LAB RECORD

23CSE111 – Object Oriented Programming

Submitted by

CH.SC.U4CSE24108 – **Chepoori Sai Vivek**

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING, CHENNAI

BONAFIDE CERTIFICATE

This is to certify that the Lab Record work for 23CSE111- Object Oriented Programming Subject submitted by ***CH.SC.U4CSE24108 –Chepoori Sai Vivek*** in **"Computer Science and Engineering"** is a bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Chennai.

This Lab examination held in 2nd Semester

Internal Examiner 1 Internal Examiner 2

Index

S.NO	TITLE	PAGE.NO
	UML DIAGRAM	7
1.	**TITLE OF UML DIAGRAM -1**	7
	1.a) ATM Management System Use Case Diagram	7
	1.b) ATM Management System Class Diagram	7
	1.c) ATM Management System Sequence Diagram	8
	1.d) ATM Management System State Diagram	8
	1.e) ATM Management System Collaboration Diagram	9
2.	**TITLE OF UML DIAGRAM -2**	10
	2.a) Bank Management System Use Case Diagram	10
	2.b) Bank Management System Class Diagram	10
	2.c) Bank Management System Sequence Diagram	11
	2.d) Bank Management System State Diagram	11
	2.e) Bank Management System Collaboration Diagram	12
3.	**BASIC JAVA PROGRAMS**	13
	3.a) BMI Calculator program in java	13
	3.b) Reverse multiplication program in java	14
	3.c) Sum of four-digited program in java	15
	3.d) Java Program to check if two numbers are equal	16
	3.e) Java Program to reverse a number	17
	3.f) Java Program to find sum of first n natural numbers	18
	3.g) Java Program to find whether number is positive or negative	19
	3.h) Java Program to find the largest among three numbers	21
	3.i) Java Program to find the largest element in an array. Program	23

	3.j) Java Program to print Pascal's Triangle	25
	INHERITANCE	27
4.	**SINGLE INHERITANCE PROGRAMS**	27
	4.a) Write a Java program using inheritance to calculate the area of a rectangle based on user input.	27
	4.b) Write a Java program using inheritance to display a restaurant menu and generate a bill based on user-selected items.	28
5.	**MULTILEVEL INHERITANCE PROGRAMS**	30
	5.a) Write a Java program demonstrating multilevel inheritance with different types of pens and their behaviors.	30
	5.b) Write a Java program using inheritance to display a categorized restaurant menu including starters, main items, desserts, and drinks.	32
6.	**HIERARCHICAL INHERITANCE PROGRAMS**	35
	6.a) Write a Java program using inheritance to model vehicles and calculate their efficiency based on speed and distance.	35
	6.b)Write a Java program using inheritance to calculate the area and perimeter of different shapes like Circle and Square.	38
7.	**HYBRID INHERITANCE PROGRAMS**	41
	7.a) Write a Java program using hybrid inheritance to model different bank accounts like savings, current, and loan.	41
	7.b) Write a Java program using multilevel and hierarchical inheritance to manage a ticketing system for different types of transport like trains and buses.	43
	POLYMORPHISM	46
8.	**CONSTRUCTOR PROGRAMS**	46
	8.a) Demonstrate constructor chaining and encapsulation with `Shape` as base class and `Circle`, `Rectangle` as derived classes showing area and color.	46
9.	**CONSTRUCTOR OVERLOADING PROGRAMS**	48
	9.a) How can constructor overloading be used to handle different types of payment details in a class?	48
10.	**METHOD OVERLOADING PROGRAMS**	49
	10.a) How does method overloading help calculate total price with varying conditions in a shop system?	49
	10.b) How is method overloading implemented to perform multiplication of different numbers of integers?	50
11.	**METHOD OVERRIDING PROGRAMS**	51
	11.a) How is method overriding used in different subclasses to	51

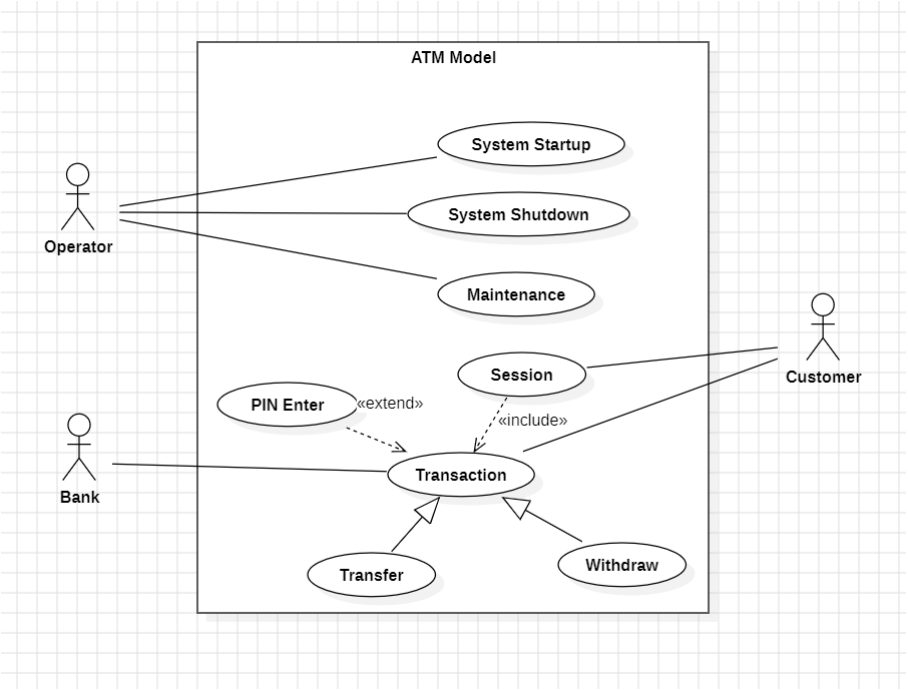
	customize the behavior of a base class method?	
	11.b) How does method overriding help in calculating different delivery item prices using inheritance?	52
	ABSTRACTION	53
12.	**INTERFACE PROGRAMS**	53
	12.a) How does Java use interfaces to enable polymorphic behavior in flying vehicles?	53
	12.b) How is the `Resizable` interface implemented to modify the dimensions of a rectangle in Java?	54
	12.c) How is the `Playable` interface used to demonstrate polymorphism in different sports games in Java?	56
	12.d) How is the concept of interface implementation demonstrated through the `Playable` interface in Java?	57
13.	**ABSTRACT CLASS PROGRAMS**	57
	13.a) How does the `BankAccount` abstract class enforce method implementation in its subclasses?	57
	13.b) How does the `Animal` abstract class enforce method overriding in its subclasses?	59
	13.c) How does the code demonstrate abstraction and method overriding using `Animal` and `RapidoBooking` classes?	60
	13.d) How does the code implement abstraction and polymorphism in calculating area and perimeter of different shapes?	62
	ENCAPSULATION	64
14.	**ENCAPSULATION PROGRAMS**	64
	14.a) How is encapsulation implemented using getter and setter methods in this Java program?	64
	14.b) How does the Rectangle class use encapsulation to manage user input for dimensions?	65
	14.c)How does the `House` class calculate the total price based on its area and price per square meter?	66
	14.d)How does the `CricketPlayer` class store and display information about a player's name, runs, and wickets?	67
	14.e)Create a `BankAccount` class that hides balance and allows secure deposit and withdrawal operations.	69
	14.f)Create a `Student` class that encapsulates marks with validation between 0 and 100.	69
	14.g)Design a `Car` class that controls speed with a max limit using encapsulation.	70
	14.h)Implement a `Patient` class that securely manages age and temperature with range checks.	71
15.	**PACKAGES PROGRAMS**	72
	15.a)How does the program use an `ArrayList` to store and print a list of names?	72
	15.b)How does the program create a TCP client to send a message to a server and receive a response?	73

	15.c)How does the program demonstrate working with current date, time, formatting, time zones, and calculating date differences using the Java `time` API?	74
	15.d)How does this program take user input for name and age and display a personalized greeting?	75
16.	**EXCEPTION HANDLING PROGRAMS**	75
	16.a)How does this program use a custom exception to handle even number inputs?	75
	16.b)How does this program use a custom exception to detect negative numbers from a file?	76
	16.c)How does this program handle odd numbers using a user-defined exception in Java?	77
	16.d)How does this Java program use a custom exception to validate if a given string is a palindrome?	78
17.	**FILE HANDLING PROGRAMS**	79
	17.a)How does this Java program create and write content to a file using `FileWriter` with exception handling?	79
	17.b)How does this Java program read and display the contents of a file using `Scanner` and handle file-not-found exceptions?	79
	17.c)How does this Java program count the total number of words in a file using `Scanner` and string splitting?	80
	17.d)**4.** How does this Java program copy the contents of one file to another using FileReader and FileWriter?	81

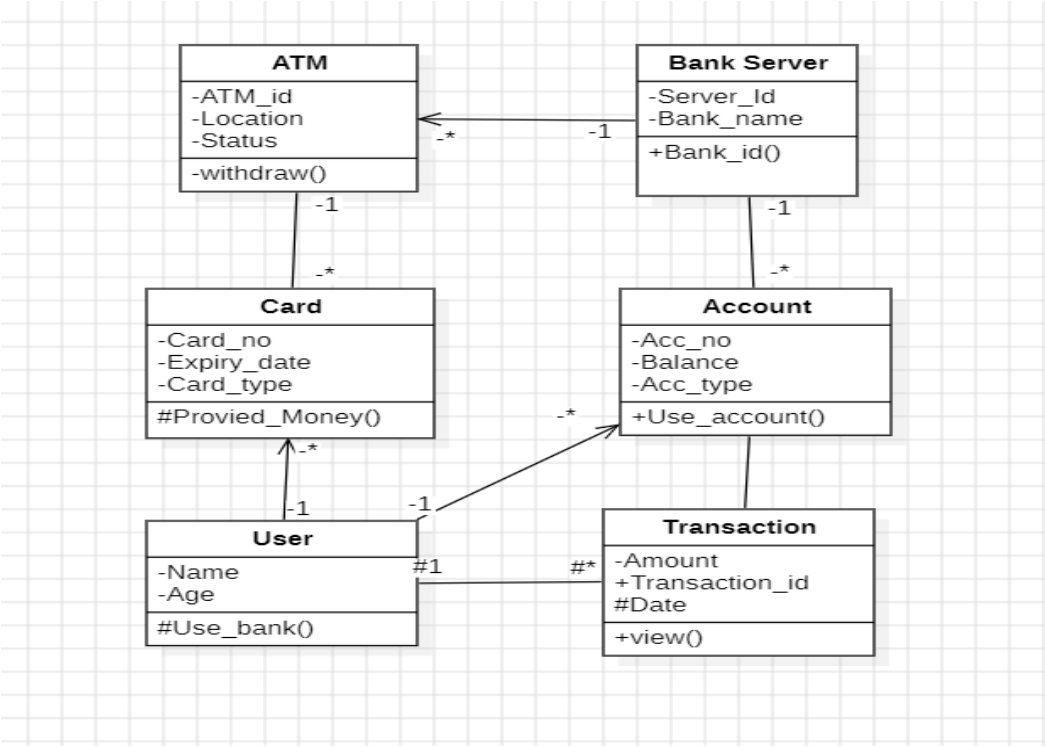
1. UML Diagrams

Problem 1 – ATM Management System

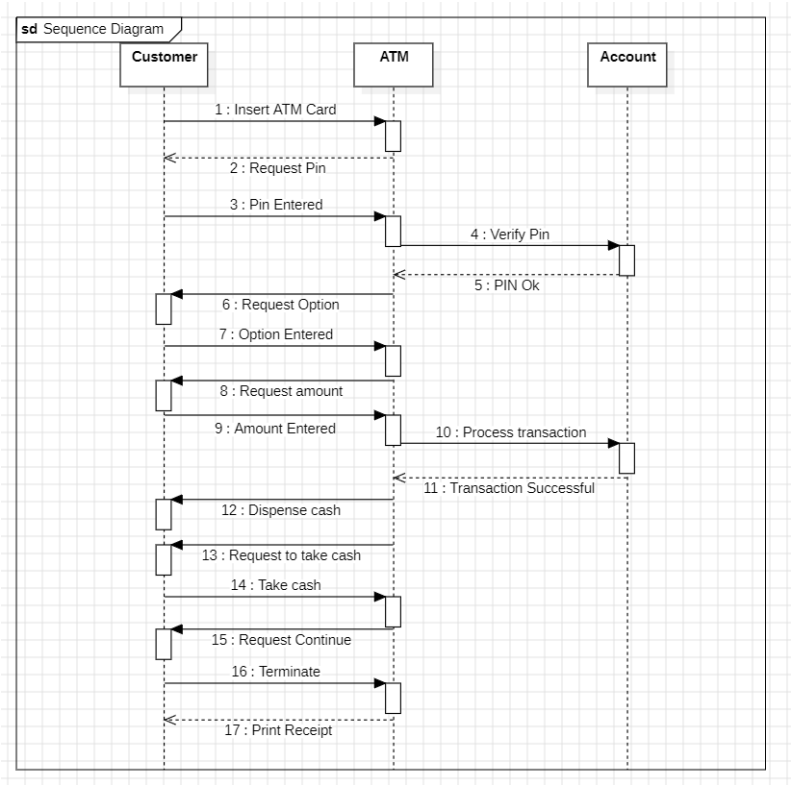
a. Use Case Diagram



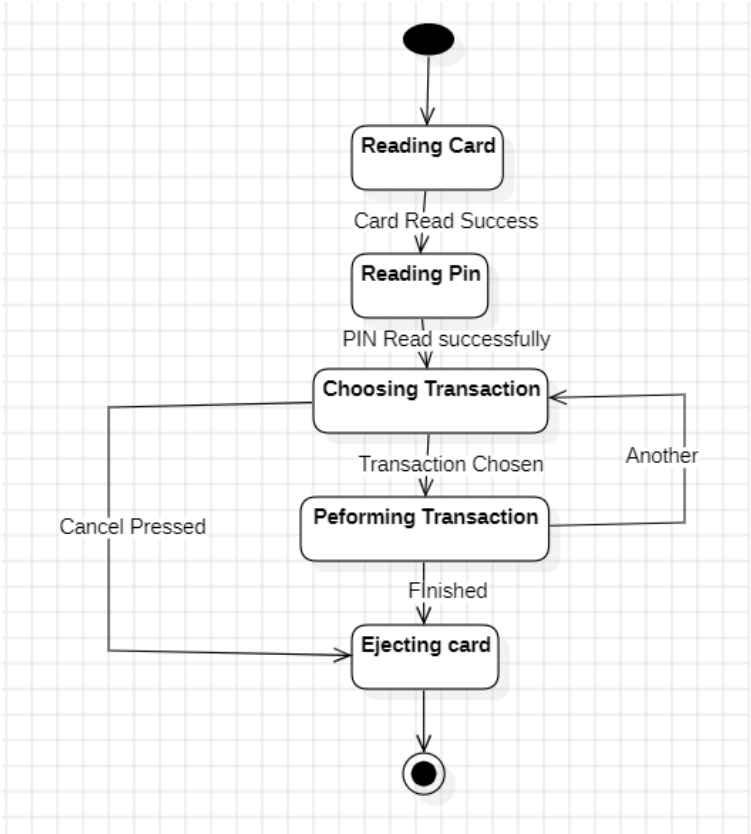
b. Class Diagram



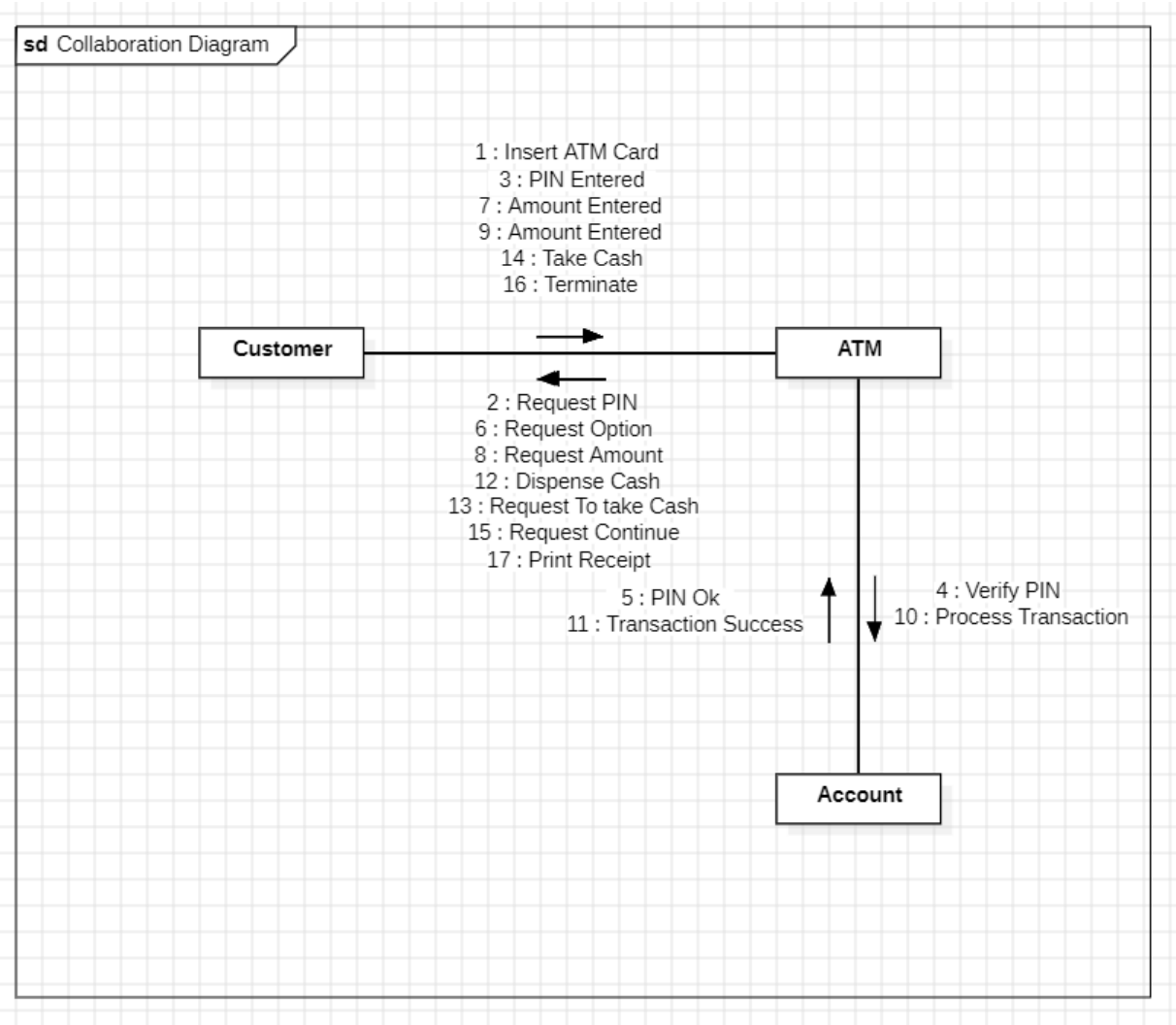
c. Sequence Diagram



d. State Diagram

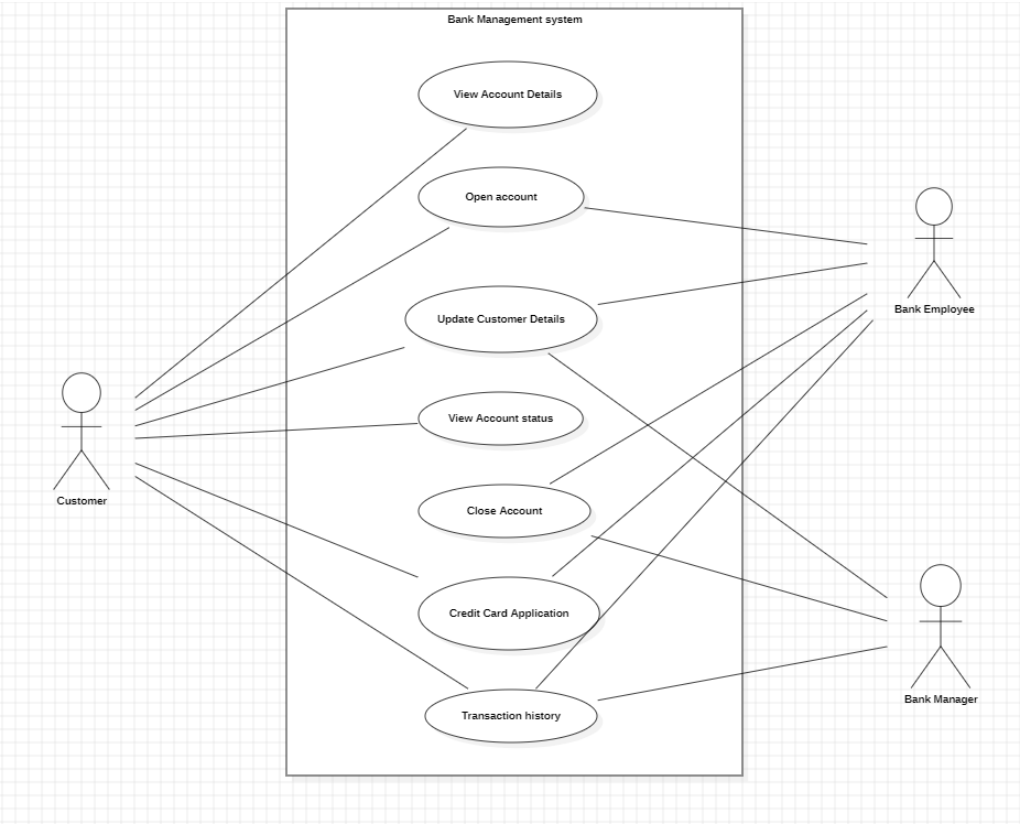


e. Collaboration Diagram

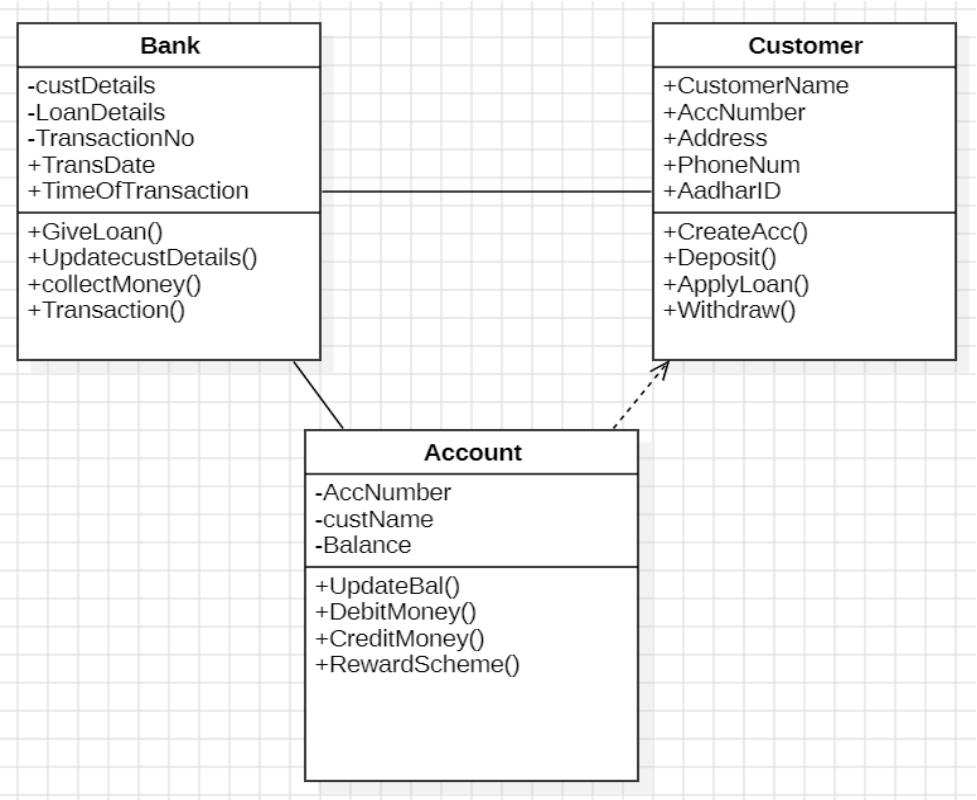


Problem 2 – Bank Management System

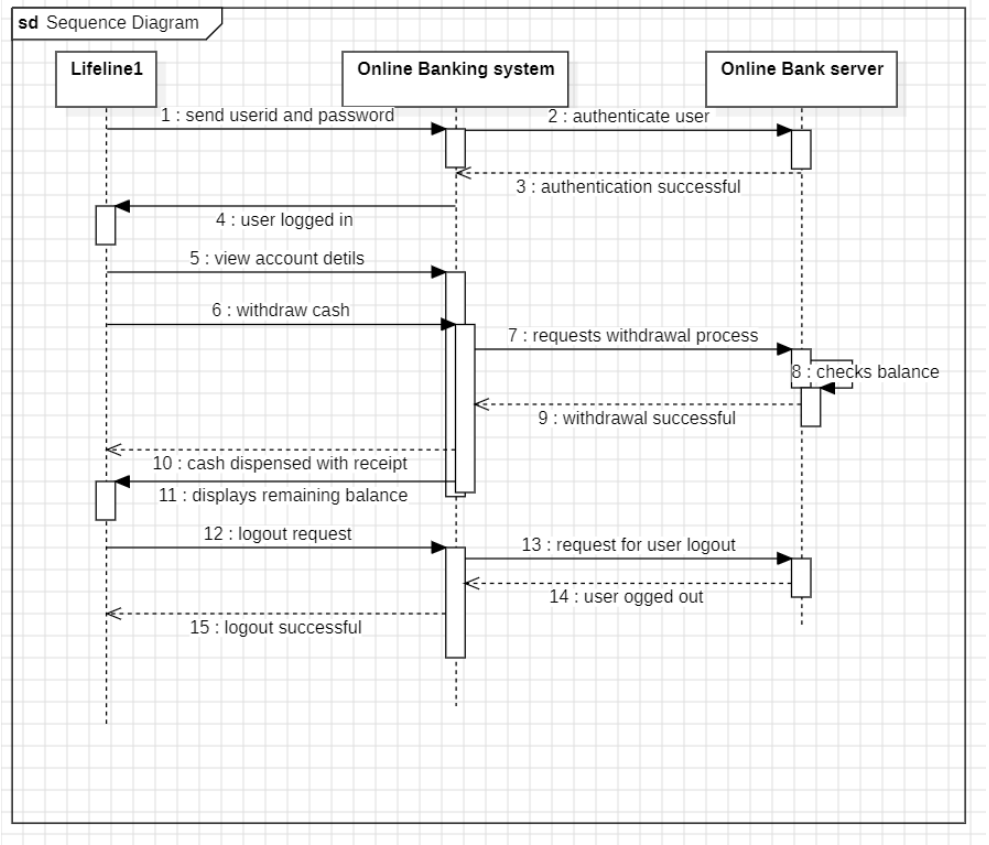
a. Use Case Diagram



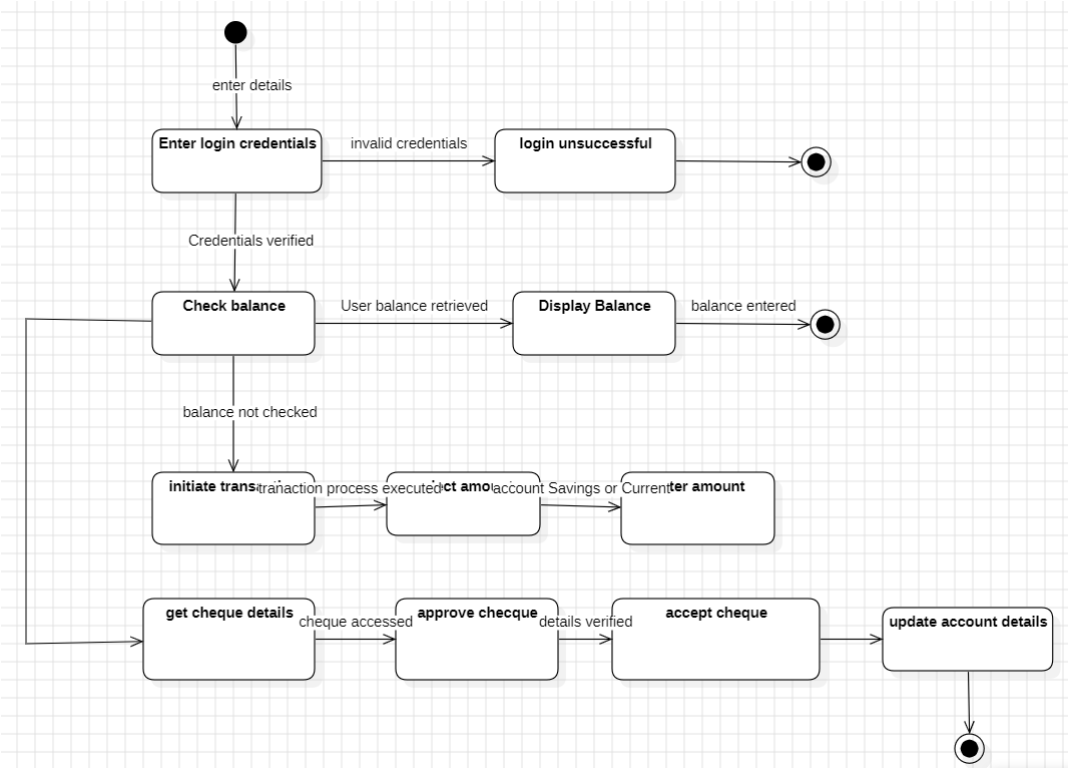
b. Class Diagram



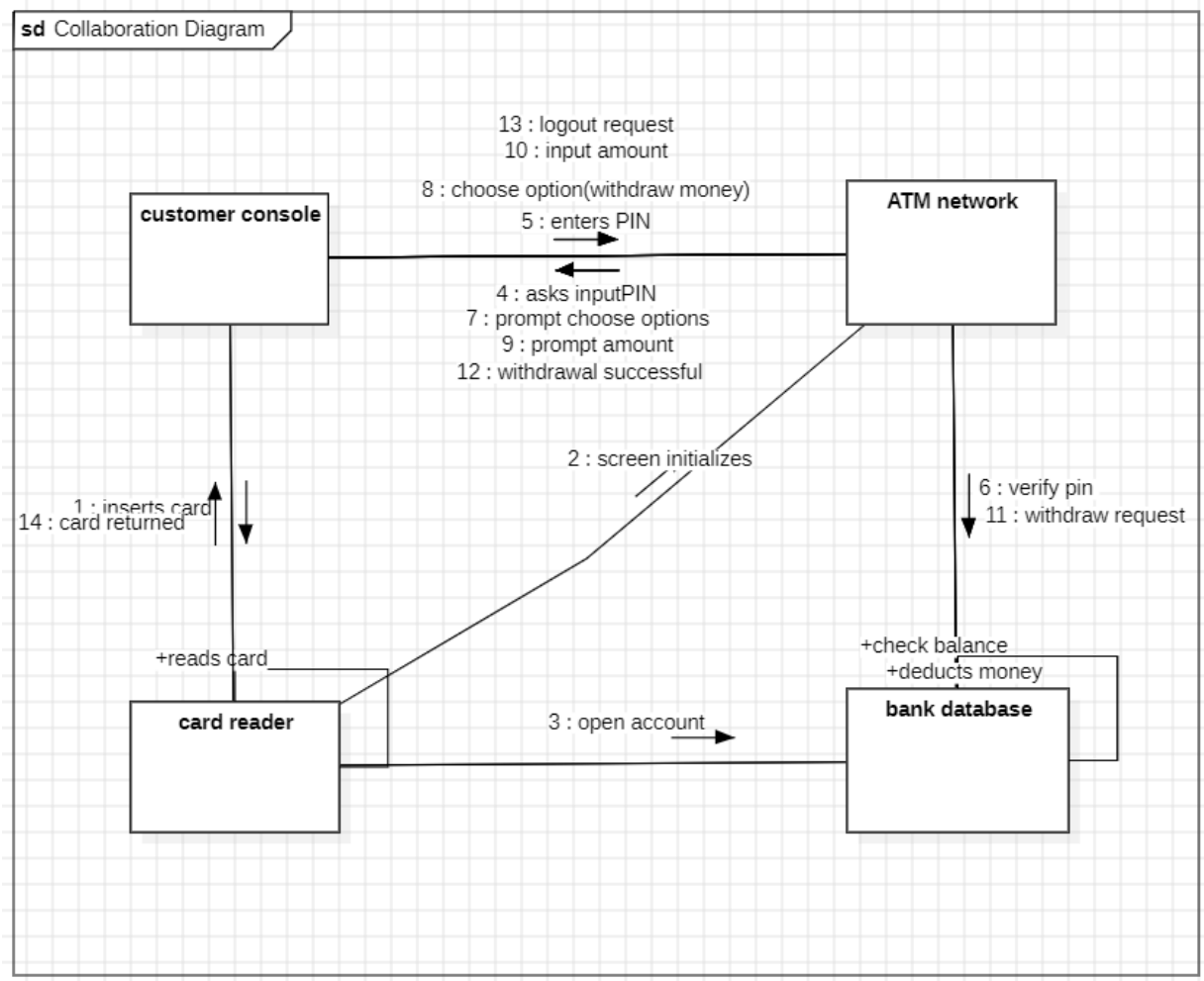
c. Sequence Diagram



d. State Diagram



e. Collaboration Diagram



Java Basic Programs

1. BMI Calculator program in java

Program:

```
import java.util.Scanner;
public class BMICalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter weight in kilograms: ");
        double weight = scanner.nextDouble();
        System.out.print("Enter height in meters: ");
        double height = scanner.nextDouble();
        double bmi = weight / (height * height);
        System.out.printf("Your BMI is: %.2f\n", bmi);
        if (bmi < 18.5) {
            System.out.println("Category: Underweight");
        } else if (bmi < 24.9) {
            System.out.println("Category: Normal weight");
        } else if (bmi < 29.9) {
            System.out.println("Category: Overweight");
        } else {
            System.out.println("Category: Obese");
        }
        scanner.close();
    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac BMICalculator.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java BMICalculator
Enter weight in kilograms: 60
Enter height in meters: 2
Your BMI is: 15.00
Category: Underweight
```

2. Reverse multiplication program in java

Program:

```
import java.util.Scanner;

public class ReverseMultiplicationTable {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number: ");

        int num = scanner.nextInt();

        System.out.print("Enter the range: ");

        int range = scanner.nextInt();

        System.out.println("Multiplication Table of " + num + " in Reverse Order:");

        for (int i = range; i >= 1; i--) {

            System.out.println(num + " x " + i + " = " + (num * i));

        }

        scanner.close();

    }

}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac ReverseMultiplicationTable.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java ReverseMultiplicationTable
Enter a number: 10
Enter the range: 10
Multiplication Table of 10 in Reverse Order:
10 x 10 = 100
10 x 9 = 90
10 x 8 = 80
10 x 7 = 70
10 x 6 = 60
10 x 5 = 50
10 x 4 = 40
10 x 3 = 30
10 x 2 = 20
10 x 1 = 10
```

3. Sum of four-digited program in java

Program:

```
import java.util.Scanner;

public class SumOfDigits {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a four-digit number: ");

        int number = scanner.nextInt();

        if (number < 1000 || number > 9999) {

            System.out.println("Please enter a valid four-digit number.");

        } else {

            int sum = 0;

            int temp = number;

            while (temp > 0) {

                sum += temp % 10;

                temp /= 10;

            }

            System.out.println("Sum of digits: " + sum);

        }

        scanner.close();

    }

}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac SumOfDigits.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java SumOfDigits
Enter a four-digit number: 1845
Sum of digits: 18
```

4. Java Program to check if two numbers are equal.

Program:

```
import java.util.Scanner;
public class Equal_Integer
{
    public static void main(String[] args)
    {
        int m, n;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the first number:");
        m = s.nextInt();
        System.out.print("Enter the second number:");
        n = s.nextInt();
        if(m == n)
        {
            System.out.println(m+" and "+n+" are equal ");
        }
        else
        {
            System.out.println(m+" and "+n+" are not equal ");
        }
    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Equal_Integer.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Equal_Integer
Enter the first number: 18
Enter the second number: 18
18 and 18 are equal
```

5. Java Program to reverse a number.

Program:

```
import java.util.Scanner;

public class Reverse_Number

{

    public static void main(String args[])

    {

        int m, n, sum = 0;

        Scanner s = new Scanner(System.in);

        System.out.print("Enter the number:");

        m = s.nextInt();

        while(m > 0)

        {

            n = m % 10;

            sum = sum * 10 + n;

            m = m / 10;

        }

        System.out.println("Reverse of a Number is "+sum);

    }

}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Reverse_Number.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Reverse_Number
Enter the number: 1845
Reverse of a Number is 5481
```

6. Java Program to find sum of first n natural numbers.

Program:

```
import java.util.Scanner;

public class Sum_Numbers

{

    int sum = 0, j = 0;

    public static void main(String[] args)

    {

        int n;

        Scanner s = new Scanner(System.in);

        System.out.print("Enter the no. of elements you want:");

        n = s.nextInt();

        int a[] = new int[n];

        System.out.print("Enter all the elements you want:");

        for(int i = 0; i < n; i++)

        {

            a[i] = s.nextInt();

        }

        Sum_Numbers obj = new Sum_Numbers();

        int x = obj.add(a, a.length, 0);

        System.out.println("Sum:"+x);
```

```
    }
    int add(int a[], int n, int i)
    {
        if(i < n)
        {
            return a[i] + add(a, n, ++i);
        }
        else
        {
            return 0;
        }
    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Sum_Numbers.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Sum_Numbers
Enter the number of elements: 3
Enter all the elements: 1
7
5
Sum: 13
```

7. Java Program to find whether number is positive or negative.

Program:

```
import java.util.Scanner;
public class Postive_Negative
{
    public static void main(String[] args)
    {
```

```
        int n;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter the number you want to check:");
        n = s.nextInt();
        if(n > 0)
        {
            System.out.println("The given number "+n+" is Positive");
        }
        else if(n < 0)
        {
            System.out.println("The given number "+n+" is Negative");
        }
        else
        {
            System.out.println("The given number "+n+" is neither Positive nor Negative ");
        }
    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Positive_Negative.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Positive_Negative
Enter a number: 2
2 is Positive
```

8. Java Program to find the largest among three numbers.

Program:

```
import java.util.Scanner;

public class Biggest_Number

{

    public static void main(String[] args)

    {

        int x, y, z;

        Scanner s = new Scanner(System.in);

        System.out.print("Enter the first number:");
```

```
        x = s.nextInt();
        System.out.print("Enter the second number:");
        y = s.nextInt();
        System.out.print("Enter the third number:");
        z = s.nextInt();
        if(x > y && x > z)
        {
            System.out.println("Largest number is:"+x);
        }
        else if(y > z)
        {
            System.out.println("Largest number is:"+y);
        }
        else
        {
            System.out.println("Largest number is:"+z);
        }

    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Biggest_Number.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Biggest_Number
Enter three numbers: 7
10
18
Largest number is: 18
```

9. Java Program to find the largest element in an array.

Program:

```
import java.util.Scanner;
public class Largest_Number
{
    public static void main(String[] args)
    {
        int n, max;
        Scanner s = new Scanner(System.in);
        System.out.print("Enter number of elements in the array:");
        n = s.nextInt();
        int a[] = new int[n];
        System.out.println("Enter elements of array:");
        for(int i = 0; i < n; i++)
        {
            a[i] = s.nextInt();
        }
        max = a[0];
        for(int i = 0; i < n; i++)
        {
            if(max < a[i])
            {
                max = a[i];
            }
        }
        System.out.println("Maximum value:"+max);
    }
}
```

Output:

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Largest_Number.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Largest_Number
Enter number of elements: 3
Enter elements:
7
10
18
Maximum value: 18
```

10. Java Program to print Pascal's Triangle.

Program:

```
public class PascalTriangle
{
    public static void main(String[] args)
    {
        int rows = 5;
        for (int i = 0; i < rows; i++)
        {
            int number = 1;
            for (int j = 0; j < rows - i; j++)
            {
                System.out.print(" ");
            }
            for (int j = 0; j <= i; j++)
            {
                System.out.print(number + " ");
                number = number * (i - j) / (j + 1);
            }
            System.out.println();
        }
```

```
    }
  }
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac PascalTriangle.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java PascalTriangle
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
```

Inheritance

Single Inheritance problems

1. Write a Java program using inheritance to calculate the area of a rectangle based on user input.

Code:

```
import java.util.Scanner;

class Shape {
    public void calculateArea() {
        System.out.println("The area has been calculating");
    }
}

class Rectangle extends Shape {
    int length, breadth;

    Rectangle(int length, int breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    public void getArea() {
        System.out.println("The area of rectangle is :" + (length
* breadth));
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner obj = new Scanner(System.in);
        System.out.println("Enter length :");
        int length = obj.nextInt();
        System.out.println("Enter breadth :");
        int breadth = obj.nextInt();
```

```
        Shape myShape = new Shape();
        myShape.calculateArea();

        Rectangle myObj = new Rectangle(length, breadth);
        myObj.getArea();
    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Main.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Main
Enter length :
12
Enter breadth :
23
The area has been calculating
The area of rectangle is :276
```

2. Write a Java program using inheritance to display a restaurant menu and generate a bill based on user-selected items.

Code:

```
import java.util.Scanner;

class RestrauntMenu {
    public void defaultMenu() {
        System.out.println("-----------####----------");
        System.out.println("Here is our menu card");
        System.out.println("Biryani");
        System.out.println("Kadhai Paneer");
        System.out.println("Butter Chicken");
        System.out.println("Nans");
        System.out.println("Sweets");
        System.out.println("Juices");
        System.out.println("Ice Creams");
        System.out.println("-------------------------");
    }
}

class PaymentBill extends RestrauntMenu {
```

```
    String Biryani, curry, icecreams;

    PaymentBill(String Biryani, String curry, String icecreams)
{
        this.Biryani = Biryani;
        this.curry = curry;
        this.icecreams = icecreams;
    }

    public void yourBill() {
        System.out.println("-----------####----------");
        System.out.println("Here is our bill card");
        System.out.println(Biryani);
        System.out.println(curry);
        System.out.println(icecreams);
        System.out.println("--------------------------");
    }
}

public class Restraunt {
    public static void main(String[] args) {
        Scanner obj = new Scanner(System.in);
        System.out.println("Enter any Biryani:");
        String Biryani = obj.nextLine();
        System.out.println("Enter any Curry:");
        String curry = obj.nextLine();
        System.out.println("Enter any Ice Creams:");
        String icecreams = obj.nextLine();

        RestrauntMenu myRes = new RestrauntMenu();
        myRes.defaultMenu();

        PaymentBill myobj = new PaymentBill(Biryani, curry,
icecreams);
        myobj.yourBill();
    }
```

}

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Restraunt.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Restraunt
Enter any Biryani:
Prawns
Enter any Curry:
chicken
Enter any Ice Creams:
vanilla
-----------####----------
Here is our menu card
Biryani
Kadhai Paneer
Butter Chicken
Nans
Sweets
Juices
Ice Creams
--------------------------
-----------####----------
Here is our bill card
Prawns
chicken
vanilla
--------------------------
```

Multilevel Inheritance

1. Write a Java program demonstrating multilevel inheritance with different types of pens and their behaviors.

Code:

```
class Pen {
    String color;
    String inkType;

    public Pen(String color, String inkType) {
        this.color = color;
        this.inkType = inkType;
    }

    public void write() {
        System.out.println("Writing with a " + color + " pen using " + inkType + "
ink.");
    }
}

class FountainPen extends Pen {
    String nibSize;
```

```
    public FountainPen(String color, String inkType, String nibSize) {
        super(color, inkType);
        this.nibSize = nibSize;
    }

    public void write() {
        System.out.println("Writing with a " + color + " fountain pen using " +
inkType + " ink with a " + nibSize + " nib.");
    }

    public void refill() {
        System.out.println("Refilling the " + color + " fountain pen.");
    }
}

class UseAndThrowPen extends FountainPen {
    public UseAndThrowPen(String color, String inkType, String nibSize) {
        super(color, inkType, nibSize);
    }

    public void write() {
        System.out.println("Writing with a " + color + " use-and-throw pen using "
+ inkType + " ink with a " + nibSize + " nib.");
    }

    public void dispose() {
        System.out.println("Disposing of the " + color + " use-and-throw pen.");
    }
}

public class PenTest {
    public static void main(String[] args) {
        Pen pen = new Pen("Blue", "gel");
        pen.write();

        FountainPen fountainPen = new FountainPen("Black", "liquid", "fine");
        fountainPen.write();
        fountainPen.refill();

        UseAndThrowPen useAndThrowPen = new UseAndThrowPen("Red",
"ballpoint", "medium");
        useAndThrowPen.write();
        useAndThrowPen.dispose();
    }
}
```

Output:

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac PenTest.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java PenTest
Writing with a Blue pen using gel ink.
Writing with a Black fountain pen using liquid ink with a fine nib.
Refilling the Black fountain pen.
Writing with a Red use-and-throw pen using ballpoint ink with a medium nib.
Disposing of the Red use-and-throw pen.
```

2. Write a Java program using inheritance to display a categorized restaurant menu including starters, main items, desserts, and drinks.

Code:

```
class RestaurantItem {
    String name;
    double price;

    public RestaurantItem(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public void display() {
        System.out.println("Item: " + name + ", Price: ₹" + price);
    }
}

class Starters extends RestaurantItem {
    public Starters(String name, double price) {
        super(name, price);
    }

    public void display() {
        System.out.println("----------###----------");
        System.out.println("Starter: " + name + ", Price: ₹" + price);
        System.out.println("----------------------");
    }

    public static void displayStarters() {
        Starters starter1 = new Starters("Garlic Bread", 120);
        Starters starter2 = new Starters("Bruschetta", 100);
        Starters starter3 = new Starters("Stuffed Mushrooms", 70);
        Starters starter4 = new Starters("Spring Rolls", 90);
        starter1.display();
        starter2.display();
        starter3.display();
        starter4.display();
    }
```

```
}

class MainItems extends RestaurantItem {
    public MainItems(String name, double price) {
        super(name, price);
    }

    public void display() {
        System.out.println("----------###----------");
        System.out.println("Main Item: " + name + ", Price: ₹" + price);
        System.out.println("---------------------");
    }

    public static void displayMainItems() {
        MainItems mainItem1 = new MainItems("Grilled Chicken", 140);
        MainItems mainItem2 = new MainItems("Pasta Primavera", 210);
        MainItems mainItem3 = new MainItems("Beef Steak", 349);
        MainItems mainItem4 = new MainItems("Vegetable Curry", 100);
        mainItem1.display();
        mainItem2.display();
        mainItem3.display();
        mainItem4.display();
    }
}

class Desserts extends RestaurantItem {
    public Desserts(String name, double price) {
        super(name, price);
    }

    public void display() {
        System.out.println("----------###----------");
        System.out.println("Dessert: " + name + ", Price: ₹" + price);
        System.out.println("---------------------");
    }

    public static void displayDesserts() {
        Desserts dessert1 = new Desserts("Chocolate Cake", 60);
        Desserts dessert2 = new Desserts("Ice Cream Sundae", 80);
        Desserts dessert3 = new Desserts("Fruit Tart", 30);
        Desserts dessert4 = new Desserts("Cheesecake", 60);
        dessert1.display();
        dessert2.display();
        dessert3.display();
        dessert4.display();
    }
```

```
}

class Drinks extends RestaurantItem {
    public Drinks(String name, double price) {
        super(name, price);
    }

    public void display() {
        System.out.println("----------###----------");
        System.out.println("Drink: " + name + ", Price: ₹" + price);
        System.out.println("----------------------");
    }

    public static void displayDrinks() {
        Drinks drink1 = new Drinks("Coke", 76);
        Drinks drink2 = new Drinks("Lemonade", 89);
        Drinks drink3 = new Drinks("Iced Tea", 200);
        Drinks drink4 = new Drinks("Water", 20);
        drink1.display();
        drink2.display();
        drink3.display();
        drink4.display();
    }
}

public class RestaurantMenu {
    public static void main(String[] args) {
        Starters.displayStarters();
        MainItems.displayMainItems();
        Desserts.displayDesserts();
        Drinks.displayDrinks();
    }
}
```

Output:

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

```
C:\Users\yogir\OneDrive\Documents\Desktop\practice>java RestaurantMenu
----------###----------
Starter: Garlic Bread, Price: ?120.0
------------------------
----------###----------
Starter: Bruschetta, Price: ?100.0
------------------------
----------###----------
Starter: Stuffed Mushrooms, Price: ?70.0
------------------------
----------###----------
Starter: Spring Rolls, Price: ?90.0
------------------------
----------###----------
Main Item: Grilled Chicken, Price: ?140.0
------------------------
----------###----------
Main Item: Pasta Primavera, Price: ?210.0
------------------------
----------###----------
Main Item: Beef Steak, Price: ?349.0
------------------------
----------###----------
Main Item: Vegetable Curry, Price: ?100.0
------------------------
----------###----------
Dessert: Chocolate Cake, Price: ?60.0
------------------------
----------###----------
Dessert: Ice Cream Sundae, Price: ?80.0
------------------------
----------###----------
Dessert: Fruit Tart, Price: ?30.0
------------------------
----------###----------
Dessert: Cheesecake, Price: ?60.0
------------------------
----------###----------
Drink: Coke, Price: ?76.0
------------------------
----------###----------
Drink: Lemonade, Price: ?89.0
------------------------
----------###----------
Drink: Iced Tea, Price: ?200.0
------------------------
----------###----------
Drink: Water, Price: ?20.0
------------------------
```

HERIECHIAL INHERITANCE

1. Write a Java program using inheritance to model vehicles and calculate their efficiency based on speed and distance.

Code:

```
import java.util.Scanner;

class Vehicle {
   String make;
   int model, year, distance, maxspeed, efficiency;

   public Vehicle(String make, int model, int year, int distance, int maxspeed)
{
      this.make = make;
      this.model = model;
      this.year = year;
      this.distance = distance;
      this.maxspeed = maxspeed;
   }
```

```
    public void special() {
        if (maxspeed != 0) {
            efficiency = (distance / maxspeed) * 100;
        } else {
            efficiency = 0;
        }
    }
}

class Truck extends Vehicle {
    public Truck(String make, int model, int year, int distance, int maxspeed) {
        super(make, model, year, distance, maxspeed);
        special();
    }

    public void displayTruckInfo() {
        System.out.println("--------@@@@---------");
        System.out.println("Make: " + make);
        System.out.println("Year: " + year);
        System.out.println("Model: " + model);
        System.out.println("The Speed of the Truck is: " + maxspeed);
        System.out.println("The distance travelled by truck is: " + distance);
        System.out.println("The efficiency is: " + efficiency + "%");
        System.out.println("----------------------");
    }
}

class Car extends Vehicle {

    public Car(String make, int model, int year, int distance, int maxspeed) {
        super(make, model, year, distance, maxspeed);
        special();
    }

    public void displayCarInfo() {
        System.out.println("--------@@@@---------");
        System.out.println("Make: " + make);
        System.out.println("Year: " + year);
        System.out.println("Model: " + model);
        System.out.println("The Speed of the Car is: " + maxspeed);
        System.out.println("The distance travelled by Car is: " + distance);
        System.out.println("The efficiency is: " + efficiency + "%");
        System.out.println("----------------------");
    }
}
```

```
public class Details {
   public static void main(String[] args) {
      Scanner obj = new Scanner(System.in);

      System.out.println("Enter the make of Truck:");
      String make = obj.nextLine();

      System.out.println("Enter the model of Truck:");
      int model = obj.nextInt();

      System.out.println("Enter the Year of Truck:");
      int year = obj.nextInt();

      System.out.println("Enter the maxspeed of Truck:");
      int maxspeed = obj.nextInt();

      System.out.println("Enter the distance travelled by Truck:");
      int distance = obj.nextInt();

      Truck obj1 = new Truck(make, model, year, distance, maxspeed);
      obj1.displayTruckInfo();

      Car obj2 = new Car(make, model, year, distance, maxspeed);
      obj2.displayCarInfo();
   }
}
```

Output:

```
Enter the make of Truck:
toyato
Enter the model of Truck:
6
Enter the Year of Truck:
2006
Enter the maxspeed of Truck:
46
Enter the distance travelled by Truck:
654
--------@@@@---------
Make: toyato
Year: 2006
Model: 6
The Speed of the Truck is: 46
The distance travelled by truck is: 654
The efficiency is: 1400%
----------------------
--------@@@@---------
Make: toyato
Year: 2006
Model: 6
The Speed of the Car is: 46
The distance travelled by Car is: 654
The efficiency is: 1400%
----------------------
```

2. Write a Java program using inheritance to calculate the area and perimeter of different shapes like Circle and Square.

Code:

```
class Shape {

    double area;

    double perimeter;



    public Shape() {

        this.area = 0.0;
```

```
        this.perimeter = 0.0;
    }

    public void calculateArea() {
        System.out.println("The area is: " + area);
    }

    public void calculatePerimeter() {
        System.out.println("The perimeter is: " + perimeter);
    }
}

class Circle extends Shape {
    int radius;

    public Circle(int radius) {
        this.radius = radius;
    }

    public void calculateArea() {
        area = Math.PI * radius * radius;
        System.out.println("The area of Circle is: " + area);
    }

    public void calculatePerimeter() {
```

```
        perimeter = 2 * Math.PI * radius;

        System.out.println("The perimeter of Circle is: " + perimeter);

    }

}


class Square extends Shape {

    int length;


    public Square(int length) {

        this.length = length;

    }


    public void calculateArea() {

        area = length * length;

        System.out.println("The area of Square is: " + area);

    }


    public void calculatePerimeter() {

        perimeter = 4 * length;

        System.out.println("The perimeter of Square is: " + perimeter);

    }

}


public class Main {

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

```
        Circle circle = new Circle(7);

        circle.calculateArea();

        circle.calculatePerimeter();



        Square square = new Square(4);

        square.calculateArea();

        square.calculatePerimeter();

    }

}
```

Output:

```
C:\Users\yogir\OneDrive\Documents\Desktop\practice>java Main
The area of Circle is: 153.93804002589985
The perimeter of Circle is: 43.982297150257104
The area of Square is: 16.0
The perimeter of Square is: 16.0

C:\Users\yogir\OneDrive\Documents\Desktop\practice>
```

HYBRID INHERITANCE

1. Write a Java program using Hybrid inheritance to model different bank accounts like savings, current, and loan.

Code:

```
class Bank {
    String bankName;

    public Bank(String bankName) {
        this.bankName = bankName;
    }

    public void displayBankInfo() {
        System.out.println("Bank Name: " + bankName);
    }
}
```

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

```
class SavingsAccount extends Bank {
   double interestRate;

   public SavingsAccount(String bankName, double interestRate) {
      super(bankName);

      this.interestRate = interestRate;
   }

   public void displaySavingsInfo() {
      System.out.println("Savings Account at " + bankName + " with Interest
Rate: " + interestRate + "%");
   }
}

class CurrentAccount extends Bank {
   double overdraftLimit;

   public CurrentAccount(String bankName, double overdraftLimit) {
      super(bankName);
      this.overdraftLimit = overdraftLimit;
   }

   public void displayCurrentInfo() {
      System.out.println("Current Account at " + bankName + " with Overdraft
Limit: ₹" + overdraftLimit);
   }
}

class LoanAccount extends SavingsAccount {
   double loanAmount;

   public LoanAccount(String bankName, double interestRate, double
loanAmount) {
      super(bankName, interestRate);
      this.loanAmount = loanAmount;
   }

   public void displayLoanInfo() {
      System.out.println("Loan Account at " + bankName + " with Loan
Amount: ₹" + loanAmount +
            " and Interest Rate: " + interestRate + "%");
   }
}

public class BankTest {
```

```
    public static void main(String[] args) {
        SavingsAccount savings = new SavingsAccount("SBI Bank", 4.5);
        savings.displayBankInfo();
        savings.displaySavingsInfo();

        CurrentAccount current = new CurrentAccount("HDFC Bank", 5000);
        current.displayBankInfo();
        current.displayCurrentInfo();


        LoanAccount loan = new LoanAccount("ICICI Bank", 5.0, 200000);
        loan.displayBankInfo();
        loan.displaySavingsInfo();
        loan.displayLoanInfo();
    }
}
```

Output:

```
C:\path\to\your\java\files> java BankTest
Bank Name: SBI
Savings Account at SBI with Interest Rate: 4.5%
Bank Name: HDFC
Current Account at HDFC with Overdraft Limit: ₹5000.0
Bank Name: ICICI
Savings Account at ICICI with Interest Rate: 5.0%
Loan Account at ICICI with Loan Amount: ₹200000.0 and Interest
```

2. Write a Java program using multilevel and hierarchical inheritance to manage a ticketing system for different types of transport like trains and buses.

Code:

```
class Ticket {
    protected double price;

    public Ticket(double price) {
        this.price = price;
    }

    public double getPrice() {
        return price;
    }

    public void displayTicketInfo() {
```

```
        System.out.println("Ticket Price: $" + price);
    }
}

class TrainTicket extends Ticket {
    private String trainNumber;

    public TrainTicket(double price, String trainNumber) {
        super(price);
        this.trainNumber = trainNumber;
    }

    public void displayTicketInfo() {
        super.displayTicketInfo();
        System.out.println("Train Number: " + trainNumber);
    }
}

class BusTicket extends Ticket {
    private String busNumber;

    public BusTicket(double price, String busNumber) {
        super(price);
        this.busNumber = busNumber;
    }

    public void displayTicketInfo() {
        super.displayTicketInfo();
        System.out.println("Bus Number: " + busNumber);
    }
}

class ACTicket extends TrainTicket {
    public ACTicket(double price, String trainNumber) {
        super(price, trainNumber);
    }

    public void displayTicketInfo() {
        System.out.println("AC Train Ticket:");
        super.displayTicketInfo();
    }
}

class SleeperTicket extends TrainTicket {
    public SleeperTicket(double price, String trainNumber) {
        super(price, trainNumber);
```

```
    }

    public void displayTicketInfo() {
        System.out.println("Sleeper Train Ticket:");
        super.displayTicketInfo();
    }
}

class ACBusTicket extends BusTicket {
    public ACBusTicket(double price, String busNumber) {
        super(price, busNumber);
    }

    public void displayTicketInfo() {
        System.out.println("AC Bus Ticket:");
        super.displayTicketInfo();
    }
}

class SleeperBusTicket extends BusTicket {
    public SleeperBusTicket(double price, String busNumber) {
        super(price, busNumber);
    }

    public void displayTicketInfo() {
        System.out.println("Sleeper Bus Ticket:");
        super.displayTicketInfo();
    }
}

public class TicketingSystem {
    public static void main(String[] args) {
        Ticket trainTicket = new TrainTicket(50.0, "12345");
        Ticket busTicket = new BusTicket(30.0, "54321");
        Ticket acTrainTicket = new ACTicket(80.0, "12345");
        Ticket sleeperTrainTicket = new SleeperTicket(60.0, "12345");
        Ticket acBusTicket = new ACBusTicket(40.0, "54321");
        Ticket sleeperBusTicket = new SleeperBusTicket(35.0, "54321");

        trainTicket.displayTicketInfo();
        System.out.println();

        busTicket.displayTicketInfo();
        System.out.println();

        acTrainTicket.displayTicketInfo();
```

```
        System.out.println();

        sleeperTrainTicket.displayTicketInfo();
        System.out.println();

        acBusTicket.displayTicketInfo();
        System.out.println();

        sleeperBusTicket.displayTicketInfo();
    }
}
```

Output:

```
C:\Users\yogir\OneDrive\Documents\Desktop\practice>java TicketingSystem
Ticket Price: $50.0
Train Number: 12345

Ticket Price: $30.0
Bus Number: 54321

AC Train Ticket:
Ticket Price: $80.0
Train Number: 12345

Sleeper Train Ticket:
Ticket Price: $60.0
Train Number: 12345

AC Bus Ticket:
Ticket Price: $40.0
Bus Number: 54321

Sleeper Bus Ticket:
Ticket Price: $35.0
Bus Number: 54321
```

POLYMORPHISM

Constructor programs

1. Demonstrate constructor chaining and encapsulation with `Shape` as base class and `Circle`, `Rectangle` as derived classes showing area and color.

Code:

```
class Shape {
    private String color;

    public Shape(String color) {
        this.color = color;
        System.out.println("A shape of color " + color + " has been created.");
    }

    public void displayColor() {
```

```
        System.out.println("Color: " + color);
    }
}

class Circle extends Shape {
    private double radius;

    public Circle(String color, double radius) {
        super(color);
        this.radius = radius;
        System.out.println("Circle created with radius: " + radius);
    }

    public void area() {
        double area = Math.PI * radius * radius;
        System.out.println("Area of Circle: " + area);
    }
}

class Rectangle extends Shape {
    private double length;
    private double width;

    public Rectangle(String color, double length, double width) {
        super(color);
        this.length = length;
        this.width = width;
        System.out.println("Rectangle created with length: " + length + " and
width: " + width);
    }

    public void area() {
        double area = length * width;
        System.out.println("Area of Rectangle: " + area);
    }
}

public class ShapeTest {
    public static void main(String[] args) {
        Circle circle = new Circle("Red", 5.0);
        circle.area();
        circle.displayColor();

        System.out.println();

        Rectangle rectangle = new Rectangle("Blue", 4.0, 6.0);
```

```
        rectangle.area();
        rectangle.displayColor();
    }
}
```

Output:

```
C:\Users\yogir\OneDrive\Documents\Desktop\practice>java ShapeTest
A shape of color Red has been created.
Circle created with radius: 5.0
Area of Circle: 78.53981633974483
Color: Red
A shape of color Blue has been created.
Rectangle created with length: 4.0 and width: 6.0
Area of Rectangle: 24.0
Color: Blue
```

CONSTRUCTOR OVERLOADING

1. How can constructor overloading be used to handle different types of payment details in a class?

Code:

```
class Payment {
    private String paymentId;
    private double amount;
    private String paymentMethod;
    private String currency;

    public Payment(String paymentId, double amount, String paymentMethod) {
        this.paymentId = paymentId;
        this.amount = amount;
        this.paymentMethod = paymentMethod;
        this.currency = "USD";
    }

    public Payment(String paymentId, double amount, String paymentMethod,
String currency) {
        this.paymentId = paymentId;
        this.amount = amount;
        this.paymentMethod = paymentMethod;
        this.currency = currency;
    }

    public Payment(String paymentId, double amount) {
        this.paymentId = paymentId;
        this.amount = amount;
        this.paymentMethod = "Cash";
```

```
        this.currency = "USD";
    }

    public void displayPaymentDetails() {
        System.out.println("Payment ID: " + paymentId);
        System.out.println("Amount: " + amount);
        System.out.println("Payment Method: " + paymentMethod);
        System.out.println("Currency: " + currency);
        System.out.println("--------------------------");
    }
}

public class OnlinePaymentSystem {
    public static void main(String[] args) {
        Payment creditCardPayment = new Payment("CC123", 150.00, "Credit
Card");
        creditCardPayment.displayPaymentDetails();


        Payment bankTransferPayment = new Payment("BT456", 200.00, "Bank
Transfer", "EUR");
        bankTransferPayment.displayPaymentDetails();

        Payment cashPayment = new Payment("C789", 50.00);
        cashPayment.displayPaymentDetails();
    }
}
```

Output:

```
C:\Users\yogir\OneDrive\Documents\Desktop\practice>java OnlinePaymentSystem
Payment ID: CC123
Amount: 150.0
Payment Method: Credit Card
Currency: USD
--------------------------
Payment ID: BT456
Amount: 200.0
Payment Method: Bank Transfer
Currency: EUR
--------------------------
Payment ID: C789
Amount: 50.0
Payment Method: Cash
Currency: USD
--------------------------
```

Method Overloading

1. How does method overloading help calculate total price with varying conditions in a shop system?

Code:

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

```
class Shop {
   public void calculateTotalPrice(double price, int quantity) {
      double total = price * quantity;
      System.out.println("Total Price (without discount): " + total);
   }

   public void calculateTotalPrice(double price, int quantity, double discount)
{
      double total = price * quantity;
      total = total - (total * discount / 100);
      System.out.println("Total Price (with discount): " + total);
   }

   public void calculateTotalPrice(double price, int quantity, double discount,
double tax) {
      double total = price * quantity;
      total = total - (total * discount / 100);
      total = total + (total * tax / 100);
      System.out.println("Total Price (with discount and tax): " + total);
   }
}

public class ShopLite {
   public static void main(String[] args) {
      Shop shop = new Shop();
      double price1 = 100.0;
      int quantity1 = 2;
      double discount1 = 10.0;
      double tax1 = 5.0;

      shop.calculateTotalPrice(price1, quantity1);
      shop.calculateTotalPrice(price1, quantity1, discount1);
      shop.calculateTotalPrice(price1, quantity1, discount1, tax1);
   }
}
```

Output:

```
C:\Users\yogir\OneDrive\Documents\Desktop\practice>java ShopLite
Total Price (without discount): 200.0
Total Price (with discount): 180.0
Total Price (with discount and tax): 189.0
```

2. How is method overloading implemented to perform multiplication of different numbers of integers?

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

Code:

```
class Multiplier {
    public void multiply(int a, int b) {
        int result = a * b;
        System.out.println("Multiplication of two numbers: " + result);
    }

    public void multiply(int a, int b, int c) {
        int result = a * b * c;
        System.out.println("Multiplication of three numbers: " + result);
    }

    public void multiply(int a, int b, int c, int d) {
        int result = a * b * c * d;
        System.out.println("Multiplication of four numbers: " + result);
    }
}

public class MultiplierExample {
    public static void main(String[] args) {
        Multiplier multiplier = new Multiplier();
        multiplier.multiply(2, 3);
        multiplier.multiply(2, 3, 4);
        multiplier.multiply(2, 3, 4, 5);
    }
}
```

Output:

```
C:\Users\yogir\OneDrive\Documents\Desktop\practice>java MultiplierExample
Multiplication of two numbers: 6
Multiplication of three numbers: 24
Multiplication of four numbers: 120
```

METHOD OVERIDING

1. How is method overriding used in different subclasses to customize the behavior of a base class method?

Code:

```
class Home {
    public void display() {
        System.out.println("We are in the home");
    }
}

class Apartment extends Home {
    public void display() {
```

```
        System.out.println("We are in the apartment");
    }
}

class Company extends Home {
    public void display() {
        System.out.println("We are in the Company");
    }
}

public class Room {
    public static void main(String[] args) {
        Home myHome = new Home();
        Home myApartment = new Apartment();
        Company myObj = new Company();

        myHome.display();
        myApartment.display();
        myObj.display();
    }
}
```

Output:

```
C:\Users\yogir\OneDrive\Documents\Desktop\practice>java Room
We are in the home
We are in the apartment
We are in the Company
```

2. How does method overriding help in calculating different delivery item prices using inheritance?

Code:

```
class DeliveryItem {
    private double basePrice;

    public void setBasePrice(double basePrice) {
        this.basePrice = basePrice;
    }

    public double getBasePrice() {
        return basePrice;
    }
```

```
    public void calculatePrice() {
        System.out.println("Price of Regular Delivery Item: " + basePrice);
    }
}

class ExpressDeliveryItem extends DeliveryItem {
    private double expressFee;

    public void setExpressFee(double expressFee) {
        this.expressFee = expressFee;
    }

    @Override
    public void calculatePrice() {
        double totalPrice = getBasePrice() + expressFee;
        System.out.println("Price of Express Delivery Item: " + totalPrice);
    }
}

public class Delivery {
    public static void main(String[] args) {
        DeliveryItem regularItem = new DeliveryItem();
        regularItem.setBasePrice(100.0);

        ExpressDeliveryItem expressItem = new ExpressDeliveryItem();
        expressItem.setBasePrice(100.0);
        expressItem.setExpressFee(20.0);

        regularItem.calculatePrice();
        expressItem.calculatePrice();
    }
}
```

Output:

```
C:\Users\yogir\OneDrive\Documents\Desktop\practice>java Delivery
Price of Regular Delivery Item: 100.0
Price of Express Delivery Item: 120.0
```

ABSTRACTION

INTERFACE PROGRAMS

1. How does Java use interfaces to enable polymorphic behavior in flying vehicles?

Code:

```
interface Flyable {
```

```
    void fly_obj();
}

class Jet implements Flyable {
    public void fly_obj() {
        System.out.println("The Jet is flying at a speed of 1500 km/h.");
    }
}

class Plane implements Flyable {
    public void fly_obj() {
        System.out.println("The Plane is flying at a speed of 900 km/h.");
    }
}

class Chopper implements Flyable {
    public void fly_obj() {
        System.out.println("The Chopper is flying at a speed of 300 km/h.");
    }
}

public class FlyableTest {
    public static void main(String[] args) {
        Flyable jet = new Jet();
        Flyable plane = new Plane();
        Flyable chopper = new Chopper();

        jet.fly_obj();
        plane.fly_obj();

        chopper.fly_obj();
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>javac FlyableTest.java

C:\Users\DELL\OneDrive\Desktop\rohan>java FlyableTest
The Jet is flying at a speed of 1500 km/h.
The Plane is flying at a speed of 900 km/h.
The Chopper is flying at a speed of 300 km/h.
```

2. How is the `Resizable` interface implemented to modify the dimensions of a rectangle in Java?

Code:

```
interface Resizable {
```

```
    void resizeWidth(int width);
    void resizeHeight(int height);
}

class Rectangle implements Resizable {
    private int width;
    private int height;

    public Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }

    public void resizeWidth(int width) {
        if (width > 0) {
            this.width = width;
            System.out.println("Width resized to: " + this.width);
        } else {
            System.out.println("Invalid width value");
        }
    }

    public void resizeHeight(int height) {
        if (height > 0) {
            this.height = height;
            System.out.println("Height resized to: " + this.height);
        } else {
            System.out.println("Invalid height value");
        }
    }

    public void display() {
        System.out.println("Rectangle Dimensions: Width = " + width + ", Height
= " + height);
    }

    public static void main(String[] args) {
        Rectangle rect = new Rectangle(10, 20);
        rect.display();
        rect.resizeWidth(30);
        rect.resizeHeight(40);
        rect.display();
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Rectangle
Rectangle Dimensions: Width = 10, Height = 20
Width resized to: 30
Height resized to: 40
Rectangle Dimensions: Width = 30, Height = 40
```

3. How is the `Playable` interface used to demonstrate polymorphism in different sports games in Java?

Code:

```
interface Playable {
   void play();
}

class Football implements Playable {
   public void play() {
      System.out.println("Playing Football");
   }
}

class Volleyball implements Playable {
   public void play() {
      System.out.println("Playing Volleyball");
   }
}

class Basketball implements Playable {
   public void play() {
      System.out.println("Playing Basketball");
   }
}

public class Main {

   public static void main(String[] args) {
      Playable football = new Football();
      Playable volleyball = new Volleyball();
      Playable basketball = new Basketball();

      football.play();
      volleyball.play();
      basketball.play();
   }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Playing Football
Playing Volleyball
Playing Basketball
```

4. How is the concept of interface implementation demonstrated through the `Playable` interface in Java?

Code:

```
interface CarOrder {
   void placeOrder();
   void trackOrder();
   void cancelOrder();
}

class Car implements CarOrder {
   public void placeOrder() {
      System.out.println("Car order placed.");
   }
   public void trackOrder() {
      System.out.println("Tracking car order.");
   }
   public void cancelOrder() {
      System.out.println("Car order canceled.");
   }
}

public class Main {
   public static void main(String[] args) {
      Car myCar = new Car();
      myCar.placeOrder();
      myCar.trackOrder();
      myCar.cancelOrder();
   }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Car order placed.
Tracking car order.
Car order canceled.
```

ABSTRACT CLASSES PROGRAMS

1. How does the `BankAccount` abstract class enforce method implementation in its subclasses?

Code:

```
abstract class BankAccount {
    double balance;

    public BankAccount(double balance) {
        this.balance = balance;
    }

    abstract void deposit(double amount);
    abstract void withdraw(double amount);

    public void displayBalance() {
        System.out.println("Current balance: " + balance);
    }
}

class SavingsAccount extends BankAccount {
    private static final double MIN_BALANCE = 500;

    public SavingsAccount(double balance) {
        super(balance);
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited " + amount + " in Savings Account.");
    }

    public void withdraw(double amount) {
        if (balance - amount >= MIN_BALANCE) {
            balance -= amount;
            System.out.println("Withdrew " + amount + " from Savings Account.");
        } else {
            System.out.println("Insufficient balance. Minimum balance must be
maintained.");
        }
    }
}

class CurrentAccount extends BankAccount {
    private static final double OVERDRAFT_LIMIT = 1000;

    public CurrentAccount(double balance) {
        super(balance);
    }
```

```
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited " + amount + " in Current Account.");
    }

    public void withdraw(double amount) {
        if (balance - amount >= -OVERDRAFT_LIMIT) {
            balance -= amount;
            System.out.println("Withdrew " + amount + " from Current Account.");
        } else {
            System.out.println("Overdraft limit exceeded. Cannot withdraw.");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        BankAccount savings = new SavingsAccount(1000);
        savings.deposit(500);
        savings.withdraw(800);
        savings.displayBalance();

        BankAccount current = new CurrentAccount(500);
        current.deposit(1000);
        current.withdraw(1800);
        current.displayBalance();
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Deposited 500.0 in Savings Account.
Withdrew 800.0 from Savings Account.
Current balance: 700.0
Deposited 1000.0 in Current Account.
Withdrew 1800.0 from Current Account.
Current balance: -300.0
```

2. How does the `Animal` abstract class enforce method overriding in its subclasses?

Code:

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

```
abstract class Animal {
   abstract void sound();
}

class Lion extends Animal {
   public void sound() {
      System.out.println("Lion roars");
   }
}

class Tiger extends Animal {
   public void sound() {
      System.out.println("Tiger growls");
   }
}

public class Main {
   public static void main(String[] args) {
      Animal lion = new Lion();
      Animal tiger = new Tiger();
      lion.sound();
      tiger.sound();
   }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Lion roars
Tiger growls
```

3. How does the code demonstrate abstraction and method overriding using `Animal` **and** `RapidoBooking` **classes?**

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

Code:

```
abstract class Animal {
   abstract void sound();
}

class Lion extends Animal {
   public void sound() {
      System.out.println("Lion roars");
   }
}

class Tiger extends Animal {
   public void sound() {
      System.out.println("Tiger growls");
   }
}

abstract class RapidoBooking {
   String pickupLocation;
   String dropLocation;

   public RapidoBooking(String pickupLocation, String dropLocation) {
      this.pickupLocation = pickupLocation;
      this.dropLocation = dropLocation;
   }

   abstract void bookRide();
}

class BikeRide extends RapidoBooking {
   public BikeRide(String pickupLocation, String dropLocation) {
      super(pickupLocation, dropLocation);
   }

   public void bookRide() {
      System.out.println("Bike ride booked from " + pickupLocation + " to " +
dropLocation);
   }
}

public class Main {
   public static void main(String[] args) {
      Animal lion = new Lion();
      Animal tiger = new Tiger();
      lion.sound();
```

```
        tiger.sound();
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Lion roars
Tiger growls
```

4. How does the code implement abstraction and polymorphism in calculating area and perimeter of different shapes?

Code:

```
abstract class GeometricShape {
    abstract void area();
    abstract void perimeter();
}

class Triangle extends GeometricShape {
    private double base, height, side1, side2, side3;

    public Triangle(double base, double height, double side1, double side2,
double side3) {
        this.base = base;
        this.height = height;
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
    }

    public void area() {
        System.out.println("Triangle Area: " + (0.5 * base * height));
    }

    public void perimeter() {
        System.out.println("Triangle Perimeter: " + (side1 + side2 + side3));
    }
}

class Square extends GeometricShape {
    private double side;

    public Square(double side) {
        this.side = side;
    }
```

```
    public void area() {
        System.out.println("Square Area: " + (side * side));
    }

    public void perimeter() {
        System.out.println("Square Perimeter: " + (4 * side));
    }
}

public class Main {
    public static void main(String[] args) {
        GeometricShape triangle = new Triangle(5, 4, 3, 4, 5);
        GeometricShape square = new Square(4);

        triangle.area();
        triangle.perimeter();
        square.area();
        square.perimeter();
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Triangle Area: 10.0
Triangle Perimeter: 12.0
Square Area: 16.0
Square Perimeter: 16.0

C:\Users\DELL\OneDrive\Desktop\rohan>
```

ENCAPSULATION

ENCAPSULATION PROGRAMS

1. How is encapsulation implemented using getter and setter methods in this Java program?

Code:

```
class Person {
    private String name;
    private int age;
    private String country;

    public void getName() {
        System.out.println("Name: " + name);
    }

    public void setName(String name) {
        this.name = name;
    }

    public void getAge() {
        System.out.println("Age: " + age);
    }

    public void setAge(int age) {
        this.age = age;
    }

    public void getCountry() {
        System.out.println("Country: " + country);
    }

    public void setCountry(String country) {
        this.country = country;
    }
}

public class Main {
    public static void main(String[] args) {
        Person person = new Person();
        person.setName("John");
        person.setAge(30);
        person.setCountry("USA");
        person.getName();
        person.getAge();
```

```
        person.getCountry();
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Name: John
Age: 30
Country: USA

C:\Users\DELL\OneDrive\Desktop\rohan>
```

2. How does the Rectangle class use encapsulation to manage user input for dimensions?

Code:

```
import java.util.Scanner;

class Rectangle {
    private double length;
    private double width;

    public void setLength(double length) {
        this.length = length;
    }

    public void setWidth(double width) {
        this.width = width;
    }

    public void getLength() {
        System.out.println("Length: " + length);
    }

    public void getWidth() {
        System.out.println("Width: " + width);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Rectangle rectangle = new Rectangle();

        System.out.print("Enter length: ");
        double length = sc.nextDouble();
        rectangle.setLength(length);
```

```
        System.out.print("Enter width: ");
        double width = sc.nextDouble();
        rectangle.setWidth(width);

        rectangle.getLength();
        rectangle.getWidth();

        sc.close();
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>javac Main

C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Enter length: 20
Enter width: 40
Length: 20.0
Width: 40.0
```

3. How does the `House` class calculate the total price based on its area and price per square meter?

Code:

```
class House {
    private String address;
    private int numberOfRooms;
    private double area;

    public House(String address, int numberOfRooms, double area) {
        this.address = address;
        this.numberOfRooms = numberOfRooms;
        this.area = area;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
```

```
    public int getNumberOfRooms() {
        return numberOfRooms;
    }

    public void setNumberOfRooms(int numberOfRooms) {
        this.numberOfRooms = numberOfRooms;
    }

    public double getArea() {
        return area;
    }

    public void setArea(double area) {
        this.area = area;
    }

    public double calculatePrice(double pricePerSquareMeter) {
        return area * pricePerSquareMeter;
    }
}

public class Main {
    public static void main(String[] args) {
        House house = new House("123 Main St", 3, 150.0);
        System.out.println("Address: " + house.getAddress());
        System.out.println("Number of Rooms: " + house.getNumberOfRooms());
        System.out.println("Area: " + house.getArea() + " sq. meters");

        double pricePerSquareMeter = 2000.0;
        double price = house.calculatePrice(pricePerSquareMeter);
        System.out.println("Price of the house: $" + price);
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Address: 123 Main St
Number of Rooms: 3
Area: 150.0 sq. meters
Price of the house: $300000.0
```

4. How does the `CricketPlayer` class store and display information about a player's name, runs, and wickets?

Code:

```
class CricketPlayer {
```

```
    private String name;
    private int runs;
    private int wickets;

    public void setName(String name) {
        this.name = name;
    }

    public void setRuns(int runs) {
        this.runs = runs;
    }

    public void setWickets(int wickets) {
        this.wickets = wickets;
    }

    public void getName() {
        System.out.println("Player Name: " + name);
    }

    public void getRuns() {
        System.out.println("Total Runs: " + runs);
    }

    public void getWickets() {
        System.out.println("Total Wickets: " + wickets);
    }
}

public class Main {
    public static void main(String[] args) {
        CricketPlayer player = new CricketPlayer();
        player.setName("Virat Kohli");
        player.setRuns(12000);
        player.setWickets(4);
        player.getName();
        player.getRuns();
        player.getWickets();
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Player Name: Virat Kohli
Total Runs: 12000
Total Wickets: 4
```

5. Create a `BankAccount` class that hides balance and allows secure deposit and withdrawal operations.

Code:

```
class BankAccount {
    private double balance;

    public BankAccount(double initialBalance) {
        balance = initialBalance;
    }

    public void deposit(double amount) {
        if (amount > 0)
            balance += amount;
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance)
            balance -= amount;
        else
            System.out.println("Insufficient balance!");
    }

    public double getBalance() {
        return balance;
    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac BankAccount.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java BankAccount
Deposited: ?500.0
Withdrawn: ?200.0
Insufficient balance! Available: ?1300.0
Deposit amount must be positive.
Final Balance: ?1300.0
```

6. Create a `Student` class that encapsulates marks with validation between 0 and 100.

Code:

```
class Student {
    private String name;
    private int marks;

    public Student(String name) {
        this.name = name;
```

```
    }

    public void setMarks(int marks) {
        if (marks >= 0 && marks <= 100)
            this.marks = marks;
        else
            System.out.println("Invalid marks entered!");
    }

    public int getMarks() {
        return marks;
    }

    public String getName() {
        return name;
    }

    public static void main(String[] args) {
        Student s = new Student("Vivek");
        s.setMarks(85);
        System.out.println(s.getName() + "'s Marks: " + s.getMarks());
    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Student.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Student
Vivek's Marks: 85
PS C:\Users\user\OneDrive\Documents\Java Programs>
```

7. Design a `Car` class that controls speed with a max limit using encapsulation.

Code:

```
class Car {
    private int speed = 0;
    private final int MAX_SPEED = 200;

    public void accelerate(int increment) {
        if (speed + increment <= MAX_SPEED)
            speed += increment;
        else
            System.out.println("Speed limit exceeded!");
    }

    public void brake(int decrement) {
        if (speed - decrement >= 0)
            speed -= decrement;
```

```
        else
            speed = 0;
    }

    public int getSpeed() {
        return speed;
    }

    public static void main(String[] args) {
        Car c = new Car();
        c.accelerate(50);
        c.brake(20);
        System.out.println("Current Speed: " + c.getSpeed() + " km/h");
    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac ^C
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Car.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Car
Current Speed: 30 km/h
PS C:\Users\user\OneDrive\Documents\Java Programs>
```

8. Implement a `Patient` class that securely manages age and temperature with range checks.

Code:

```
class Patient {
    private int age;
    private double temperature;

    public void setAge(int age) {
        if (age >= 0)
            this.age = age;
        else
            System.out.println("Invalid age!");
    }

    public void setTemperature(double temperature) {
        if (temperature >= 95 && temperature <= 108)
            this.temperature = temperature;
        else
            System.out.println("Temperature out of range!");
    }

    public int getAge() {
```

```
        return age;
    }

    public double getTemperature() {
        return temperature;
    }

    public static void main(String[] args) {
        Patient p = new Patient();
        p.setAge(25);
        p.setTemperature(98.6);
        System.out.println("Age: " + p.getAge() + ", Temperature: " +
p.getTemperature() + "°F");
    }
}
```

Output:

```
PS C:\Users\user\OneDrive\Documents\Java Programs> javac Patient.java
PS C:\Users\user\OneDrive\Documents\Java Programs> java Patient
Age: 25, Temperature: 98.6°F
PS C:\Users\user\OneDrive\Documents\Java Programs>
```

PACKAGES PROGRAMS

1. How does the program use an `ArrayList` to store and print a list of names?

Code:

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> names = new ArrayList<>();
        names.add("Alice");
        names.add("Bob");
        names.add("Charlie");

        System.out.println("Names in the list:");
        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Names in the list:
Alice
Bob
Charlie
```

2. How does the program create a TCP client to send a message to a server and receive a response?

Code:

```
import java.io.*;
import java.net.*;

public class Main {
    public static void main(String[] args) {
        String hostname = "localhost";
        int port = 12345;

        try (
            Socket socket = new Socket(hostname, port);
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()))
        ) {
            out.println("Hello Server!");
            String response = in.readLine();
            System.out.println("Server response: " + response);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
java.net.ConnectException: Connection refused: connect
        at java.base/sun.nio.ch.Net.connect0(Native Method)
        at java.base/sun.nio.ch.Net.connect(Net.java:589)
        at java.base/sun.nio.ch.Net.connect(Net.java:578)
        at java.base/sun.nio.ch.NioSocketImpl.connect(NioSocketImpl.java:583)
        at java.base/java.net.SocksSocketImpl.connect(SocksSocketImpl.java:327)
        at java.base/java.net.Socket.connect(Socket.java:760)
        at java.base/java.net.Socket.connect(Socket.java:695)
        at java.base/java.net.Socket.<init>(Socket.java:564)
        at java.base/java.net.Socket.<init>(Socket.java:328)
        at Main.main(Main.java:9)
```

3. How does the program demonstrate working with current date, time, formatting, time zones, and calculating date differences using the Java `time` API?

Code:

```
import java.time.*;
import java.time.format.DateTimeFormatter;

public class Main {
   public static void main(String[] args) {
      LocalDate currentDate = LocalDate.now();
      LocalTime currentTime = LocalTime.now();
      LocalDateTime currentDateTime = LocalDateTime.now();

      DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd HH:mm:ss");
      String formattedDateTime = currentDateTime.format(formatter);

      ZonedDateTime zonedDateTime =
ZonedDateTime.now(ZoneId.of("America/New_York"));

      LocalDate pastDate = LocalDate.of(2020, Month.JANUARY, 1);
      Period period = Period.between(pastDate, currentDate);

      System.out.printf("Current Date: %s%n", currentDate);
      System.out.printf("Current Time: %s%n", currentTime);
      System.out.printf("Formatted DateTime: %s%n", formattedDateTime);
      System.out.printf("Zoned DateTime (New York): %s%n",
zonedDateTime);
      System.out.printf("Period from 2020-01-01 to now: %d years, %d months,
%d days%n",
            period.getYears(), period.getMonths(), period.getDays());
   }
}
```

Output:

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

```
C:\Users\DELL\OneDrive\Desktop\rohan>javac Main.java

C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Current Date: 2025-04-03
Current Time: 23:11:12.310761500
Formatted DateTime: 2025-04-03 23:11:12
Zoned DateTime (New York): 2025-04-03T13:41:12.319745500-04:00[America/New_York]
Period from 2020-01-01 to now: 5 years, 3 months, 2 days
```

4. How does this program take user input for name and age and display a personalized greeting?

Code:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your name: ");
        String name = scanner.nextLine();
        System.out.print("Enter your age: ");
        int age = scanner.nextInt();
        System.out.println("Hello, " + name + "! You are " + age + " years old.");
        scanner.close();
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Enter your name: VIRAT KOHLI
Enter your age: 37
Hello, VIRAT KOHLI! You are 37 years old.
```

EXCEPTION HANDLING

1. How does this program use a custom exception to handle even number inputs?

Code:

```
class EvenNumberException extends Exception {
    public EvenNumberException(String message) {
        super(message);
    }
}

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

```
        try {
            checkNumber(4); // Change this number to test different cases
        } catch (EvenNumberException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }

    public static void checkNumber(int number) throws EvenNumberException
{
        if (number % 2 == 0) {
            throw new EvenNumberException("The number " + number + " is
divisible by 2.");
        } else {
            System.out.println("The number " + number + " is not divisible by 2.");
        }
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Exception caught: The number 4 is divisible by 2.
```

2. How does this program use a custom exception to detect negative numbers from a file?

Code:

```
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

class NegativeNumberException extends Exception {
    public NegativeNumberException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        String filePath = "numbers.txt"; // Change this to the path of your file
        try {
            checkNumbers(filePath);
            System.out.println("All numbers are non-negative.");
        } catch (NegativeNumberException e) {
            System.out.println("Exception caught: " + e.getMessage());
```

```
        } catch (IOException e) {
            System.out.println("An error occurred while reading the file: " +
e.getMessage());
        }
    }

    public static void checkNumbers(String filePath) throws
NegativeNumberException, IOException {
        Path path = Paths.get(filePath);
        if (Files.notExists(path)) {
            throw new IOException("File does not exist.");
        }
        List<String> lines = Files.readAllLines(path);
        for (String line : lines) {
            try {
                int number = Integer.parseInt(line.trim());
                if (number < 0) {
                    throw new NegativeNumberException("Negative number found: "
+ number);
                }
            } catch (NumberFormatException e) {
                System.out.println("Invalid number format: " + line);
            }
        }
    }
}
```

Output:

```
:\Users\DELL\OneDrive\Desktop\rohan>java Main
n error occurred while reading the file: File does not exist.
```

3. How does this program handle odd numbers using a user-defined exception in Java?

Code:

```
class OddNumberException extends Exception {
    public OddNumberException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        try {
            checkNumber(5); // Change this number to test different cases
```

```
        } catch (OddNumberException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }

    public static void checkNumber(int number) throws OddNumberException {
        if (number % 2 != 0) {
            throw new OddNumberException("The number " + number + " is
odd.");
        } else {
            System.out.println("The number " + number + " is even.");
        }
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Exception caught: The number 5 is odd.
```

4. How does this Java program use a custom exception to validate if a given string is a palindrome?

Code:

```
class InvalidInputException extends Exception {
    public InvalidInputException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        String input = "madam";
        try {
            checkPalindrome(input);
            System.out.println(input + " is a palindrome.");
        } catch (InvalidInputException e) {
            System.out.println("Exception caught: " + e.getMessage());
        }
    }

    public static void checkPalindrome(String str) throws
InvalidInputException {
        if (str == null || str.isEmpty()) {
            throw new InvalidInputException("Input string is null or empty.");
        }
```

```
        String reversedStr = new StringBuilder(str).reverse().toString();
        if (!str.equals(reversedStr)) {
            throw new InvalidInputException(str + " is not a palindrome.");
        }
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
madam is a palindrome.

C:\Users\DELL\OneDrive\Desktop\rohan>
```

FILE HANDLING

1. How does this Java program create and write content to a file using `FileWriter` with exception handling?

Code:

```
import java.io.FileWriter;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        try (FileWriter writer = new FileWriter("example.txt")) {
            writer.write("Hello, this is a sample file.\n");
            writer.write("Java File Handling Example.\n");
            System.out.println("File created and written successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
File created and written successfully.

C:\Users\DELL\OneDrive\Desktop\rohan>
```

2. How does this Java program read and display the contents of a file using `Scanner` and handle file-not-found exceptions?

Code:

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        try {
            File file = new File("example.txt");
            Scanner reader = new Scanner(file);
            while (reader.hasNextLine()) {
                System.out.println(reader.nextLine());
            }
            reader.close();
        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
            e.printStackTrace();
        }
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Hello, this is a sample file.
Java File Handling Example.
```

3. How does this Java program count the total number of words in a file using `Scanner` and string splitting?

Code:

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        try {
            File file = new File("example.txt");
            Scanner reader = new Scanner(file);
            int wordCount = 0;
            while (reader.hasNextLine()) {
                String line = reader.nextLine();
                String[] words = line.split("\\s+");
                wordCount += words.length;
            }
            reader.close();
```

```
            System.out.println("Total words in file: " + wordCount);
        } catch (FileNotFoundException e) {
            System.out.println("File not found.");
            e.printStackTrace();
        }
    }
}
```

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>javac Main.java

C:\Users\DELL\OneDrive\Desktop\rohan>java Main
Total words in file: 10
```

4. How does this Java program copy the contents of one file to another using `FileReader` and `FileWriter`?

Code:

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        String sourceFile = "example.txt";
        String destinationFile = "copy_example.txt";
        try {
            FileReader reader = new FileReader(sourceFile);
            FileWriter writer = new FileWriter(destinationFile);
            int ch;
            while ((ch = reader.read()) != -1) {
                writer.write(ch);
            }
            reader.close();
            writer.close();
            System.out.println("File copied successfully.");
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

(Chepoori Sai Vivek – CH.SC.U4CSE24108)

Output:

```
C:\Users\DELL\OneDrive\Desktop\rohan>javac Main.java

C:\Users\DELL\OneDrive\Desktop\rohan>java Main
File copied successfully.
```

****