**Name :** Mufid Vahora

**Roll no :** 20BCE307

**Subject :** Object Oriented Programming (2CS302)

**Practical :** 9

**Practical Name :** 9A

**Aim :** An interface Polygon containing the members as given below:

float area, float perimeter

void calcArea( ); abstract method to calculate area of a particular polygon given its dimensions

void calcPeri( ); abstract method to calculate perimeter of a particular polygon given its dimensions

void display( ); method to display the area and perimeter of the given polygon.

Create a class Square that implements Polygon and has the following member:

float side

Square(float s); constructor to initialize side of square

Create another class Rectangle that implements Polygon and has the following member:

float length

float breadth

Rectangle(int len, int bre); constructor to initialize length and breadth of a rectangle

Outside the package, create a class that imports the above package an instantiates an object of the Square class and an object of the Rectangle class. Call the above methods on each of the classes to calculate the area and perimeter given the side and the length/breadth of the Square class and the Rectangle class respectively.

## Methodology followed :

**OopPractical9a.java**

```
package p9a2; import Polygon.*;

import java.util.*;
```

```
public class OopPractical9a{
public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
System.out.println("Enter Length and Breadth of Rectangle:");
Rectangle r=new Rectangle(sc.nextFloat(),sc.nextFloat());
r.CalArea();
r.CalPeri();
r.Display(); System.out.println("Enter Length of Square:");
Square s=new Square(sc.nextFloat()); s.CalArea();
s.CalPeri();
s.Display();
sc.close();

}
}
```

**Polygone.java**

```
package p9a1;

public interface Polygon { public void Display();

public void CalArea();

public void CalPeri();
}
```

```
}
```

**Rectangle.java**

```
package p9a1;

public class Rectangle implements Polygon { float length, breadth;

float area, perimeter;


public Rectangle(float length, float breadth) { this.length = length;

this.breadth = breadth;


}




public void Display() {

System.out.println("Area of rectangle with length " + length + " and breadth " + breadth + " is "
+ area);

System.out.println("Perimeter of rectangle with length " + length + " and breadth " + breadth + "
is " + perimeter);

}


public void CalArea() {

area = length * breadth;


}


public void CalPeri() {

perimeter = 2 * (length + breadth);


}

}
```

**Square.java**

```
package p9a1;


public class Square implements Polygon { float length;
   float area, perimeter;


   public Square(float length) { this.length = length;
   }


   public void Display() {
   System.out.println("Area of square with side " + length + " is " + area);
   System.out.println("Perimeter of square with side " + length + " is " + perimeter);
   }


   public void CalArea() { area = length * length;
   }


   public void CalPeri() { perimeter = 4 * length;
   }




   }
```

## Input / Output :

```
Enter Length and Breadth of Rectangle: 10 20

Area of rectangle with length 10 and breadth 20 is 200
```

```
Perimeter of rectangle with length 10 and breadth 20 is 60


Enter Length of Square:
10
Area of square with side 10 is 100 Perimeter of square with side 10 is 40
```

**Conclusion :** From this practical we can calculate the perimeter and area of rectangle and square.

**Practical Name :** 9B

**Aim :** Create a class called CalcAverage that has the following method:

public double avgFirstN(int N)

This method receives an integer as a parameter and calculates the average of first N natural numbers. If N is not a natural number, throw an exception IllegalArgumentException with an appropriate message.

**Methodology followed :**

```
import java.util.*;
class CalcAverage{
  public double avgFirst(int n){
    double avg = 0;
    for(int i =1;i<=n;++i){
      avg += i;
    }
    avg = (double)avg/n;
    return avg;
  }
}
class OopPractical9b
{
    public static void main(String[] args)
    {
        CalcAverage c = new CalcAverage();
        Scanner sc=new Scanner(System.in);
        int n,i=1;
        double avg;
            System.out.print("Enter a natural number : ");
            try
            {
                n=sc.nextInt();
                if(n<=0)
                {
                    throw new IllegalArgumentException("n must be positive");
                }
                avg=c.avgFirst(n);
                System.out.println("Average of " + n + " natural numbers is = " + avg);
            }
            catch(IllegalArgumentException e)
            {
                System.out.println("Enter a positive number" + e);
            }
        }
}
```

**Input / Output :**

```
Enter a natural number : 4

Average of 4 natural numbers is = 2.5
```

**Conclusion :** From this practical we can able to find the average of any natural number.

**Practical Name :** 9C

**Aim :** Create a class Number having the following features:

Attributes:

| | |
|---|---|
| int | first number |
| int | second number |
| result | double |

stores the result of arithmetic operations performed on a and b

Member functions:

- Number(x, y) : constructor to initialize the values of a and b
- add( ) : stores the sum of a and b in result
- sub( ) : stores difference of a and b in result
- mul( ) : stores product in result
- div( ) : stores a divided by b in result

Test to see if b is 0 and throw an appropriate exception since division by zero is undefined.

Display a menu to the user to perform the above four arithmetic operations.

**Methodology followed :**

```
import java.util.Scanner;

class Number

{

   int first,second;
```

```
double result;
Number(int x, int y)
{
      this.first=x;
      this.second=y;
}
void add()
{
      this.result=this.first+this.second;
}
void sub()
{
       this.result=this.first-this.second;
}
void mul()
{
       this.result=this.first*this.second;
}
void div()
{
       int res=0;
       try
       {
       if(this.second==0)
          {
                  res=1;
                  throw new Exception("Division by 0 is not defined!");
```

```
            }
          }catch(Exception e)
    {
    e.printStackTrace();
    }
       if(res==1)System.exit(1);
       this.result=this.first/this.second;
    }
}

public class OopPractical9c
{
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter First Number : ");
int first=sc.nextInt();
System.out.println("Enter Second Number : ");
int second=sc.nextInt();
Number n=new Number(first,second);
boolean b=true;
while(b)
{
System.out.println("1. Add Numbers");
System.out.println("2. Subtract Numbers");
System.out.println("3. Multiply Numbers");
System.out.println("4. Division Numbers");
System.out.println("5. Exit Program");
int choice=sc.nextInt();
if(choice==1)n.add();
else if(choice==2)n.sub();
else if(choice==3)n.mul();
else if(choice==4)n.div();
else if(choice==5)System.exit(1);

System.out.println("Result : "+n.result);
}
sc.close();
}
}
```

## Input / Output :

```
Enter First Number :
75
Enter Second Number :
45
1. Add Numbers
2. Subtract Numbers
3. Multiply Numbers
4. Division Numbers
5. Exit Program
1
Result : 120.0
1. Add Numbers
2. Subtract Numbers
```

```
3. Multiply Numbers

4. Division Numbers

5. Exit Program

2

Result : 30.0

1. Add Numbers

2. Subtract Numbers

3. Multiply Numbers

4. Division Numbers

5. Exit Program

3

Result : 3375.0

1. Add Numbers

2. Subtract Numbers

3. Multiply Numbers

4. Division Numbers

5. Exit Program

4

Result : 1.0

1. Add Numbers

2. Subtract Numbers

3. Multiply Numbers

4. Division Numbers

5. Exit Program

5
```

**Conclusion :** In this practical first we enter two numbers and then we can able to add them subtract , multiplication and division.

**Practical Name :** 9D

**Aim :** Create a class BankAccount having the members as given below:

accNo integer

custName string

accType string (indicates ‘Savings’ or ‘Current’)

balance float

Include the following methods in the BankAccount class:

· void deposit(float amt);

· void withdraw(float amt);

· float getBalance();

deposit(float amt) method allows you to credit an amount into the current balance. If amount is negative, throw an exception NegativeAmount to block the operation from being performed.

withdraw(float amt) method allows you to debit an amount from the current balance. Please ensure a minimum balance of Rs. 1000/- in the account for savings account and Rs. 5000/- for current account, else throw an exception InsufficientFunds and block the withdrawal operation. Also throw an exception NegativeAmount to block the operation from being performed if the amt parameter passed to this function is negative.

getBalance( ) method returns the current balance. If the current balance is below the minimum required balance, then throw an exception LowBalanceException accordingly.

Have constructor to which you will pass, accno, cust_name, acctype and initial balance.

And check whether the balance is less than 1000 or not in case of savings account and less than 5000 in case of a current account. If so, then raise a LowBalanceException.

In either case if the balance is negative then raise the NegativeAmount exception accordingly.

## Methodology followed :

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

```
class BankAccount
{
  int accNo;
  String custName;
  String accType ;//(indicates ‘Savings’ or ‘Current’)
  float balance;
  BankAccount(int a,String c,String acc,float b)
  {
    accNo = a;
    custName = c;
    accType = acc;
    balance = b;
  }
  void deposit(float amt)
  {
    if(amt < 0)
      throw new IllegalArgumentException("Negative amount cannot be entered ");
    else
    {
      balance  = balance + amt;
    }
  }
  void withdraw(float amt)
  {
    float t;
    t = balance - amt;
    if(accType.equals("Saving"))
```

```
 {
   if(t<1000)
   {
     throw new  IllegalArgumentException("Balance must be minimum of Rs.1000");
   }
   else
     balance = t;
 }
 else
 {
   if(t<5000)
   {
     throw new  IllegalArgumentException("Balance must be minimum of Rs.5000");
   }
   else
   balance = t;
 }
}
float getBalance()
{
 return balance;
}
void check(float a,String ac)
{
 if(ac.equals("Saving"))
 {
   if(a<1000)
```

```
        {
          throw new IllegalArgumentException("Balance must be minimum of Rs.1000");
        }
      }
      else
      {
        if(a<5000)
        {
          throw new IllegalArgumentException("Balance must be minimum of Rs.5000");
        }
      }
    }


}
class OopPractical9d
{
  public static void main(String args[])
  {

    Scanner sc = new Scanner(System.in);
    System.out.println("Enter your name : ");
    String c = sc.next();
    System.out.println("Enter your account number : ");
    int a = sc.nextInt();
    System.out.println("Enter your account type (Saving or current) : ");
    String acc = sc.next();
    System.out.println("Enter your balance");
    float b = sc.nextFloat();


      BankAccount ba = new BankAccount(a,c,acc,b);
      ba.check(b,acc);


    int choice = 0;
    float amt;
    do
    {
      System.out.println("1. Deposite\n2. Withdraw\n3. Display balance\n4. Exit\nEnter your
choice => ");
      choice = sc.nextInt();
      switch(choice)
      {
        case 1:
        System.out.println("Enter amount to deposite : ");
        amt = sc.nextFloat();
        ba.deposit(amt);
        break;
        case 2:
        System.out.println("Enter amount to withdraw : ");
        amt = sc.nextFloat();
        ba.withdraw(amt);
        break;
        case 3:
        amt = ba.getBalance();
        System.out.println("Your current balance is Rs."+amt);
        break;
      }
    }while(choice!=4);


  }
}
```

## Input / Output :

```
Enter your name :
abdd
Enter your account number :
786452
Enter your account type (Saving or current) :
saving
Enter your balance
10000
1. Deposite
2. Withdraw
3. Display balance
4. Exit
Enter your choice =>
1
Enter amount to deposite :
65345
1. Deposite
2. Withdraw
3. Display balance
4. Exit
```

```
Enter your choice =>
3
Your current balance is Rs.75345.0
1. Deposite
2. Withdraw
3. Display balance
4. Exit
Enter your choice =>
2
Enter amount to withdraw :
4500
1. Deposite
2. Withdraw
3. Display balance
4. Exit
Enter your choice =>
3
Your current balance is Rs.70845.0
1. Deposite
2. Withdraw
3. Display balance
4. Exit
Enter your choice =>
4
```

**Conclusion :** This practical is about bank management system. In this we can able to check our current balance. Withdraw money and deposite money.

**Practical Name :** 9E

**Aim :** Create a class with following specifications.

Class Emp

| | | |
|---|---|---|
| empId | int | |
| empName | string | |
| designation | string | |
| basic | double | |
| hra | double | readOnly |

Methods

- printDET( )
- kalculateHRA( )
- printDET( ) methods will show details of the EMP.

calculateHRA( ) method will calculate HRA based on basic.

There will 3 designations supported by the application.

If designation is “Manager” - HRA will be 10% of BASIC

if designation is “Officer” - HRA will be 12% of BASIC

if category is “CLERK” - HRA will be 5% of BASIC

Have constructor to which you will pass, empId, designation, basic and price.

And checks whether the BASIC is less than 500 or not. If it is less than 500 raise a custom Exception as given below

Create LowSalException class with proper user message to handle BASIC less than 500.

## Methodology followed :

```
import java.util.*;


class LowSalException extends Exception
```

```
{
   LowSalException(String ex)
   {
      super(ex);
   }

}

class Emp
{
   int empId;
   String empName;
   String designation;
   double basic;
   double hra;

   Emp()
   {
      empId = 0;
      empName = "xyz";
      designation = "xyz";
      basic = 0.00;
   }
   void setDET(int id, String name, String desc, double sal) throws LowSalException
   {
      empId = id;
      empName = name;
```

```
    designation = desc;
    basic = sal;
    if(basic < 500)
    {
        throw new LowSalException("Basic is Low ! ");
    }
}
void printDET()
{
    System.out.print("\nEmployee ID : " +empId);
    System.out.print("\nEmployee Name : " +empName);
    System.out.print("\nEmployee Designation : " +designation);
    System.out.print("\nEmployee Basic : " +basic);
    System.out.print("\nEmployee HRA : " +hra);
}
void calculateHRA()
{
    if(designation.equals("Manager"))
    {
        hra = basic * 0.10;
    }
    else if(designation.equals("officer"))
    {
        hra = basic * 0.12;
    }
    else if(designation.equals("clerk"))
    {
            hra = basic * 0.05;
        }
    }

}

class OopPractical9e
{
    public static void main (String[] args)
    {
        Scanner sc = new Scanner(System.in);
        int id;
        String name, desc;
        double sal;

        System.out.print("Enter employee id : ");
        id = Integer.parseInt(sc.nextLine());

        System.out.print("Enter employee name : ");
        name = sc.nextLine();

        System.out.print("Enter employee designation : ");
        desc = sc.nextLine();

        System.out.print("Enter employee basic : ");
        sal = Double.parseDouble(sc.nextLine());
        Emp obj = new Emp();
        try
        {
           obj.setDET(id,name,desc,sal);
        }
        catch(LowSalException e)
        {
           System.out.print(e);
        }
        obj.calculateHRA();
        obj.printDET();

    }
}
```

**Input / Output :**

```
Enter employee id : 4531
Enter employee name : abc
Enter employee designation : clerk
Enter employee basic : 17000

Employee ID : 4531
Employee Name : abc
Employee Designation : clerk
Employee Basic : 17000.0
Employee HRA : 850.0

Enter employee id : 435
Enter employee name : xyz

Enter employee designation : officer

Enter employee basic : 40000


Employee ID : 435

Employee Name : xyz

Employee Designation : officer

Employee Basic : 40000.0

Employee HRA : 4800.0
```

**Conclusion :** In this practical we calculate the HRA of clerk , officer and manager. And display the list of each employee.

**Practical Name :** 9F

**Aim :** Create a class USERTRAIL with following specifications.

val1, val2 type    int

Method

boolean show( ) will check if val1 and val2 are greater or less than Zero

Have constructor which will val1, val2 and check whether if it is less than 0  then raise a custom Exception (name: Illegal value exception.)

## Methodology followed :

```
import java.util.*;


class IllegalValueException extends Exception

{

IllegalValueException(String ex)
{
super(ex);
}
}

class UserTrail
{
int val1, val2;

UserTrail(int val1, int val2)
{
this.val1 = val1;
this.val2 = val2;
}
boolean show() throws IllegalValueException
{
if(val1>0 && val2>0)
{
return true;
}
else
{
throw new IllegalValueException("Value is less than zero");
//return false;
}
}
}
```

```
class OopPractical9f
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);
boolean b;

System.out.print("Enter Value 1 : ");
int val1 = sc.nextInt();

System.out.print("Enter Value 2 : ");
int val2 = sc.nextInt();

UserTrail obj = new UserTrail(val1, val2);

try
{
b = obj.show();
if(b)
{
System.out.println("Value 1 and Value 2 is greater than zero");
}
}
catch(IllegalValueException ex)
{
System.out.println(ex);
}
}
}
```

## Input / Output :

```
Enter Value 1 : 74
Enter Value 2 : 65
Value 1 and Value 2 is greater than zero

Enter Value 1 : 74
Enter Value 2 : -15
IllegalValueException: Value is less than zero
```

**Conclusion :** In this practical we can find that the entered number is less then zero or greater than zero. If numbers is greater than 0 then it prints greater than zero otherwise it prints illegal value.