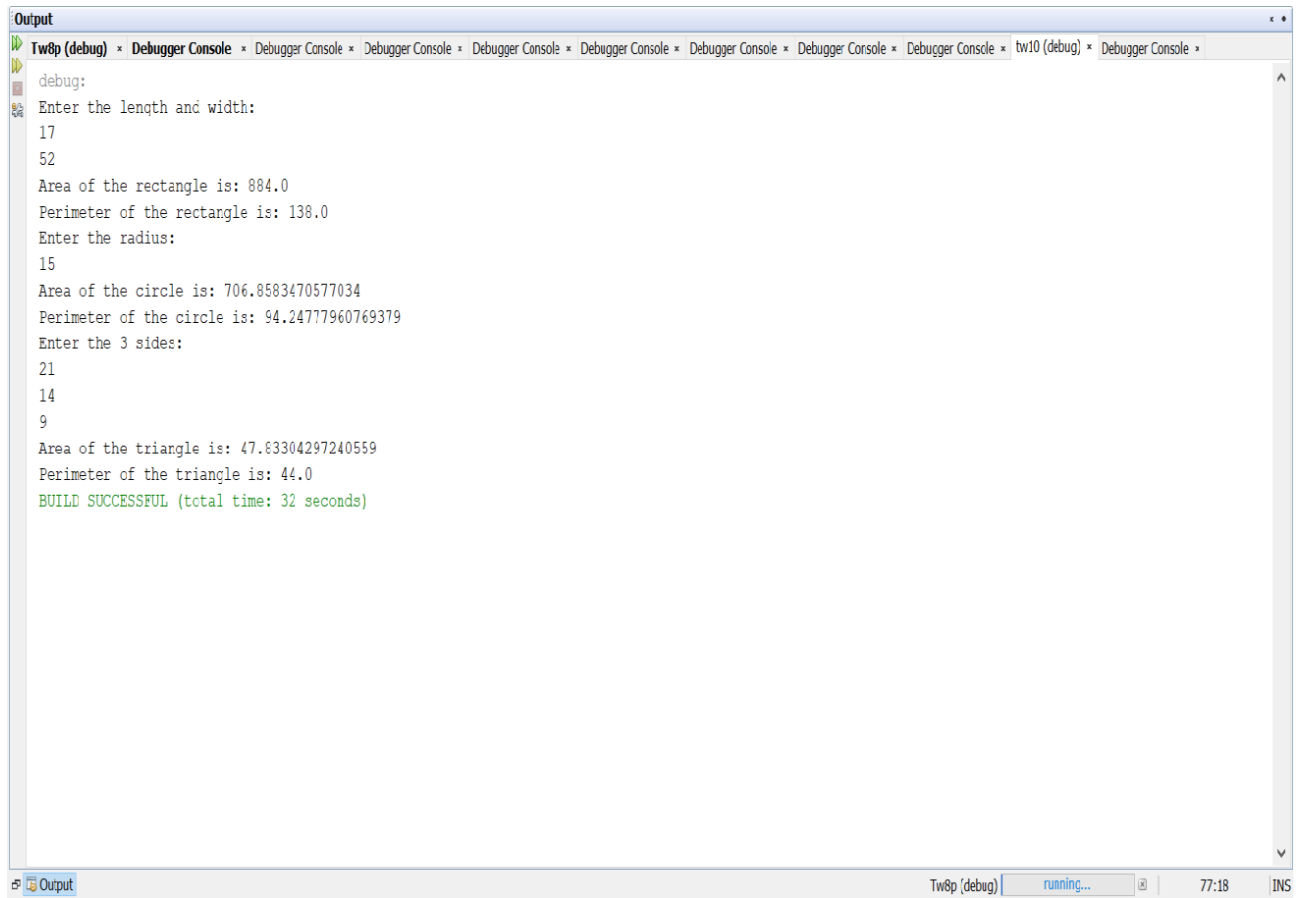
```

Output
Tw8p (debug) * Debugger Console * Debugger Console * Debugger Console * Debugger Console * Debugger Console * Debugger Console * tw10 (debug) * Debugger Console * Debugger Console *
debug:
Enter the length and width:
15
24
Area of the rectangle is: 360.0
Perimeter of the rectangle is: 78.0
Enter the radius:
7
Area of the circle is: 153.93804002569985
Perimeter of the circle is: 43.982297150257104
Enter the 3 sides:
10
26
32
Area of the triangle is: 114.2628548566856
Perimeter of the triangle is: 68.0
BUILD SUCCESSFUL (total time: 48 seconds)
|

```

Tw8p (debug) running... 77:13 INS

Case2:



The screenshot shows an IDE's Output window with the following content:

```
debug:
Enter the length and width:
17
52
Area of the rectangle is: 884.0
Perimeter of the rectangle is: 138.0
Enter the radius:
15
Area of the circle is: 706.8583470577034
Perimeter of the circle is: 94.24777960769379
Enter the 3 sides:
21
14
9
Area of the triangle is: 47.83304297240559
Perimeter of the triangle is: 44.0
BUILD SUCCESSFUL (total time: 32 seconds)
```

The window title bar indicates the project is 'Tw8p (debug)' and the status is 'running...'. The bottom status bar shows the time '77:18' and the user 'INS'.

Title of the Experiment (Interfaces)**Experiment No.**____07____**Date :**__24/12/20____**Problem Statement:**

Write a Java application to implement the following UML diagram.

- PrimeTester class implements isPrime() method by iterating from 2 to n-1 for a given number n
- ImprPrimeTester class implements isPrime() method by iterating from 2 to n/2
- FasterPrimeTester class implements isPrime() method by iterating from 2 to
- FastestPrimeTester class implements isPrime() method using Fermat's Little theorem.
 - o Fermat's Little Theorem:
 - o If n is a prime number, then for every a, $1 < a < n-1$, $a^{n-1} \% n = 1$

Objectives of the Experiment:

1. Learn declaration and initialization of variables and Interfaces in Java.
2. Understand the use of Interfaces in a real-life application.
3. Learn the usage of Looping constructs and control statements.
4. Learn to Display the result in a readable/proper format.

Problem Source Code:

```
package termwork_7;

public interface Prime {
    boolean isPrime(int n);
}

package termwork_7;

class PrimeTester implements Prime {
    public boolean isPrime(int n) {
        boolean flag = true;
        for (int i = 2; i < n; i++) {
            if (n % i == 0) {
                flag = false;
                break;
            }
        }
        return flag;
    }
}
```

```

class ImprPrimeTester implements IPrime {
    public boolean isPrime(int n) {
        boolean flag = true;
        for (int i = 2; i < n/2; i++) {
            if (n % i == 0) {
                flag = false;
                break;
            }
        }
        return flag;
    }
}

```

```

class FasterPrimeTester implements IPrime {
    public boolean isPrime(int n) {
        boolean flag = true;
        for (int i = 2; i < Math.sqrt(n); i++) {
            if (n % i == 0) {
                flag = false;
                break;
            }
        }
        return flag;
    }
}

```

```

class FastestPrimeTester implements IPrime {
    public boolean isPrime(int n) {
        int a = 2;
        if (Math.pow(a, n-1) % n == 1) {
            return true;
        }
        else {
            return false;
        }
    }
}

```

```

public class Prime {

    public static void main(String[] args) {
        PrimeTester p1 = new PrimeTester();
        ImprPrimeTester p2 = new ImprPrimeTester();
        FasterPrimeTester p3 = new FasterPrimeTester();
        FastestPrimeTester p4 = new FastestPrimeTester();
        System.out.println(p1.isPrime(12));
        System.out.println(p1.isPrime(13));
        System.out.println(p2.isPrime(12));
        System.out.println(p2.isPrime(13));
    }
}

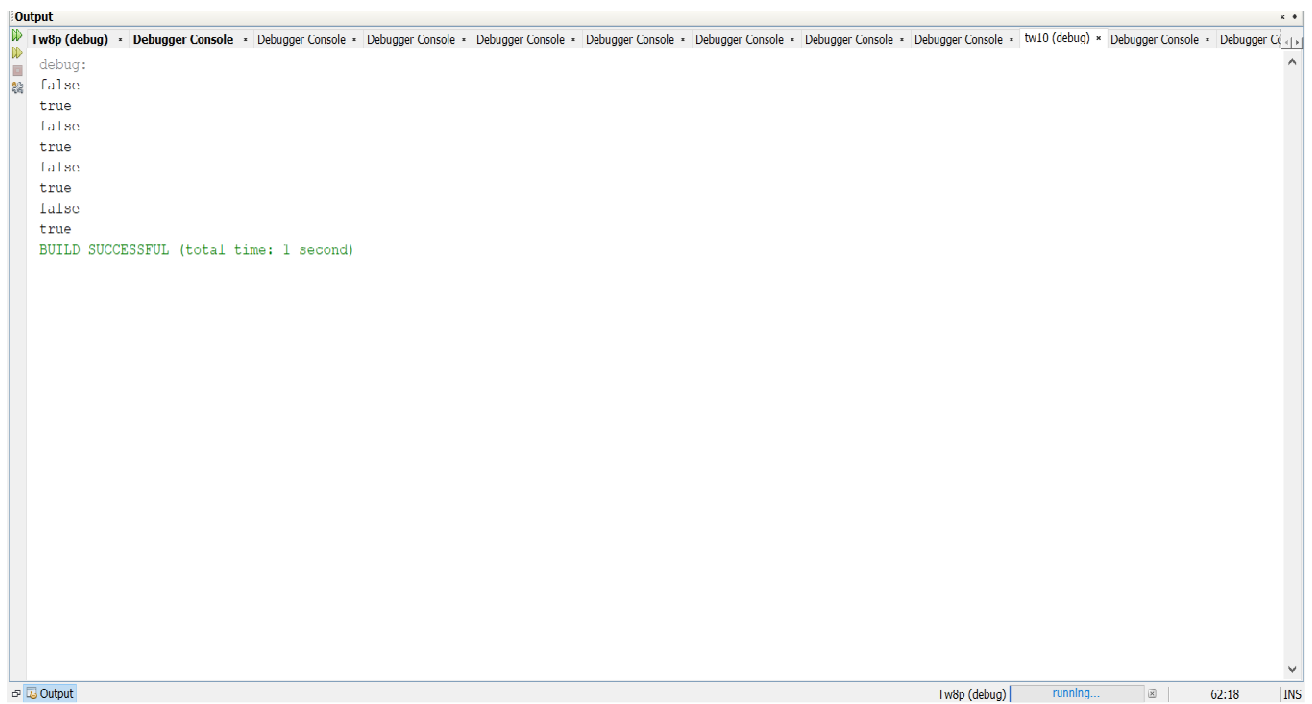
```

```

        System.out.println(p3.isPrime(12));
        System.out.println(p3.isPrime(13));
        System.out.println(p4.isPrime(12));
        System.out.println(p4.isPrime(13));
    }
}

```

Output:



Outcomes of the Experiment:

1. Able to Demonstrate the use of Interfaces in solving real-life problems.
2. Identify appropriate variables and their types
3. Identify appropriate looping constructs (for)
4. Check if one loop will suffice or use nesting
5. Identify the control statements needed to meet the problem requirements.

Conclusions:

From the given problem statement, we could identify the necessary variables of appropriate type, and looping/control statements and the necessary program logic. The program was written in Eclipse IDE by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of built-in class System and its method println to display the result. The program was executed for two sets of input and result obtained were verified to be correct and recorded.

Practice Problem Statement:

Write a JAVA program which has:

- i. An Interface class for Stack Operations (viz., push(), pop(), peek(), display())
- ii. A Class that implements the Stack Interface and creates a fixed length Stack.
- iii. A Class that implements the Stack Interface and creates a Dynamic Length Stack.
- iv. A Class that uses both the above Stacks through Interface reference and does the Stack operations that demonstrates the runtime binding.

Problem Source Code:

```
package termwork_7_pp1;
```

```
public interface Stack {  
    void push(int e);  
    void pop();  
    void peek();  
    void display();  
}
```

```
package termwork_7_pp1;  
import java.util.ArrayList;
```

```
class FStack implements Stack {  
    int [] element;  
    int top;  
  
    FStack(int size) {  
        element = new int [size];  
        top = -1;  
    }  
  
    public void push(int e) {  
        if (top == element.length - 1) {  
            System.out.println("Stack Overflow");  
        }  
        else {  
            element[++top] = e;  
        }  
    }  
  
    public void pop() {  
        if (top == -1) {  
            System.out.println("Stack Underflow");  
        }  
        else {  
            System.out.println(element[top--] + " is popped");  
        }  
    }  
}
```

```

    }

    public void peek() {
        if (top == -1) {
            System.out.println("No elements in the Stack");
        }
        else {
            System.out.println(element[top] + " is on top of the Stack");
        }
    }

    public void display() {
        System.out.println("The status of the Stack: ");
        for (int i = top; i >= 0; i--) {
            System.out.println(element[i]);
        }
    }
}

class DStack implements Stack {
    ArrayList<Integer> element;
    int top;

    DStack() {
        top = 0;
        element = new ArrayList();
    }

    public void push(int e) {
        element.add(e);
        top++;
    }

    public void pop() {
        if (element.size() == 0) {
            System.out.println("Stack Underflow");
        }
        else {
            System.out.println(element.remove(--top) + " is popped");
        }
    }

    public void peek() {
        if (element.size() == 0) {
            System.out.println("No elements in the Stack");
        }
        else {
            System.out.println(element.get(top-1) + " is on top of the Stack");
        }
    }
}

```



```

        public void display() {
            System.out.println("The status of the Stack: ");
            for(int i=element.size()-1; i>= 0; i--) {
                System.out.println(element.get(i));
            }
        }
    }

    public class tw7a {
        public static void main(String[] args) throws Exception{
            DStack ds = new DStack();
            ds.push(10);
            ds.push(20);
            ds.push(30);
            ds.push(40);
            ds.push(50);
            ds.display();
            ds.peek();
            ds.pop();
            ds.pop();
            ds.pop();
            ds.pop();
        }
    }
}

```

Output:

```

Output
Tw6p (debug) * Debugger Console * Debugger Console * Debugger Console * Debugger Console * Debugger Console * Debugger Console * Debugger Console * Debugger Console * tw10 (debug) * Debugger Console * Debugger Console
debug:
The status of the Stack:
50
40
30
20
10
50 is on top of the Stack
50 is popped
40 is popped
30 is popped
20 is popped
10 is popped
BUILD SUCCESSFUL (total time: 1 second)

```



Title of the Experiment (Exception Handling)**Experiment No.** __08__**Date :** __24/12/20__**Problem Statement :**

Assume that you have received a request from the transport authority for automating the task of issuing the permanent license for two wheelers. The mandatory condition to issue the license are: 1) the applicant must over 18 years of age and 2) holder of a valid learner's license and 3) no accident cases in the last one year.

Write a Java program that reads user details as required (use the Scanner class). Create user defined exceptions to check for the three conditions imposed by the transport authority. Based on the inputs entered by the user, decide and display whether or not a license has to be issued or an error message as defined by the user exception.

Objectives of the Experiment :

1. Learn the use of Exception Handling in Java
2. Understand the use of Exception Handling in a real-life application
3. To develop user defined exceptions.
4. Learn to Display the result in a readable/proper format

Program Source Code :

```
package tw8;

import java.util.Scanner;

class License{// class named License is created

    String gender,name;

    Intno_of_cases,age;

    char valid_LL;

    // necessary variables are declared

    License() // constructor is defined

    {

        Scanner s = new Scanner(System.in);

        System.out.println("Name of the Applicant    :");

        name = s.nextLine();
```

```

System.out.println("Age          :");

    age = Integer.parseInt(s.nextLine());

System.out.println("Gender (M/F)      :");

    gender = s.nextLine();

System.out.println("Is valid LL issued (Y/N)?   :");

Valid_LL = s.nextLine().charAt(0);

System.out.println("Num of accidents in past year :");

no_of_cases =Integer.parseInt(s.nextLine());

System.out.println();

// the variables will be initialised by the users input
}

void validateData() // method to validate the data provided by the user
{
try // try block to check for user defined exceptions
{
    if(age<18)

        throw new UnderAgeException("Invalid Age");

    if(valid_LL!='Y')

        throw new InvalidLLException("Invalid LLR");

    if(no_of_cases>0)

        throw new AccidentException("Involved in Accidents ");

System.out.println("The License can be issued ");

}

catch(UnderAgeException e) /* catch block to provide the message related to under age exception*/

{

System.out.println(e.getMessage());

System.out.println(e);

}
}

```

```
catch(InvalidLLEException e) /* catch block to provide the message related to Invalidlearning license exception*/
```

```
{  
System.out.println(e.getMessage());  
System.out.println(e);  
}
```

```
catch(AccidentException e) // catch block to provide the message related to accident Exception
```

```
{  
System.out.println(e.getMessage());  
System.out.println(e);  
}  
}  
}
```

```
class UnderAgeException extends Exception /* subclass UnderAgeException is extending the superclass Exception*/
```

```
{  
UnderAgeException(String msg)  
{  
    super(msg);  
}  
public String toString(){ // overriding the toString method.  
    return "The person is underage";  
}  
}
```

```
class InvalidLLEException extends Exception/*subclass InvalidLLEException is extending the superclass Exception*/
```

```
{
```

```
InvalidLLException(String msg)
```

```
{
```

```
    super(msg);
```

```
}
```

```
public String toString(){ // overriding the toString method.
```

```
    return "The person hasn't issued for the LLR yet";
```

```
}
```

```
}
```

```
class AccidentException extends Exception /*subclass AccidentException is extending the superclass  
Exception.*/
```

```
{
```

```
AccidentException(String msg)
```

```
{
```

```
    super(msg);
```

```
}
```

```
public String toString(){ // overriding the toString method.
```

```
    return "The person has many accidents case on him";
```

```
}
```

```
}
```

```
public class Tw8 {
```

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

```
        License l = new License(); // object of class license is instantiated.
```

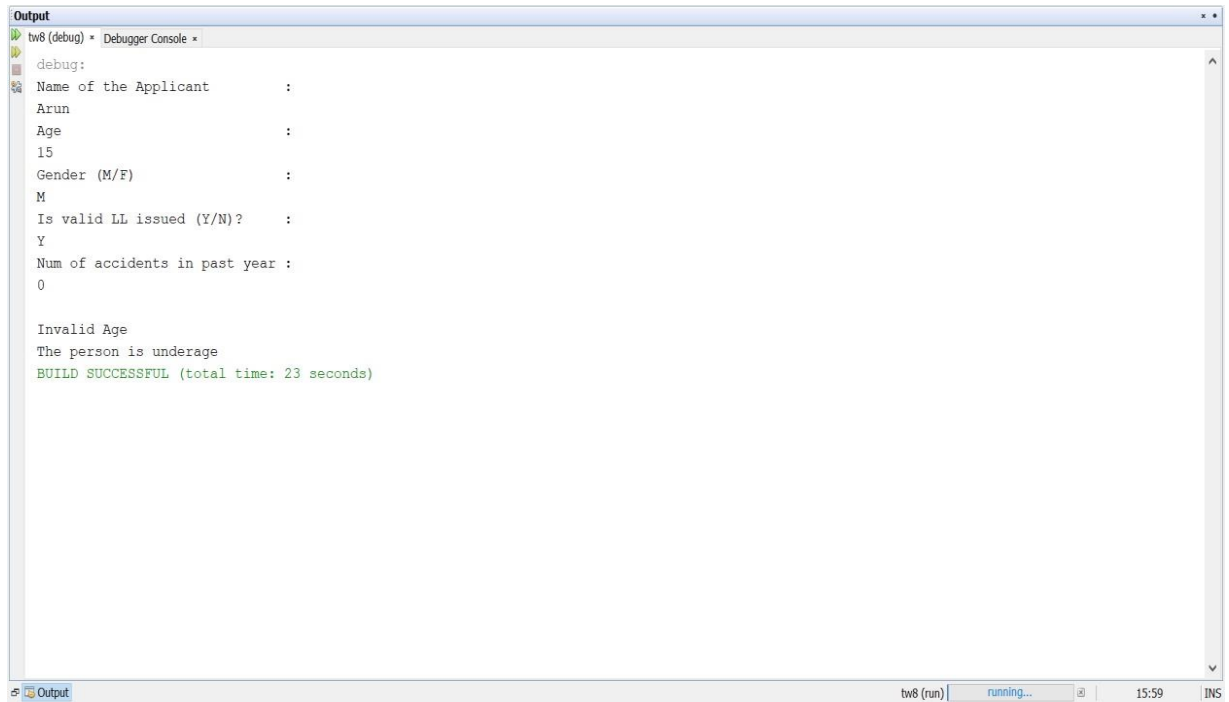
```
l.validateData(); // method validate data is called
```

```
    }
```

```
}
```

Output :

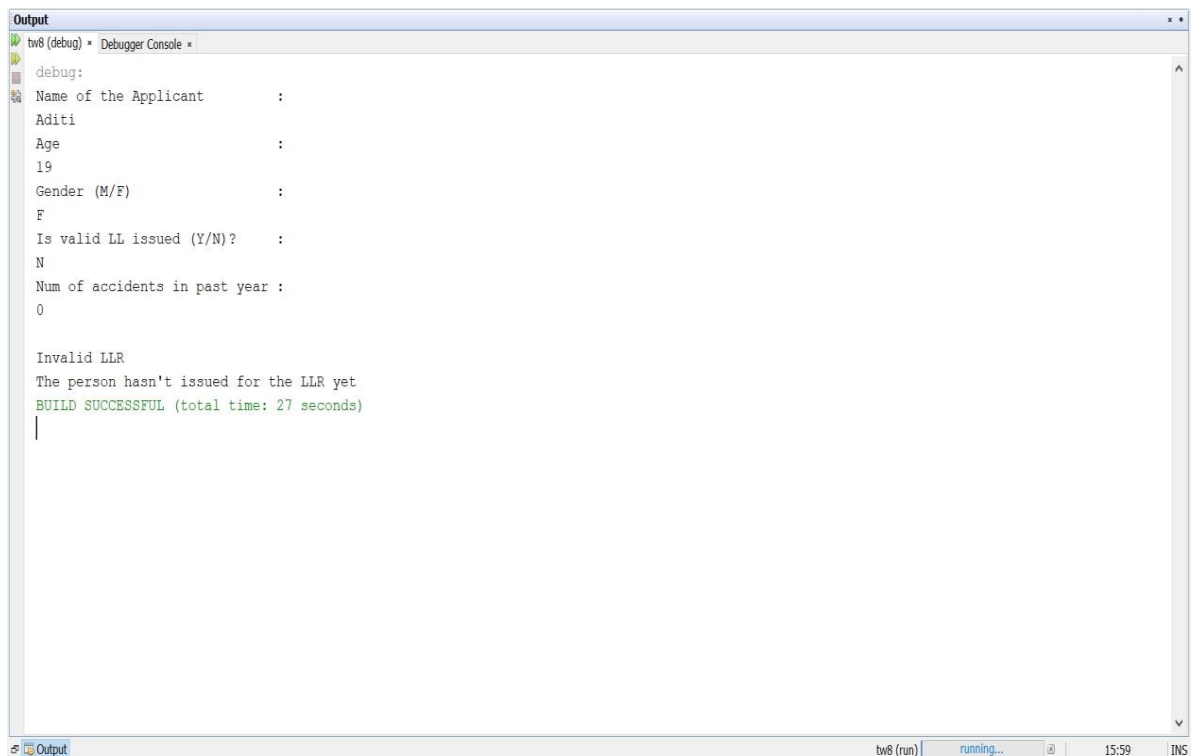
Case 1:



```
debug:
Name of the Applicant      :
Arun
Age                        :
15
Gender (M/F)              :
M
Is valid LL issued (Y/N)?  :
Y
Num of accidents in past year :
0

Invalid Age
The person is underage
BUILD SUCCESSFUL (total time: 23 seconds)
```

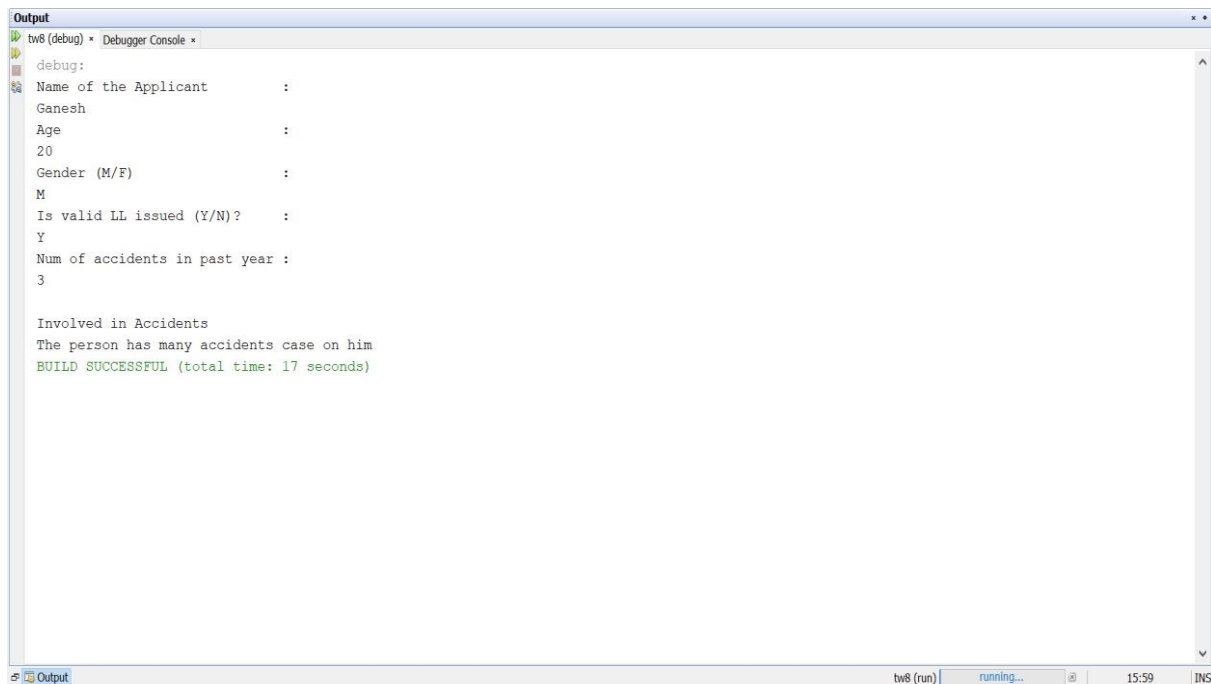
Case 2:



```
debug:
Name of the Applicant      :
Aditi
Age                        :
19
Gender (M/F)              :
F
Is valid LL issued (Y/N)?  :
N
Num of accidents in past year :
0

Invalid LLR
The person hasn't issued for the LLR yet
BUILD SUCCESSFUL (total time: 27 seconds)
```

Case 3:



The screenshot shows the 'Output' window of a NetBeans IDE. The window title is 'Output'. Inside, there is a tab labeled 'tw8 (debug) * Debugger Console *'. The output text is as follows:

```
debug:
Name of the Applicant      :
Ganesh
Age                        :
20
Gender (M/F)              :
M
Is valid LL issued (Y/N)?  :
Y
Num of accidents in past year :
3

Involved in Accidents
The person has many accidents case on him
BUILD SUCCESSFUL (total time: 17 seconds)
```

At the bottom of the window, there is a status bar showing 'tw8 (run)' and 'running...' with a small icon, and the time '15:59' and 'JNS'.

Outcomes of the Experiment : At the end of the laboratory sessions the students should be able to

1. Demonstrate the use Exception Handling in solving real-life problems.
2. Identify appropriate variables and their types
3. Identify appropriate user definedExceptions
4. Identify the control statements needed to meet the problem requirements.

Conclusions : From the given problem statement, we could identify the necessary variables of appropriate type, and looping/control statements and the necessary program logic. The program was written in NetBeans IDE(Mention the one you actually used) by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We came to know about the Exceptions and ways of handling those Exceptions also understand how to define our own Exceptions and implement it on the given problem statement.

We also understood the use of built-in class System and its method println to display the result. The program was executed for two-three sets of input and result obtained were verified to be correct and recorded.

Practice Problem:

Demonstrate a class CustomerAccount that has acctNum, custName and balance as member variables and a constructor to initialize these. Implement withdraw and depositAmount methods that accept amount as argument and it must throw a user-defined exception called InsufficientBalance/InvalidAmount exception when amount is greater than balance/ amount is negative respectively. Design two classes InsufficientBalance and InvalidAmount that extend the Exception class and override toString method. Demonstrate the working of the user-defined exceptions by instantiating an object of CustomerAccount class and invoking withdraw and depositAmount in try...catch....finally block.

Program Source Code :

```
package tw8p;

import java.util.Scanner;

class CustomerAccount{ // class CustomerAccount is created

    int accNo;

    String custName;

    double balance;// required variables are initialised

    CustomerAccount(int accNo,String custName,double balance)// constructor of the class

    {

        this.accNo = accNo;

        this.custName = custName;

        this.balance = balance;

        // variables are initialised in the constructor

    }

    void withdrawAmt(double Amt) // method used to withdraw the amount

    {

        try{// try block to check for exception when the amt is withdrawn

            if(balance-Amt <0)

                throw new InsufficientBalException("Insufficient Balance");

            balance-=Amt;
```

```
System.out.println("Balance in the account is "+ balance);

    }

    catch (InsufficientBalException e) // catch block to deal with the insufficient Balance Exception

    {

System.out.println(e.getMessage());

System.out.println(e);

System.out.println();

    }

}

void depositAmt(double Amt) // method used to deposit the amount

{

    try // try block to check for exception while depositing the amount

    {

if(Amt<0)

        throw new InvalidAmtException("Invalid amount");

        balance+=Amt;

System.out.println("Balance in the account is "+ balance);

    }

catch( InvalidAmtException e) // catch block to deal with the invalid amount exception

    {

System.out.println(e.getMessage());

System.out.println(e);

System.out.println();

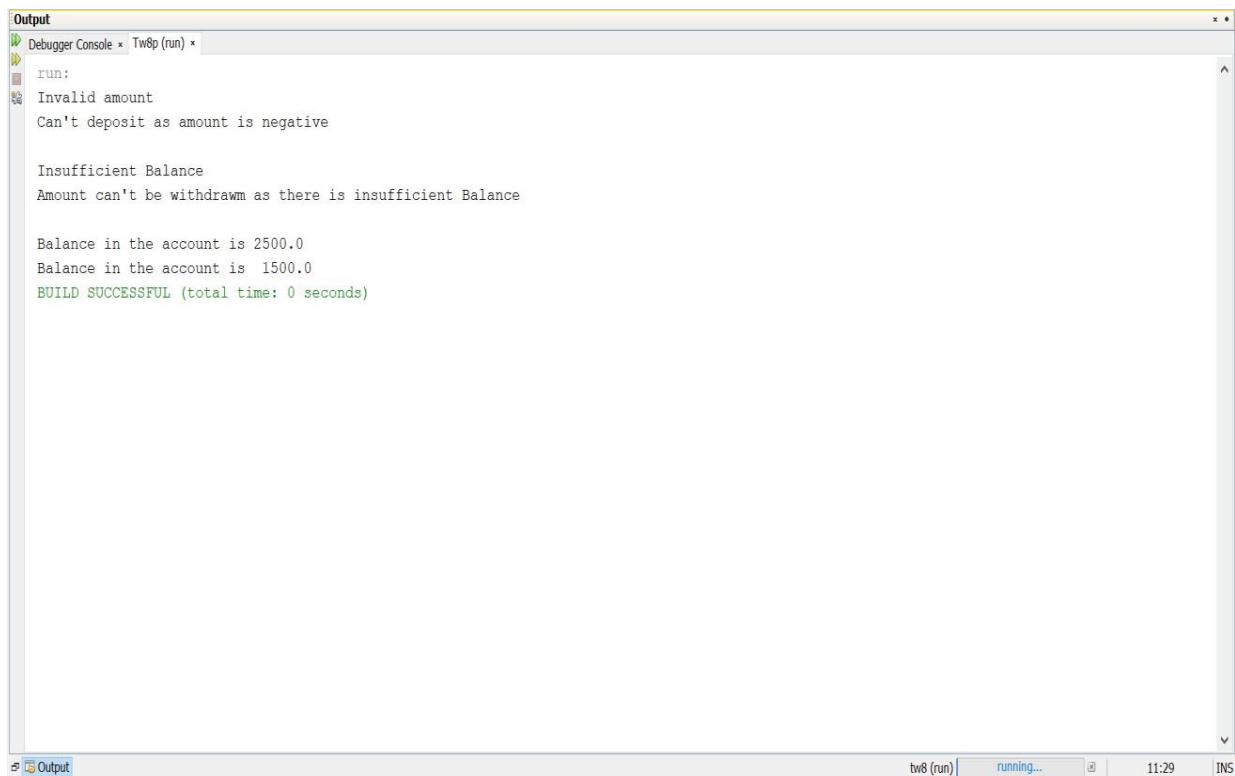
    }

}
```

```
    }  
}  
  
class InsufficientBalException extends Exception{ /* subclass insufficientBalException is extending  
superclass Exception*/  
  
InsufficientBalException(String msg)  
  
    {  
  
        super(msg);  
  
    }  
  
    public String toString()// toString method is overridden  
  
    {  
  
        return "Amount can't be withdrawm as there is insufficient Balance";  
  
    }  
}  
  
class InvalidAmtException extends Exception{ /* subclass invalidAmtException is extending  
superclass Exception*/  
  
InvalidAmtException(String msg)  
  
    {  
  
        super(msg);  
  
    }  
  
    public String toString()// toString method is overridden  
  
    {  
  
        return "Can't deposit as amount is negative";  
  
    }  
}
```

```
public class Tw8p {  
  
    public static void main(String[] args) {  
  
        CustomerAccount c =new CustomerAccount(101,"Amit shah",1000);/*object is instantiated of the  
        class CustomerAccount */  
  
        c.depositAmt(-500);// depositAmt method is called  
  
        c.withdrawAmt(1200);// withdrawAmt method is called  
  
        c.depositAmt(1500);  
  
        c.withdrawAmt(1000);  
  
    }  
  
}
```

Output :



```
run:  
Invalid amount  
Can't deposit as amount is negative  
  
Insufficient Balance  
Amount can't be withdrawm as there is insufficient Balance  
  
Balance in the account is 2500.0  
Balance in the account is 1500.0  
BUILD SUCCESSFUL (total time: 0 seconds)
```

tw8 (run) | running... | 11:29 | INS

Title of the Experiment: STRING HANDLING**Experiment No. 9****Date : 24/10/20****Problem Statement :**

9.1) Read a string containing 3_4 words using Scanner class object. Split it into words and for each word check if it's palindrome by writing a function isPalindrome(String the myWord, int s, int e) which return true if its palindrome else return false. Where s is start index and e is end index of the input myWord. Print it in uppercase if it is palindrome else reverse the string and print it in lowercase. Use appropriate string functions to implement the above problem statement.

Objectives of the Experiment :

1. Using String Handling concept to store the string data in the main memory
2. manipulating the data of the String, retrieving the part of the String etc
3. Using String Handling learn a lot of concepts that can be performed on a string such as concatenation of string, comparison of string, find sub string etc

Program Source Code :

```
import java.util.Scanner;

public class TW9 {
    public static void main(String[] args) {
        String str;
        String[] words;
        Scanner input = new Scanner(System.in);
        System.out.println("Enter The String...!!!");
        str = input.next();
        words = str.split(" ");
        for (String s : words) {
            if (isPalindrome(s, 0, s.length() - 1)) {
                System.out.println(s.toUpperCase());
                System.out.println("It is A Palindrome");
            } else {
                System.out.println(reverseString(s).toLowerCase());
                System.out.println("Not A Palindrome");
            }
        }
    }
}
```

```

    }
}

public static boolean isPalindrome (String word,int s, int t){
    if (word.charAt(s) == word.charAt(t)) {
        if (s < t)
            return isPalindrome(word, s + 1, t - 1);
        else if (s == t || s == t + 1)
            return true;
        }
    return false;
}

public static String reverseString(String s){
    String rs = " ";
    for (int i = s.length() - 1; i >= 0; i--)
        rs = rs + s.charAt(i);
    return rs;
}
}

```

Output :

Case 1:

The screenshot shows the IntelliJ IDEA interface with the 'Run' window open. The output of the program is as follows:

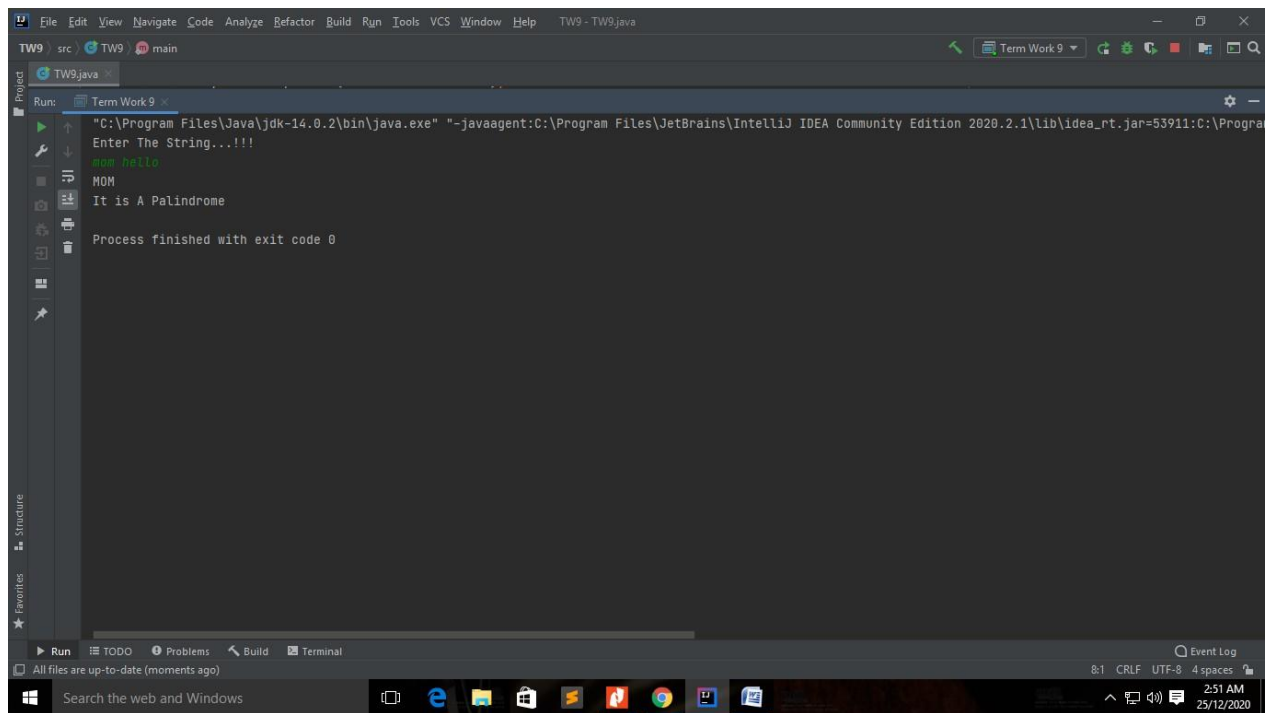
```

"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.1\lib\idea_rt.jar=53908:C:\Progra
Enter The String...!!!
hello mom
olleh
Not A Palindrome
Process finished with exit code 0

```

The status bar at the bottom indicates 'Build completed successfully in 2 sec, 692 ms (2 minutes ago)'.

Case 2:



```
Run: Term Work 9
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.1\lib\idea_rt.jar=53911:C:\Progra
Enter The String...!!!
mom hello
MOM
It is A Palindrome
Process finished with exit code 0
```

Outcomes of the Experiment : At the end of the laboratory sessions the students should be able to

1. Demonstrate the use of String Handling Concept.
2. Understand Java String contains an immutable sequence of Unicode characters
3. Understand string is an object that represents a sequence of characters or char values.
The *java.lang.String* class is used to create a Java string object
4. Will Understand the Importance of String Handling Functions
5. Will understand the use of Various use of String Functions

Conclusions : From the given problem statement, we could identify the necessary variables of appropriate type, and looping/control statements and the necessary program logic. The program was written in IntelliJ IDE (Mention the one you actually used) by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We also understood the use of built-in class `System` and its method `println` to display the result. The program was executed for two-three sets of input and result obtained were verified to be correct and recorded.

Strings are a sequence of characters and are widely used in Java programming. In the Java programming language, strings are objects. The String class has over 60 methods and 13 constructors.

The String class also includes a number of utility methods, among them `split()`, `toLowerCase()`, `toUpperCase()`, and `valueOf()`. The latter method is indispensable in converting user input strings to numbers. The Number subclasses also have methods for converting strings to numbers and vice versa.

In addition to the String class, there is also a String Builder Class In Java class. Working with String Builder Class In Java objects can sometimes be more efficient than working with strings. The String Builder Class In Java class offers a few methods that can be useful for strings, among them `reverse()`. In general, however, the String class has a wider variety of methods.

A string can be converted to a string builder using a `StringBuilder` constructor. A string builder can be converted to a string with the `toString()` method.

Problem Statement(Practice) :

9.2) Two strings will be anagram to each other if and only if they contain the same number of characters (order of the characters doesn't matter). That is, If the two strings are anagram to each other, then one string can be rearranged to form the other string. For Example: creative and reactive are anagrams. Write a Java program to test whether two strings are anagrams or not. (listen and silent, stressed and desserts, dusty and study)

Program Source Code :

```
import java.util.Arrays;
import java.util.Scanner;

class TW9b {
    public static void main(String[] args) {
        String str1;
        String str2;
        Scanner input=new Scanner(System.in);
        System.out.println("Enter String 1...!!!");
        str1=input.next();
        System.out.println("Enter String 2...!!!");
        str2=input.next();

        // check if length is same
        if(str1.length() == str2.length()) {

            // convert strings to char array
            char[] charArray1 = str1.toCharArray();
            char[] charArray2 = str2.toCharArray();

            // sort the char array
            Arrays.sort(charArray1);
            Arrays.sort(charArray2);

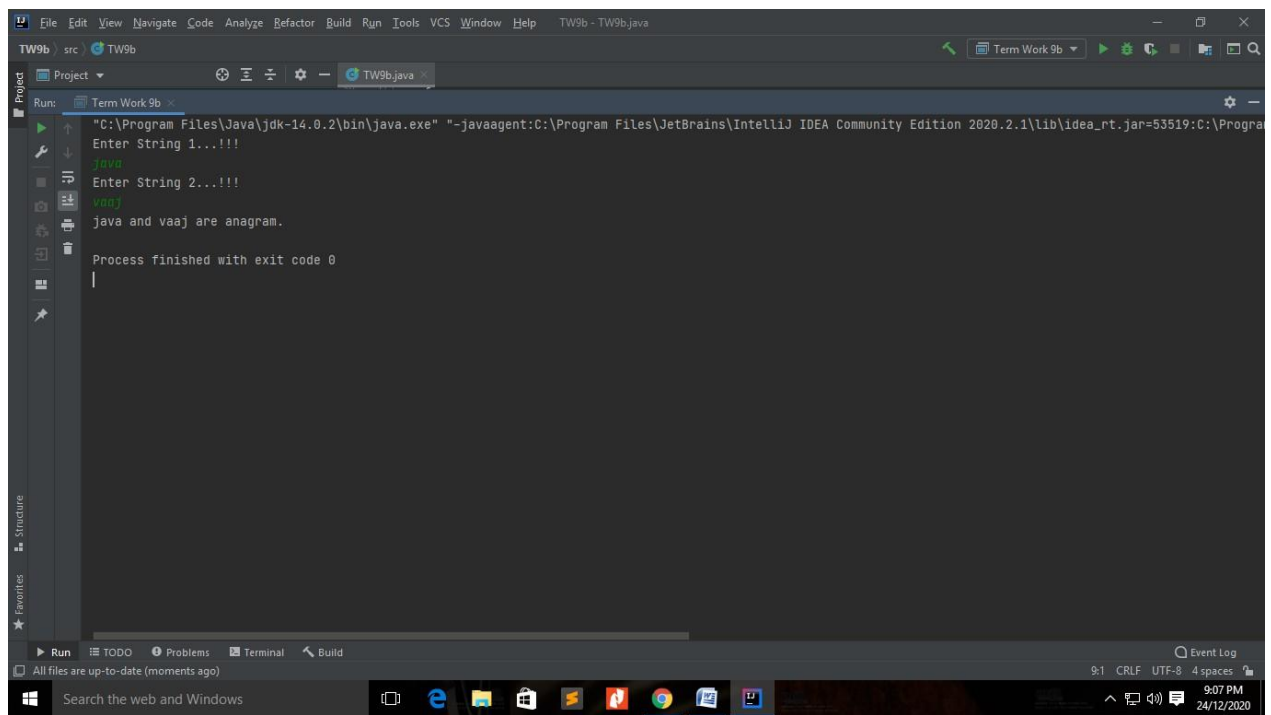
            // if sorted char arrays are same
            // then the string is anagram
            boolean result = Arrays.equals(charArray1, charArray2);

            if(result) {
                System.out.println(str1 + " and " + str2 + " are anagram.");
            }
            else {
```

```
        System.out.println(str1 + " and " + str2 + " are not  
anagram.");  
    }  
    }  
    else {  
        System.out.println(str1 + " and " + str2 + " are not anagram.");  
    }  
}  
}
```

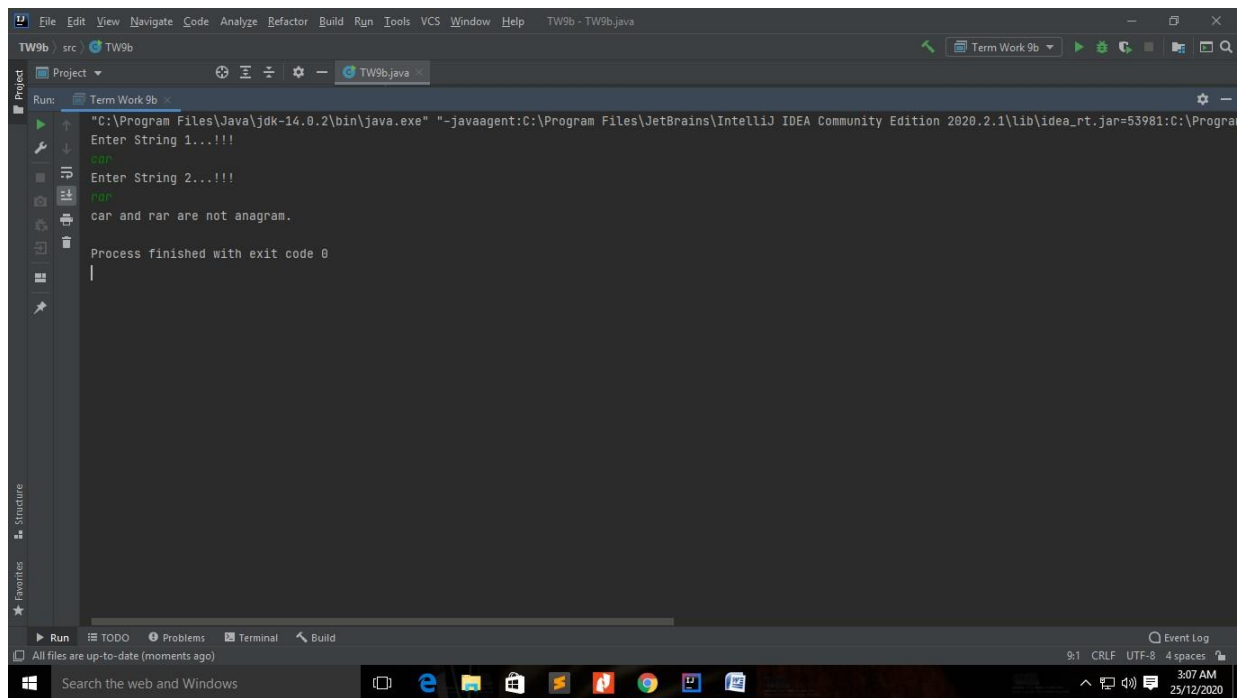
Output :

Case1



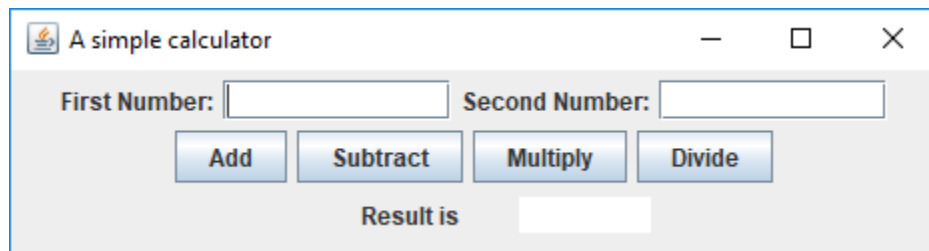
```
Run: Term Work 9b x  
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2020.2.1\lib\idea_rt.jar=53519:C:\Progra  
Enter String 1...!!!  
java  
Enter String 2...!!!  
vaaj  
java and vaaj are anagram.  
Process finished with exit code 0  
|
```

Case 2



Title of the Experiment (GUI – [Graphical User Interface])**Experiment No.**____10_____**Date :** _24/12/20_____**Problem Statement :**

Design and develop a GUI application as shown below. Assume the two numbers to be integers. The application must check for invalid division condition and throw an appropriate exception.

**Objectives of the Experiment :**

1. Learn to design and develop GUI (Graphical User Interface).
2. Understand the use of GUI in a real-life application.
3. Learn to Display the result in a readable/proper format .

Program Source Code :

```
package tw10;

import javax.swing.JOptionPane;

public class Calculator extends javax.swing.JFrame {

    /**
     * Creates new form Calculator
     */
    public Calculator() {
        initComponents();
    }
}
```

```
/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-BEGIN: initComponents
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    t1 = new javax.swing.JTextField();
    jLabel2 = new javax.swing.JLabel();
    t2 = new javax.swing.JTextField();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jButton3 = new javax.swing.JButton();
    jButton4 = new javax.swing.JButton();
    t3 = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("A simple calculator");

    jLabel1.setText("First Number");

    jLabel2.setText("Second Number");

    jButton1.setText("Add");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        jButton1ActionPerformed(evt);
    }
});
```

```
jButton2.setText("Subtract");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});
```

```
jButton3.setText("Multiply");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});
```

```
jButton4.setText("Divide");
jButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton4ActionPerformed(evt);
    }
});
```

```
jLabel3.setText("Result is ");
```

```
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup())

.addGap(29, 29, 29)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup())

.addComponent(jButton1)

.addGap(42, 42, 42)

.addComponent(jButton2)

.addGap(40, 40, 40)

.addComponent(jButton3)

.addGap(73, 73, 73)

.addComponent(jButton4)

.addContainerGap())

.addGroup(layout.createSequentialGroup())

.addGap(0, 0, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createSequentialGroup())

.addComponent(jLabel1)

.addGap(18, 18, 18)

.addComponent(t1, javax.swing.GroupLayout.PREFERRED_SIZE, 104,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addComponent(jLabel3))

.addGap(57, 57, 57)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())

.addComponent(jLabel2)

.addGap(27, 27, 27)

.addComponent(t2, javax.swing.GroupLayout.PREFERRED_SIZE, 108,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```

        .addGap(57, 57, 57))

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())

        .addComponent(t3, javax.swing.GroupLayout.PREFERRED_SIZE, 156,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addGap(179, 179, 179))))))

);

layout.setVerticalGroup(

    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(layout.createSequentialGroup()

            .addGap(90, 90, 90)

            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(t2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jLabel2)

                .addComponent(jLabel1)

                .addComponent(t1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

            .addGap(32, 32, 32)

            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(jButton2)

                .addComponent(jButton3)

                .addComponent(jButton4)

                .addComponent(jButton1))

            .addGap(62, 62, 62)

            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(jLabel3)

                .addComponent(t3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

        .addContainerGap(97, Short.MAX_VALUE))

);

```



```
        pack();

    }// </editor-fold>{//GEN-END:initComponents

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton1ActionPerformed

        t3.setText(""+(Integer.parseInt(t1.getText())+Integer.parseInt(t2.getText())));

    }//GEN-LAST:event_jButton1ActionPerformed

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton2ActionPerformed

        t3.setText(""+(Integer.parseInt(t1.getText())-Integer.parseInt(t2.getText())));

    }//GEN-LAST:event_jButton2ActionPerformed

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton3ActionPerformed

        t3.setText(""+Integer.parseInt(t1.getText())*Integer.parseInt(t2.getText()));

    }//GEN-LAST:event_jButton3ActionPerformed

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jButton4ActionPerformed

        int a = Integer.parseInt(t1.getText());

        int b = Integer.parseInt(t2.getText());

        try{

            float res = a /(float)b;

            t3.setText(""+ String.format("%.2f",res));

        }

    }
```

```

        catch(ArithmeticException e)
        {
            JOptionPane.showMessageDialog(null, "invalid arithmetic operation!");
        }
    }
} //GEN-LAST:event_jButton4ActionPerformed

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */

    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
            javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    }

    catch (ClassNotFoundException ex) {

        java.util.logging.Logger.getLogger(Calculator.class.getName()).log(java.util.logging.Level.SEVERE,
            null, ex);
    }
}

```

```

catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Calculator.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);

    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Calculator.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Calculator.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);

    }
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new Calculator().setVisible(true);
    }
});
}

// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;

```

```
private javax.swing.JTextField t1;

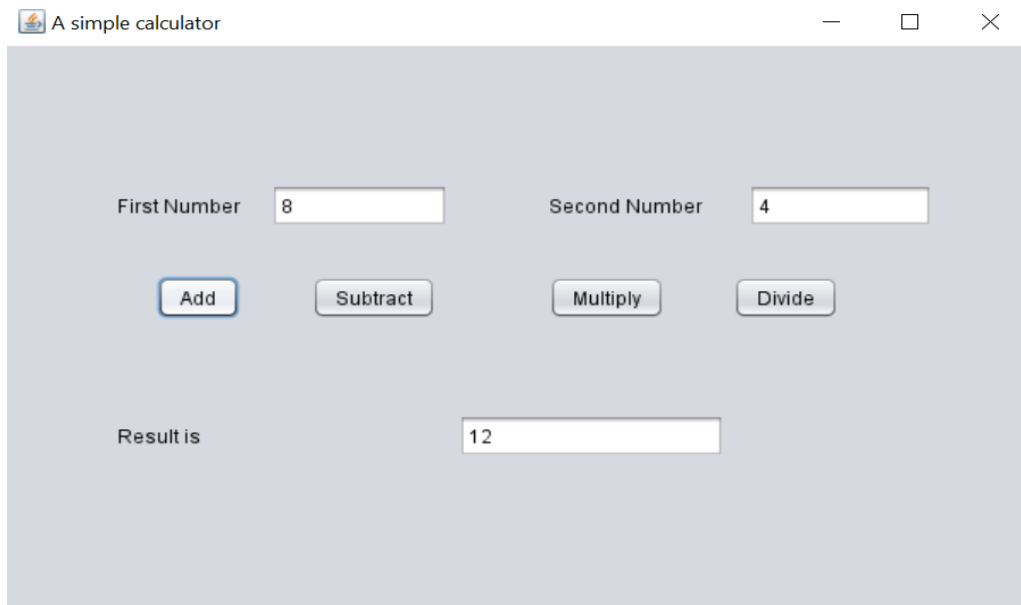
private javax.swing.JTextField t2;

private javax.swing.JTextField t3;

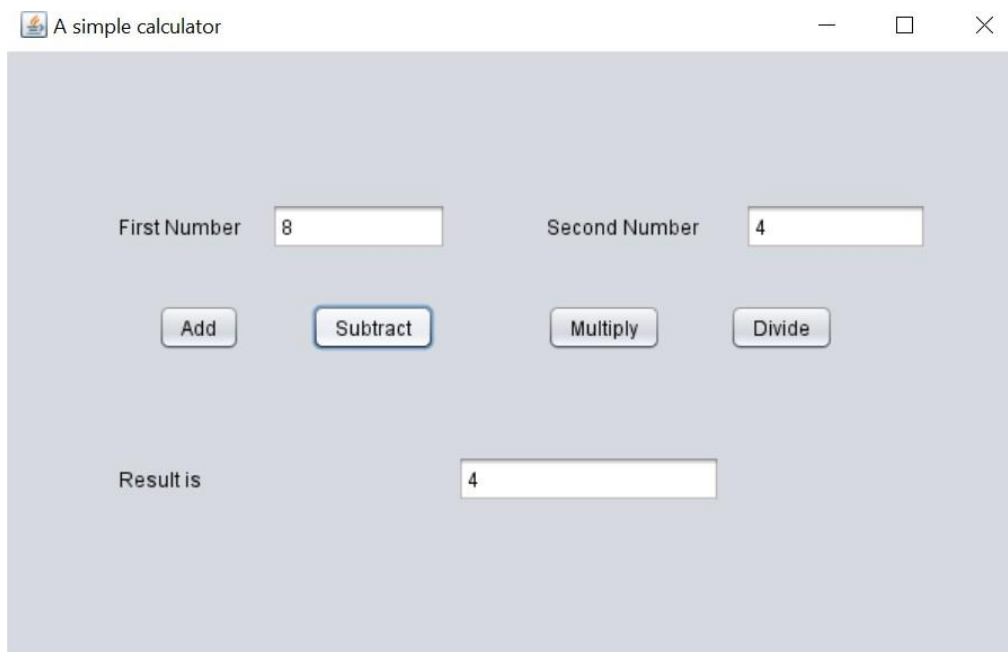
// End of variables declaration//GEN-END:variables
}
```

Output :

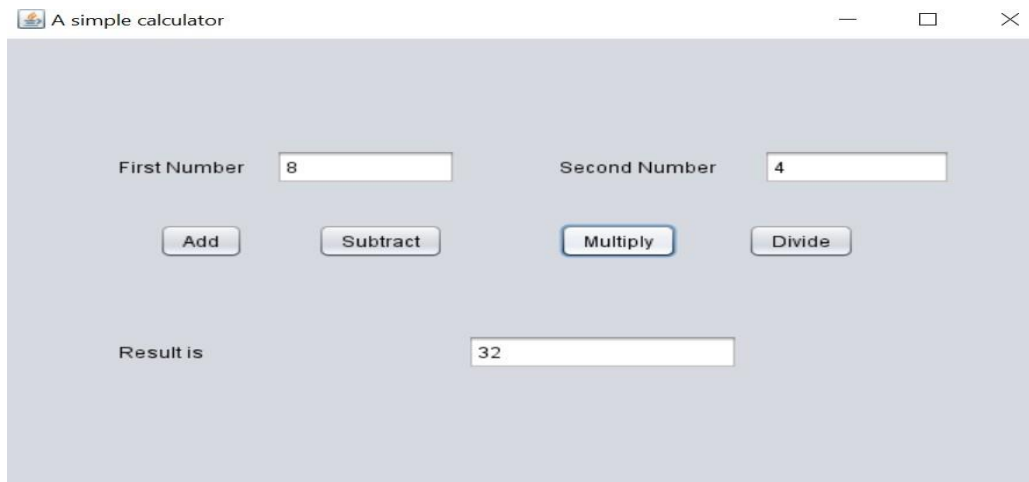
Case 1:



Case 2:



Case 3:



A simple calculator

First Number: 8

Second Number: 4

Buttons: Add, Subtract, Multiply, Divide

Result is: 32

The screenshot shows a window titled "A simple calculator". Inside, there are two input fields: "First Number" with the value "8" and "Second Number" with the value "4". Below these are four buttons: "Add", "Subtract", "Multiply", and "Divide". The "Multiply" button is highlighted with a blue border. At the bottom, there is a label "Result is" followed by an input field containing the value "32".

Case 4:



A simple calculator

First Number: 7

Second Number: 6

Buttons: Add, Subtract, Multiply, Divide

Result is: 1.17

The screenshot shows a window titled "A simple calculator". Inside, there are two input fields: "First Number" with the value "7" and "Second Number" with the value "6". Below these are four buttons: "Add", "Subtract", "Multiply", and "Divide". The "Divide" button is highlighted with a blue border. At the bottom, there is a label "Result is" followed by an input field containing the value "1.17".

Case 5:



A simple calculator

First Number: 5

Second Number: 0

Buttons: Add, Subtract, Multiply, Divide

Result is:

Message: invalid arithmetic operation!

OK

The screenshot shows a window titled "A simple calculator". Inside, there are two input fields: "First Number" with the value "5" and "Second Number" with the value "0". Below these are four buttons: "Add", "Subtract", "Multiply", and "Divide". The "Divide" button is highlighted with a blue border. At the bottom, there is a label "Result is" followed by an empty input field. A message box is overlaid on the calculator window. The message box has a title bar "Message" and a close button. It contains an information icon (a blue circle with a white 'i') and the text "invalid arithmetic operation!". There is an "OK" button at the bottom right of the message box.

Outcomes of the Experiment : At the end of the laboratory sessions the students should be able to

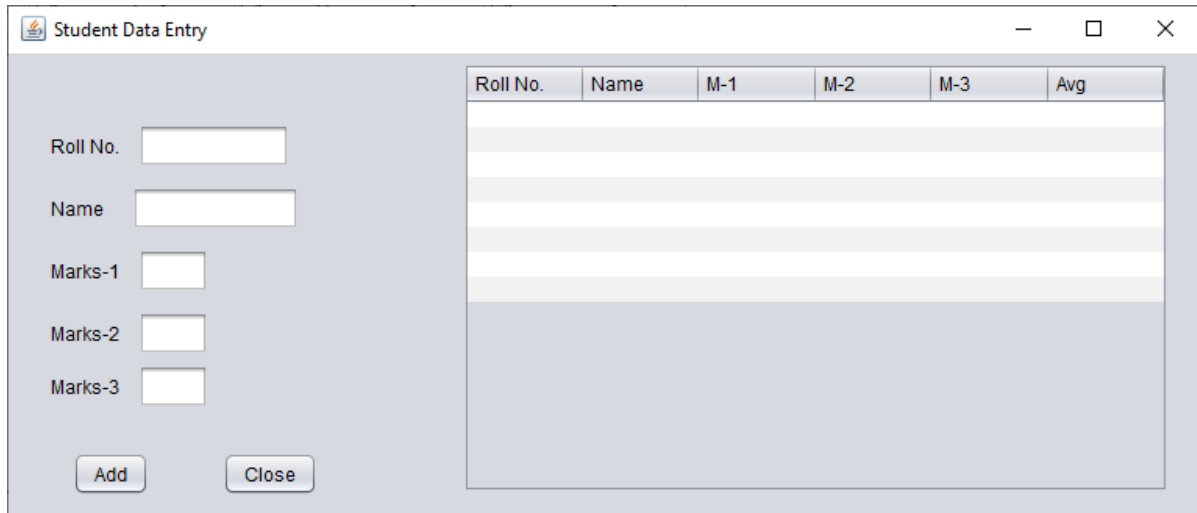
1. Demonstrate the use of GUI in solving real-life problems.
2. Identify appropriate variables and their types.
3. Demonstrate the design of a GUI.
4. Identify the control statements needed to meet the problem requirements.

Conclusions : From the given problem statement, we could identify the necessary variables of appropriate type, and looping/control statements and the necessary program logic. The program was written in NetBeans IDE(Mention the one you actually used) by creating a project. We understood the usage of the IDE in typing the code, debugging, running the program and observing the output. We came to know how to design and develop a GUI application, we learned how to apply the business logic and controller to a JFrame. we also came to know how to apply exception handling for a particular situation when an exception occurs in the operation of a GUI application.

We also understood the use of built-in class System and its method println to display the result. The program was executed for two-three sets of input and result obtained were verified to be correct and recorded.

Practice problem:

Design and develop a GUI application to accepts student details as shown in the GUI below. Allow the user to add records one after the other. Once the user clicks on close, compute the average score and display the details using JTable component.



Rename close as “Summarize”

Source code:

```
package tw10;
```

```
import java.util.ArrayList;
```

```
public class StudentData extends javax.swing.JFrame {  
    class StudentRecord { //Inner class  
        String name;  
        int rNo, m1, m2,m3;  
        StudentRecord(int rNo, String name, int m1, int m2, int m3) {  
            this.name = name;  
            this.rNo = rNo;  
            this.m1 = m1;  
            this.m2 = m2;  
            this.m3 = m3;  
        }  
    }  
}
```

```

    }

    ArrayList<StudentRecord> records = new ArrayList<StudentRecord>();

    /**
     * Creates new form StudentData
     */
    public StudentData() {
        initComponents();

    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jTextField2 = new javax.swing.JTextField();
        jTextField3 = new javax.swing.JTextField();
        jTextField4 = new javax.swing.JTextField();
        jTextField5 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
    }

```



```
jScrollPane1 = new javax.swing.JScrollPane();

jTable1 = new javax.swing.JTable();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("Student Details");

jLabel1.setText(" Roll No. :");


jLabel2.setText("Name :");


jLabel3.setText("Marks 1 :");


jLabel4.setText(" Marks 2 :");


jLabel5.setText(" Marks 3 :");


jTextField4.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jTextField4ActionPerformed(evt);

    }

});


jButton1.setText("Add");

jButton1.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton1ActionPerformed(evt);

    }

});


jButton2.setText("Summarize");

jButton2.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {
```



```

        .addGroup(layout.createSequentialGroup())
        .addComponent(jButton1)
        .addGap(41, 41, 41)
        .addComponent(jButton2))
    .addGroup(layout.createSequentialGroup())
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jLabel5)
            .addComponent(jLabel4)
            .addComponent(jLabel3)
            .addComponent(jLabel2)
            .addComponent(jLabel1))
        .addGap(33, 33, 33)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                .addComponent(jTextField2, javax.swing.GroupLayout.DEFAULT_SIZE, 76,
Short.MAX_VALUE)
                .addComponent(jTextField3)
                .addComponent(jTextField4)
                .addComponent(jTextField5))
            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 93,
javax.swing.GroupLayout.PREFERRED_SIZE))))
        .addGap(18, 18, 18)
        .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 626,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(96, Short.MAX_VALUE))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(47, 47, 47)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```
.addComponent(jLabel1)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel2)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel3)

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel4)

.addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel5)

.addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGap(99, 99, 99)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton1)

.addComponent(jButton2))

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addGroup(layout.createSequentialGroup())

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

.addGap(0, 103, Short.MAX_VALUE))

);
```

```
        pack();
    } // </editor-fold>

    private void jTextField4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        int rowCount = 0;

        for(StudentRecord s : records) {
            jTable1.setValueAt(s.rNo, rowCount, 0);
            jTable1.setValueAt(s.name, rowCount, 1);
            jTable1.setValueAt(s.m1, rowCount, 2);
            jTable1.setValueAt(s.m2, rowCount, 3);
            jTable1.setValueAt(s.m3, rowCount, 4);
            float avg = (s.m1+s.m2+s.m3) / 3.0f;
            jTable1.setValueAt(String.format("%.2f",avg), rowCount, 5);
            rowCount++;
        }
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        StudentRecord s1 = new StudentRecord(Integer.parseInt(jTextField1.getText()),
            jTextField2.getText(), Integer.parseInt(jTextField3.getText()),
            Integer.parseInt(jTextField4.getText()),
            Integer.parseInt(jTextField5.getText()));
        records.add(s1);
        jTextField1.setText("");
        jTextField2.setText("");
    }
}
```

```

        jTextField3.setText("");

        jTextField4.setText("");

        jTextField5.setText("");
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
                javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

            java.util.logging.Logger.getLogger(StudentData.class.getName()).log(java.util.logging.Level.SEVERE,
                null, ex);

        } catch (InstantiationException ex) {

            java.util.logging.Logger.getLogger(StudentData.class.getName()).log(java.util.logging.Level.SEVERE,
                null, ex);

        }
    }

```

```
catch (IllegalAccessException ex) {
```

```
java.util.logging.Logger.getLogger(StudentData.class.getName()).log(java.util.logging.Level.SEVERE,  
null, ex);
```

```
}
```

```
catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(StudentData.class.getName()).log(java.util.logging.Level.SEVERE,  
null, ex);
```

```
}
```

```
//</editor-fold>
```

```
/* Create and display the form */
```

```
java.awt.EventQueue.invokeLater(new Runnable() {
```

```
    public void run() {
```

```
        new StudentData().setVisible(true);
```

```
    }
```

```
});
```

```
}
```

```
// Variables declaration - do not modify
```

```
private javax.swing.JButton jButton1;
```

```
private javax.swing.JButton jButton2;
```

```
private javax.swing.JLabel jLabel1;
```

```
private javax.swing.JLabel jLabel2;
```

```
private javax.swing.JLabel jLabel3;
```

```
private javax.swing.JLabel jLabel4;
```

```
private javax.swing.JLabel jLabel5;
```

```
private javax.swing.JScrollPane jScrollPane1;
```

```
private javax.swing.JTable jTable1;
```

```
private javax.swing.JTextField jTextField1;
```

```
private javax.swing.JTextField jTextField2;
```

```
private javax.swing.JTextField jTextField3;  
  
private javax.swing.JTextField jTextField4;  
  
private javax.swing.JTextField jTextField5;  
  
// End of variables declaration  
}
```

Output :



Roll No.	Name	M-1	M-2	M-3	Average
1	Arun	20	25	28	24.33
2	Chandan	24	26	28	26.00
3	Ganesh	27	28	29	28.00
4	Raghvendra	27	20	19	22.00