16 November 2020

Aayush Shah
19BCE245

Practical 9

OOP Lab

Practical 9 A

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.

CODE

```
package Intpack;

public interface Polygon
{
	void calcArea();
	void calcPeri();
	void display();
}
```

```
package Intpack;

import java.awt.*;
```

```
public class Square implements Polygon
{
	float side;
	float area;
	float perimeter;
	public Square(float s)
	{
		side=s;
		//System.out.println("in side squre const side="+s);
	}

	public void calcArea()
	{
		area=side*side;
		//System.out.println("in side squre area side="+area);
	}

	public void calcPeri()
	{
		perimeter=4*side;
		//System.out.println("in side squre peri ="+perimeter);
	}

	public void display()
	{
		System.out.println("Square");
		System.out.println("side"+side);
		System.out.println("Area="+area);
		System.out.println("Perimeter"+perimeter);
	}
}

package Intpack;

import java.awt.*;

public class Rectangle implements Polygon
{
	float legnth;
	float bredth;
	float area;
	float perimeter;
	public Rectangle(float l,float b)
	{
		legnth=l;
	   bredth=b;
	}

	public void calcArea()
```

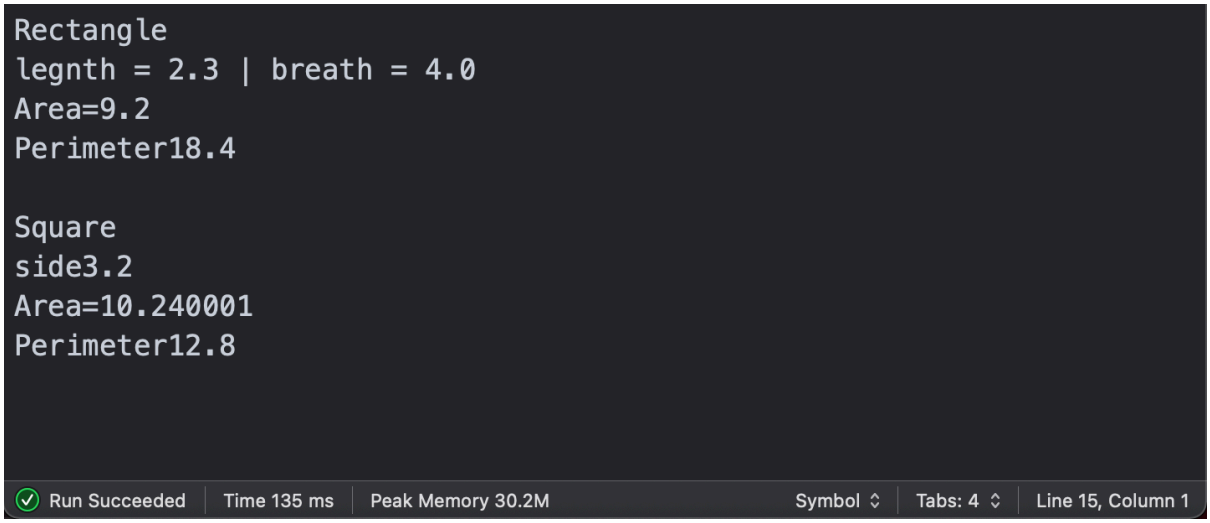
```
	{
		area=legnth*bredth;
	}

	public void calcPeri()
	{
		perimeter=2*legnth*bredth;
	}

	public void display()
	{
		System.out.println("Rectangle");
		System.out.println("legnth = "+legnth+" l breath = "+bredth);
		System.out.println("Area="+area);
		System.out.println("Perimeter"+perimeter);
	}
}

package Intpack;

class Main {
	public static void main(String[] args) {
		Polygon s = new Square(5);
		s.calcArea();
		s.calcPeri();
		s.display();
		Polygon r = new Rectangle(2, 3);
		r.calcArea();
		r.calcPeri();
		r.display();
	}
}

import  Intpack.*;
public class Main2
{
	public static void main(String[] args)
	{
		Square s=new Square(3.2f);
		s.calcArea();
		s.calcPeri();

		Rectangle r=new Rectangle(2.3f,4f);
		r.calcArea();
		r.calcPeri();

		r.display();
		System.out.println();
		s.display();
```

```
    }
}
```

INPUT :

-

OUTPUT :



CONCLUSION :

From the practical 9 A, We learned about the java packages and its various useful properties. Also we revised the concepts of classes and objects. We also applied interface and its applications.

Practical 9 B

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.

CODE

```
import java.util.*;

class Main {

	public static double avgFirst(int N){
		try {
			if(N<1)
			throw new IllegalArgumentException("N must be a Natural
Number.");
			return ((double)(N*(N+1))/(double)4);
		} catch (IllegalArgumentException e) {
			System.out.println("Exception occured : " + e.getMessage());
		}
		return -1;
	}

	public static void main(String[] args) {
		int choice = 0,N;
		Scanner sc = new Scanner(System.in);
		while(choice != 2){
			System.out.println("[1.]Enter new number (press 1) ");
			System.out.println("[2.]Exit             (press 2) ");
			System.out.print("Enter choice : ");
			choice = sc.nextInt();
			switch (choice) {
				case 1:
					System.out.print("Enter natural number N : ");
					N = sc.nextInt();
					if(avgFirst(N)!=-1)
					System.out.println("Average : " + avgFirst(N));
					break;
```

```
                    case 2:
                        System.out.println("FINISHED.");
                        break;
                    default:
                        System.out.println("Invalid choice :(");
                        break;
                }
            }
        }
}
```

INPUT :

```
1
5
1
-2
2
```

OUTPUT :

```
[1.]Enter new number (press 1)
[2.]Exit             (press 2)
Enter choice : 1
Enter natural number N : 5
Average : 7.5
[1.]Enter new number (press 1)
[2.]Exit             (press 2)
Enter choice : 1
Enter natural number N : -2
Exception occured : N must be a Natural Number.
[1.]Enter new number (press 1)
[2.]Exit             (press 2)
Enter choice : 2
FINISHED.
```

Run Succeeded | Time 193 ms | Peak Memory 34.3M | Symbol | Tabs: 4 | 41 Lines, 981 Characters

CONCLUSION :

From the practical 9 B, We learned about exception handling. Here we applied IllegalArgumentException whenever we encounter that the number is not a natural number.

Practical 8 C

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.

CODE :

```
import java.text.*;
import java.util.*;

class Number{
	int firstNumber,secondNumber;
	double result;
	Number(int x, int y){
		this.firstNumber = x;
		this.secondNumber = y;
	}
	double add(){
		this.result = (double)(this.firstNumber + this.secondNumber);
		return result;
	}
	double sub(){
		this.result = Math.abs((double)(this.firstNumber-this.secondNumber));
		return result;
	}
	double mul(){
		this.result = (double)(this.firstNumber*this.secondNumber);
		return result;
	}
	double div(){
		try {
			this.result = (double)(this.firstNumber/this.secondNumber);
```

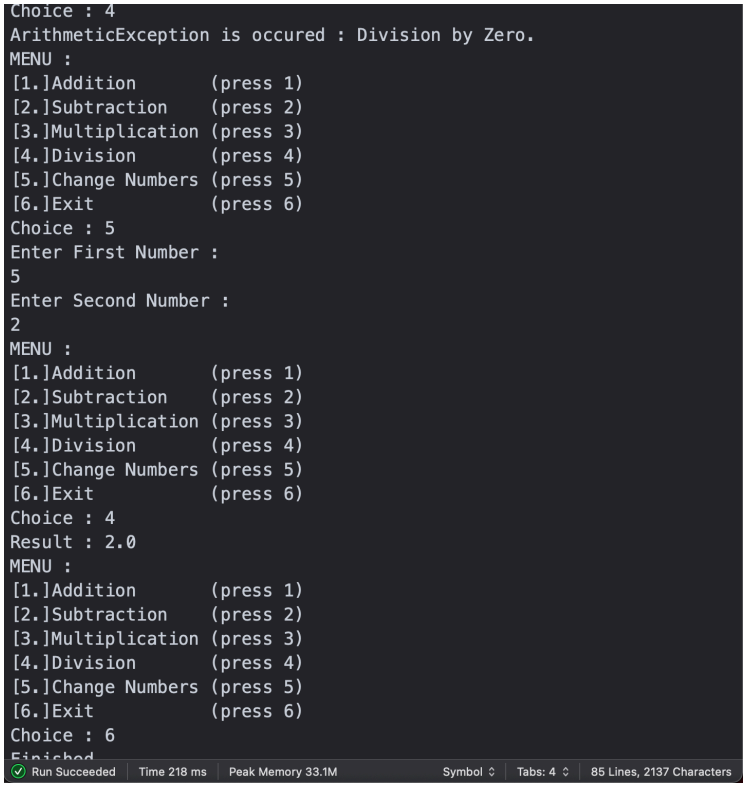
```
                return result;
            } catch (ArithmeticException e) {
                System.out.println("ArithmeticException is occured : Division by
Zero.");
            }
//          this.result = (double)this.firstNumber/(double)this.secondNumber;
            return 0;
        }

}
class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int choice = 0,x,y;
        System.out.print("Enter First Number : ");
        x = sc.nextInt();
        System.out.print("Enter Second Number : ");
        y = sc.nextInt();
        Number n = new Number(x,y);
        while(choice!=6){
            System.out.println("MENU :");
            System.out.println("[1.]Addition       (press 1)");
            System.out.println("[2.]Subtraction    (press 2)");
            System.out.println("[3.]Multiplication (press 3)");
            System.out.println("[4.]Division       (press 4)");
            System.out.println("[5.]Change Numbers (press 5)");
            System.out.println("[6.]Exit           (press 6)");
            System.out.print("Choice : ");
            choice = sc.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Result : " + n.add());
                    break;
                case 2:
                    System.out.println("Result : " + n.sub());
                    break;
                case 3:
                    System.out.println("Result : " + n.mul());
                    break;
                case 4:
                    if(n.div()!=0)
                    System.out.println("Result : " + n.div());
                    break;
                case 5:
                    System.out.println("Enter First Number : ");
                    x = sc.nextInt();
                    System.out.println("Enter Second Number : ");
                    y = sc.nextInt();
                    n = new Number(x,y);
```

```
                        break;
                    case 6:
                        System.out.println("Finished.");
                        break;
                    default:
                        break;
                }
            }

    }
}
```

INPUT :

```
5
0
1
2
3
4
5
5
2
4
6
```

OUTPUT :



```
Enter First Number : 5
Enter Second Number : 0
MENU :
[1.]Addition       (press 1)
[2.]Subtraction    (press 2)
[3.]Multiplication (press 3)
[4.]Division       (press 4)
[5.]Change Numbers (press 5)
[6.]Exit           (press 6)
Choice : 1
Result : 5.0
MENU :
[1.]Addition       (press 1)
[2.]Subtraction    (press 2)
[3.]Multiplication (press 3)
[4.]Division       (press 4)
[5.]Change Numbers (press 5)
[6.]Exit           (press 6)
Choice : 2
Result : 5.0
MENU :
[1.]Addition       (press 1)
[2.]Subtraction    (press 2)
[3.]Multiplication (press 3)
[4.]Division       (press 4)
[5.]Change Numbers (press 5)
[6.]Exit           (press 6)
Choice : 3
Result : 0.0
MENU :
[1.]Addition       (press 1)
```

Run Succeeded | Time 218 ms | Peak Memory 33.1M | Symbol | Tabs: 4 | 85 Lines, 2137 Characters

CONCLUSION :

For the practical 9 C, We again used the concepts of exceptional handling. Here we applied ArithmeticException whenever division by zero is encountered. We defined this exception by try-catch block.

Practical 9 D

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.

CODE :

```
import java.util.*;

class NegativeAmount extends Exception{
	public NegativeAmount(String s) {
		super(s);
	}
}
class InsufficientFunds extends Exception{
	public InsufficientFunds(String s) {
		super(s);
	}
}
class LowBalanceException extends Exception{
	public LowBalanceException(String s) {
		super(s);
	}
}
```

```
class BankAccount{
	Scanner sc = new Scanner(System.in);
	int accNo;
	String custName, accType;
	float balance;

	BankAccount(int accNo, String custName, String accType, float initialBalance){
		this.accNo = accNo;
		this.custName = custName;
		this.accType = accType;
		try {
			if(initialBalance<0)
				throw new NegativeAmount("You entered Negative Amount :(");
		}
		catch (NegativeAmount ex) {
			System.out.println("Exception occured.");
			System.out.println(ex.getMessage());
			while(initialBalance<0){
				System.out.print("Enter initial Balance again : ");
				initialBalance = sc.nextFloat();
			}
		}
		try{
			if(accType.equalsIgnoreCase("Savings") && (initialBalance<1000))
				throw new LowBalanceException("Insufficient balance for
saving account creation.Balance should be atleast 1000 :(");
			if(accType.equalsIgnoreCase("Current") && (initialBalance<5000))
				throw new LowBalanceException("Insufficient balance for
current account creation. Balance should be atleast 5000 :(");
		}
		catch (LowBalanceException ex) {
			System.out.println("Exception occured.");
			System.out.println(ex.getMessage());
			while(true){
				System.out.print("Enter initial Balance again : ");
				initialBalance = sc.nextFloat();
				if((accType.equalsIgnoreCase("Savings") &&
(initialBalance>=1000)) || (accType.equalsIgnoreCase("Current") &&
(initialBalance>=5000)))
				break;
			}
		}
		this.balance = initialBalance;
	}

	void deposit(float amt) {
		try {
			if(amt<0)
```

```
                        throw new NegativeAmount("You entered Negative Amount :(");
                    this.balance += amt;
                    System.out.println("amount of rs. " + amt + " is credited and updated
balance is " + this.balance);
                }
                catch(NegativeAmount ex){
                    System.out.println("Exception occured.");
                    System.out.println(ex.getMessage());
                }
        }
        void withdraw(float amt) {
                try {
                    if(amt<0)
                        throw new NegativeAmount("You entered Negative Amount :(");
                    if(this.accType.equalsIgnoreCase("Savings") && (this.balance-
amt)<1000)
                        throw new InsufficientFunds("Insufficient balance for saving
account withdrawal :(");
                    if(this.accType.equalsIgnoreCase("Current") && (this.balance-
amt)<5000)
                        throw new InsufficientFunds("Insufficient balance for current
account withdrawal :(");
                    this.balance -= amt;
                    System.out.println("amount of rs. " + amt + " is debited and updated
balance is " + this.balance);
                }
                catch(NegativeAmount ex){
                    System.out.println("Exception occured.");
                    System.out.println(ex.getMessage());
                }
                catch(InsufficientFunds ex){
                    System.out.println("Exception occured.");
                    System.out.println(ex.getMessage());
                }
        }
        float getBalance() {
                try {
                    if(this.accType.equalsIgnoreCase("Savings") &&
(this.balance)<1000)
                        throw new InsufficientFunds("Insufficient balance for saving
account :(");
                    if(this.accType.equalsIgnoreCase("Current") && (this.balance)<5000)
                        throw new InsufficientFunds("Insufficient balance for current
account :(");
                    return this.balance;
                }
                catch(InsufficientFunds ex){
                    System.out.println("Exception occured.");
                    System.out.println(ex.getMessage());
```

```
            }
            return -1;
        }
}
public class Main {

        public static void main(String[] args) {

            int accNo,choice = 0;
            String custName, accType;
            float balance,amt;

            Scanner sc = new Scanner(System.in);
            System.out.println("Create Account : ");

            System.out.print("\tEnter account number : ");
            accNo = sc.nextInt();
            System.out.print("\tEnter customer name : ");
            custName = sc.next();
            System.out.print("\tEnter account type (Current/Savings) : ");
            accType = sc.next();
            System.out.print("\tEnter initial balance : ");
            balance = sc.nextFloat();
            BankAccount B = new BankAccount(accNo, custName, accType,
balance);

            while (choice!=4) {
                System.out.println("Menu : ");
                System.out.println("[1.]Deposit amount (press 1)");
                System.out.println("[2.]Withdraw amount (press 2)");
                System.out.println("[3.]Get balance (press 3)");
                System.out.println("[4.]Exit (press 4)");
                System.out.print("Enter choice : ");
                choice = sc.nextInt();
                switch (choice) {
                    case 1:
                        System.out.print("Enter amount : ");
                        amt = sc.nextFloat();
                        B.deposit(amt);
                        break;
                    case 2:
                        System.out.print("Enter amount : ");
                        amt = sc.nextFloat();
                        B.withdraw(amt);
                        break;
                    case 3:
                        amt = B.getBalance();
                        if(amt != -1)
                        System.out.println("Your current balance is " + amt);
```

```
                        break;
                    case 4:
                        System.out.print("*** THANK YOU ***");
                        break;
                    default:
                        System.out.println("Invalid Choice :(");
                        break;
                }
            }


    }

}
```

INPUT :

```
1234
Aayush
Savings
10
1000
1
100
2
5000
2
100
3
4
```

OUTPUT :

```
Create Account :
    Enter account number : 1234
    Enter customer name : Aayush
    Enter account type (Current/Savings) : Savings
    Enter initial balance : 10
Exception occured.
Insufficient balance for saving account creation.Balance should be
    atleast 1000 :(
Enter initial Balance again : 1000
Menu :
[1.]Deposit amount (press 1)
[2.]Withdraw amount (press 2)
[3.]Get balance (press 3)
[4.]Exit (press 4)
Enter choice : 1
Enter amount : 100
amount of rs. 100.0 is credited and updated balance is 1100.0
Menu :
[1.]Deposit amount (press 1)
[2.]Withdraw amount (press 2)
[3.]Get balance (press 3)
[4.]Exit (press 4)
Enter choice : 2
Enter amount : 5000
Exception occured.
Insufficient balance for saving account withdrawal :(
Menu :
[1.]Deposit amount (press 1)
[2.]Withdraw amount (press 2)
[3.]Get balance (press 3)
[4.]Exit (press 4)
Run Succeeded | Time 281 ms | Peak Memory 36.9M | main | Tabs: 4 | Line 160, Column 9
```

```
Menu :
[1.]Deposit amount (press 1)
[2.]Withdraw amount (press 2)
[3.]Get balance (press 3)
[4.]Exit (press 4)
Enter choice : 2
Enter amount : 100
amount of rs. 100.0 is debited and updated balance is 1000.0
Menu :
[1.]Deposit amount (press 1)
[2.]Withdraw amount (press 2)
[3.]Get balance (press 3)
[4.]Exit (press 4)
Enter choice : 3
Your current balance is 1000.0
Menu :
[1.]Deposit amount (press 1)
[2.]Withdraw amount (press 2)
[3.]Get balance (press 3)
[4.]Exit (press 4)
Enter choice : 4
*** THANK YOU ***
Run Succeeded | Time 281 ms | Peak Memory 36.9M | main | Tabs: 4 | Line 160, Column 9
```

CONCLUSION :

For the practical 9 D, We used the various concepts of java like exception handling , objects and classes etc. Here we used custom exceptional handling concept. We defined three custom exceptions :

- NegativeAmount
- InsufficientFunds
- LowBalanceException

We applied them whenever corresponding situation occurred.

Practical 9 E

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.

CODE :

```
import java.util.Scanner;

class LowSalException extends Exception{
	public LowSalException(String s){
		super(s);
	}
}

class Emp{
	int empId;
	String empName,designation;
	double basic;
	private double hra;
	Emp(int empId, String empName, String designation, double basic){
		this.empId = empId;
		this.empName = empName;
		this.designation = designation;
		this.basic = basic;
		if(this.basic<500) {
```

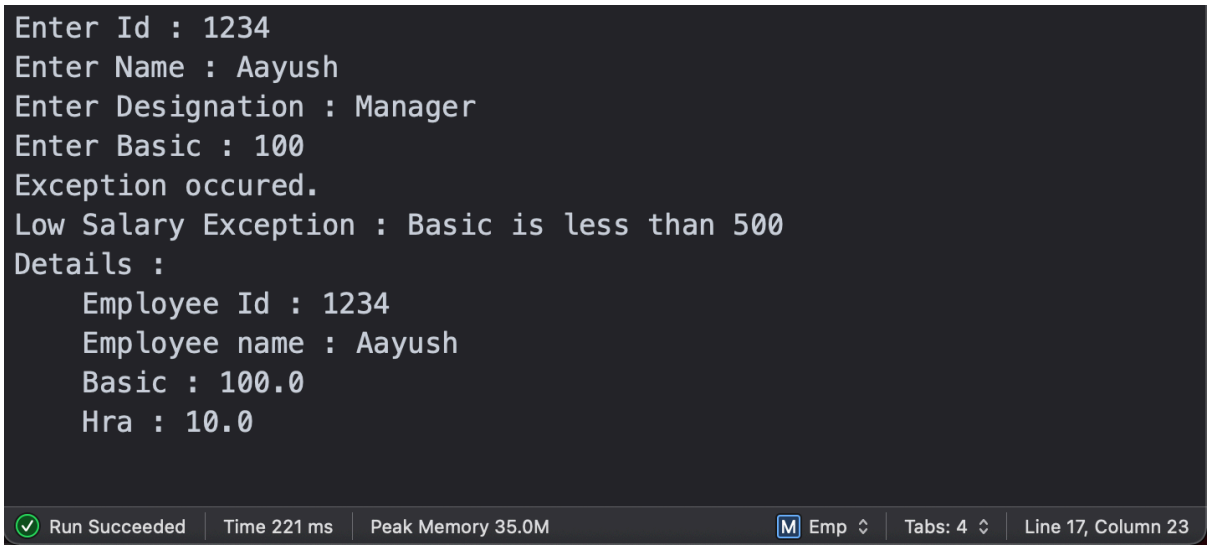
```
                try {
                        throw new LowSalException("Low Salary Exception : Basic is
less than 500");
                }
                catch(LowSalException ex){
                        System.out.println("Exception occured.");
                        System.out.println(ex.getMessage());
                }
        }

        setHra(calculateHRA());
    }
    void printDET(){
        System.out.println("Details : ");
        System.out.println("\tEmployee Id : " + this.empId);
        System.out.println("\tEmployee name : " + this.empName);
        System.out.println("\tBasic : " + this.basic);
        System.out.println("\tHra : " + this.hra);
    }
    double calculateHRA() {
        if(this.designation.equals("Manager"))
            return ((0.10)*(this.basic));
        else if(this.designation.equals("Officer"))
            return ((0.12)*(this.basic));
        else if(this.designation.equals("CLERK"))
            return ((0.05)*(this.basic));
        else
            return 0;
    }
    void setHra(double hra) {
        this.hra = hra;
    }
}

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Id : ");
        int id = sc.nextInt();
        System.out.print("Enter Name : ");
        String name = sc.next();
        System.out.print("Enter Designation : ");
        String designation = sc.next();
        System.out.print("Enter Basic : ");
        double basic = sc.nextDouble();

        Emp e = new Emp(id, name, designation, basic);
        e.printDET();
```

```
    }
}
```

INPUT :

1234
Aayush
Manager
100

OUTPUT :



CONCLUSION :

For the practical 8 E, We made use of the concept Exceptional Handling. We defined custom exception named LowSalException and applied it whenever basic is less then 500 and also calculated different HRA according to employee's designation.

Practical 9 F

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.)

CODE :

```
import java.util.Scanner;

class MyException extends Exception{
	int val1,val2;
	public MyException(String s){
		super(s);
	}
}

public class USERTRAIL {
	int val1,val2;

	USERTRAIL(int val1, int val2){
		this.val1 = val1;
		this.val2 = val2;
		if(val1<0 || val2<0) {
			try {
				throw new MyException("Illegal value exception.");
			}
			catch(MyException ex){
				System.out.print("Exception occured : ");
				System.out.println(ex.getMessage());
			}
		}
		else {
			System.out.println("Exception not occured.");
		}

	}

	boolean show() {
		if(val1!=0 && val2!=0) {
			System.out.println("val1(" + val1 + ") and val2(" + val2 + ") are
greater than or less than zero.");
```

```
                return true;
            }
            else {
                System.out.println("val1("+ val1 + ") or val2(" + val2 + ") or both
equal to zero.");
                return false;
            }
        }

        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            int value1,value2;
            System.out.print("Enter value 1 : ");
            value1 = sc.nextInt();
            System.out.print("Enter value 2 : ");
            value2 = sc.nextInt();
            USERTRAIL u = new USERTRAIL(value1,value2);
            if(u.show())
                System.out.println("show method returned : True.");
            else
                System.out.println("show method returned : False.");
        }

}
```

Some Sample INPUTs & OUTPUTs :

```
Enter value 1 : 1
Enter value 2 : 2
Exception not occured.
val1(1) and val2(2) are greater than or less than zero.
show method returned : True.
```

Run Succeeded | Time 199 ms | Peak Memory 32.8M | Symbol | Tabs: 4 | 57 Lines, 1277 Characters

```
Enter value 1 : 1
Enter value 2 : 0
Exception not occured.
val1(1) or val2(0) or both equal to zero.
show method returned : False.
```

Run Succeeded | Time 198 ms | Peak Memory 32.8M | Symbol | Tabs: 4 | 57 Lines, 1277 Characters

```
Enter value 1 : 1
Enter value 2 : -1
Exception occured : Illegal value exception.
val1(1) and val2(-1) are greater than or less than zero.
show method returned : True.
```

Run Succeeded | Time 185 ms | Peak Memory 32.6M | Symbol | Tabs: 4 | 57 Lines, 1277 Characters

CONCLUSION :

For the practical 9 E, We made use of the concept Exceptional Handling. Here we defined the custom exception name MyException which shows Illegal value Exception whenever we entered negative number.