SCHOOL OF
COMPUTING

VAISHNAV H

CH.SC.U4CSE24049

OBJECT ORIENTED PROGRAMMING (23CSE111)

LAB RECORD

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.U4CSE24049 – VAISHNAV H*** 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 on 11/03/2025

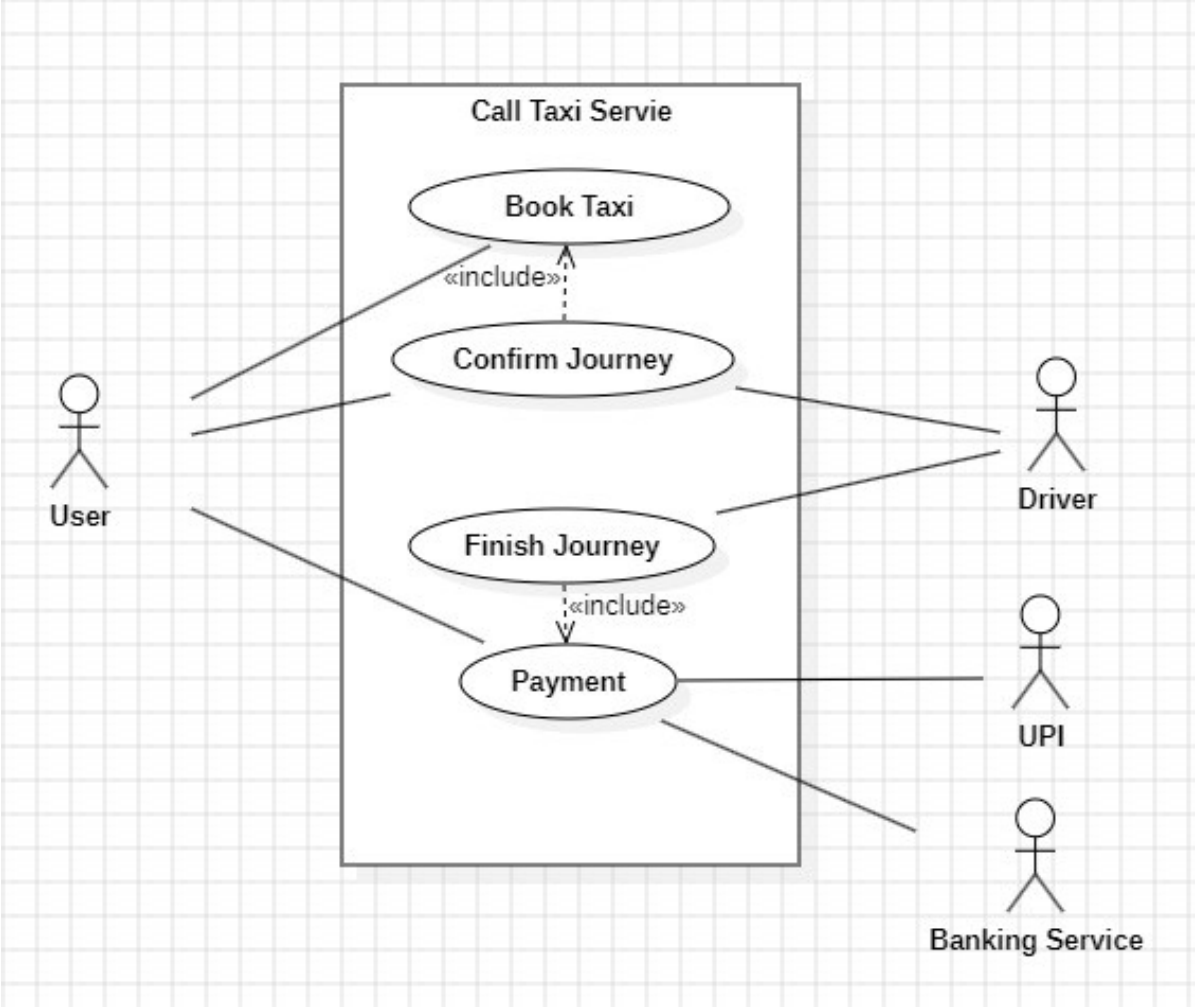
Internal Examiner 1 Internal Examiner 2

INDEX

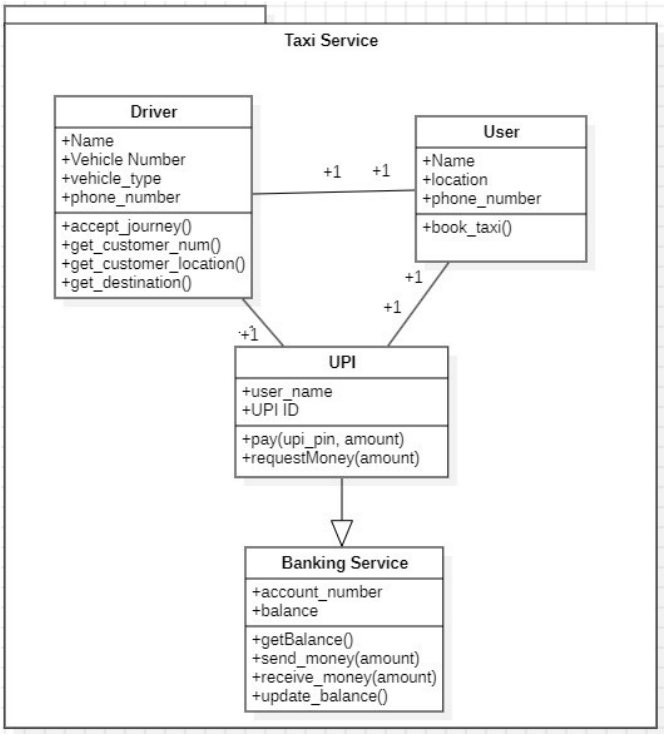
UML DIAGRAMS

1. TAXI SERVICE APPLICATION

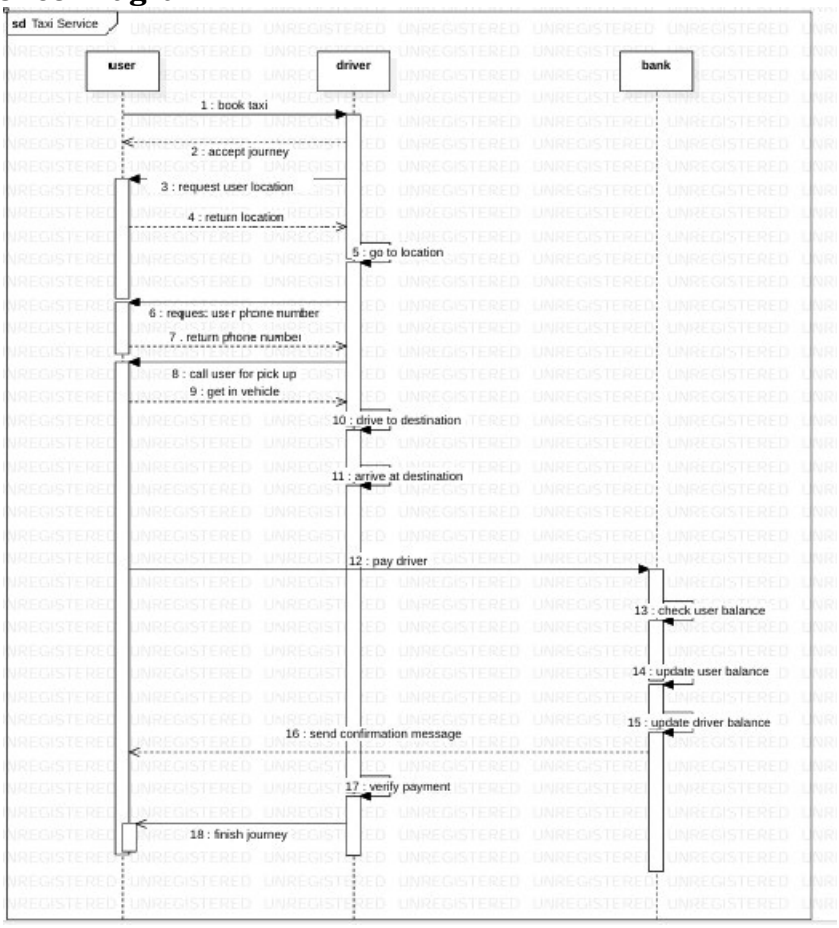
1.a) Use Case Diagram:



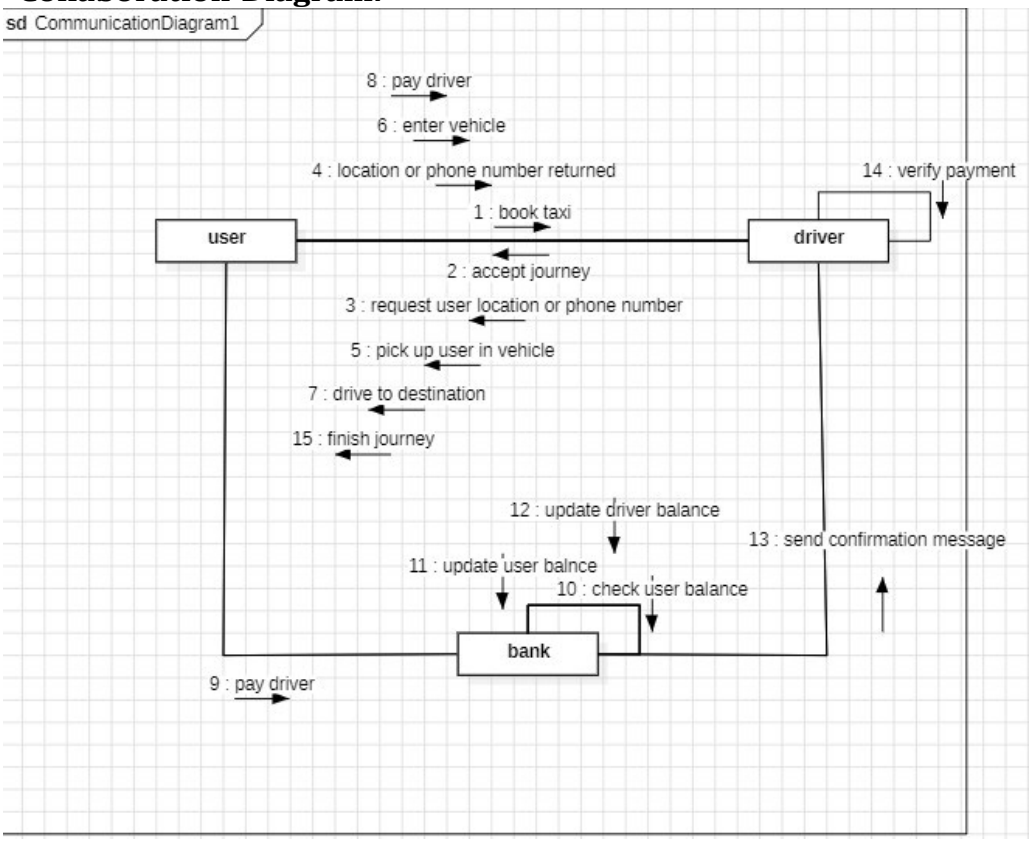
1.b) Class Diagram:



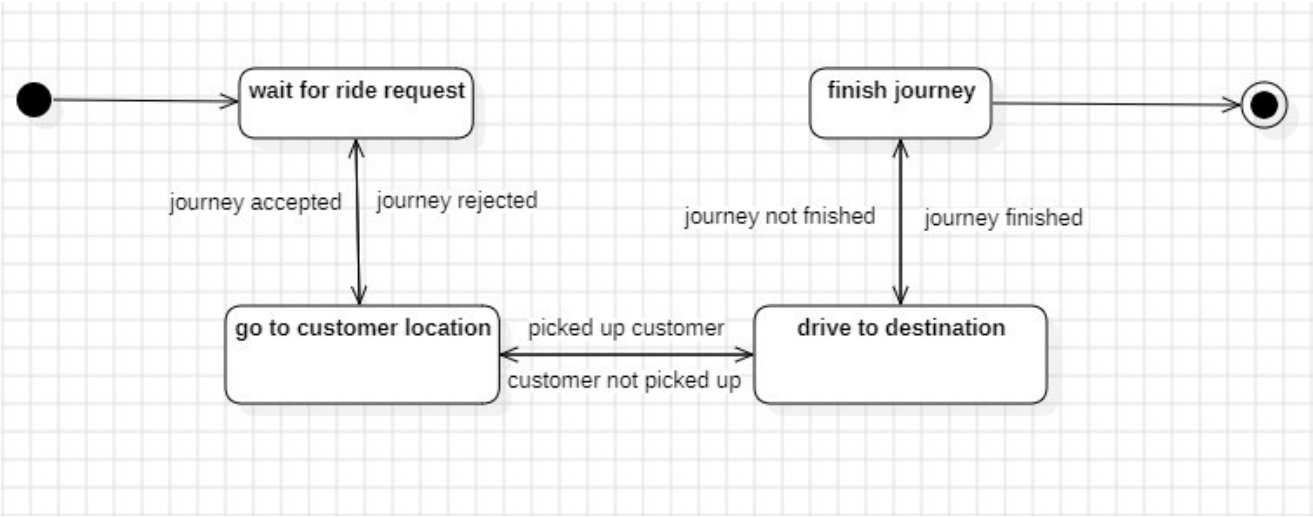
1.c) Sequence Diagram:



1.d) Collaboration Diagram:

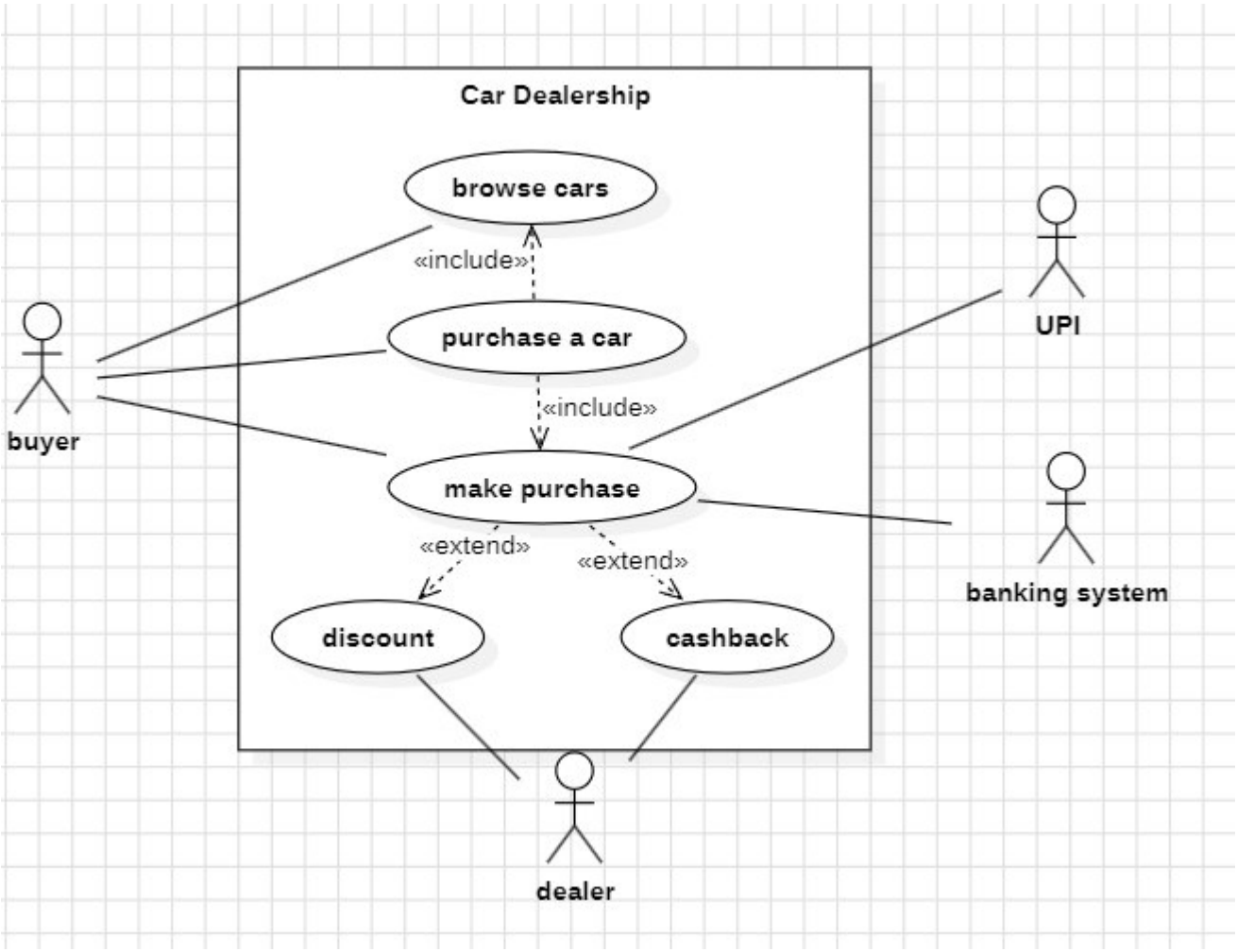


1.e) State-Activity Diagram:

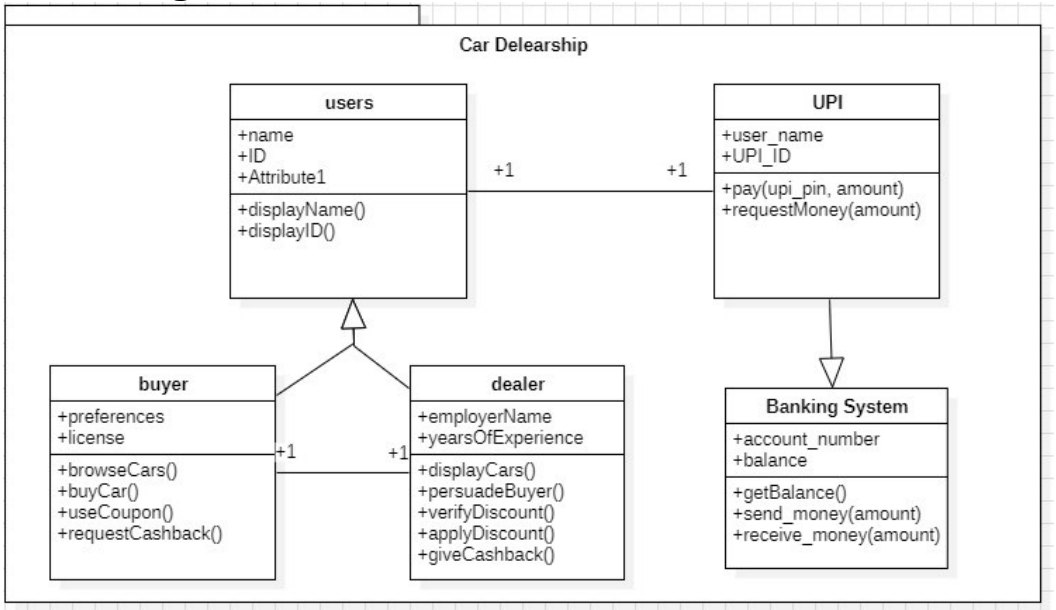


2. CAR DEALERSHIP APPLICATION

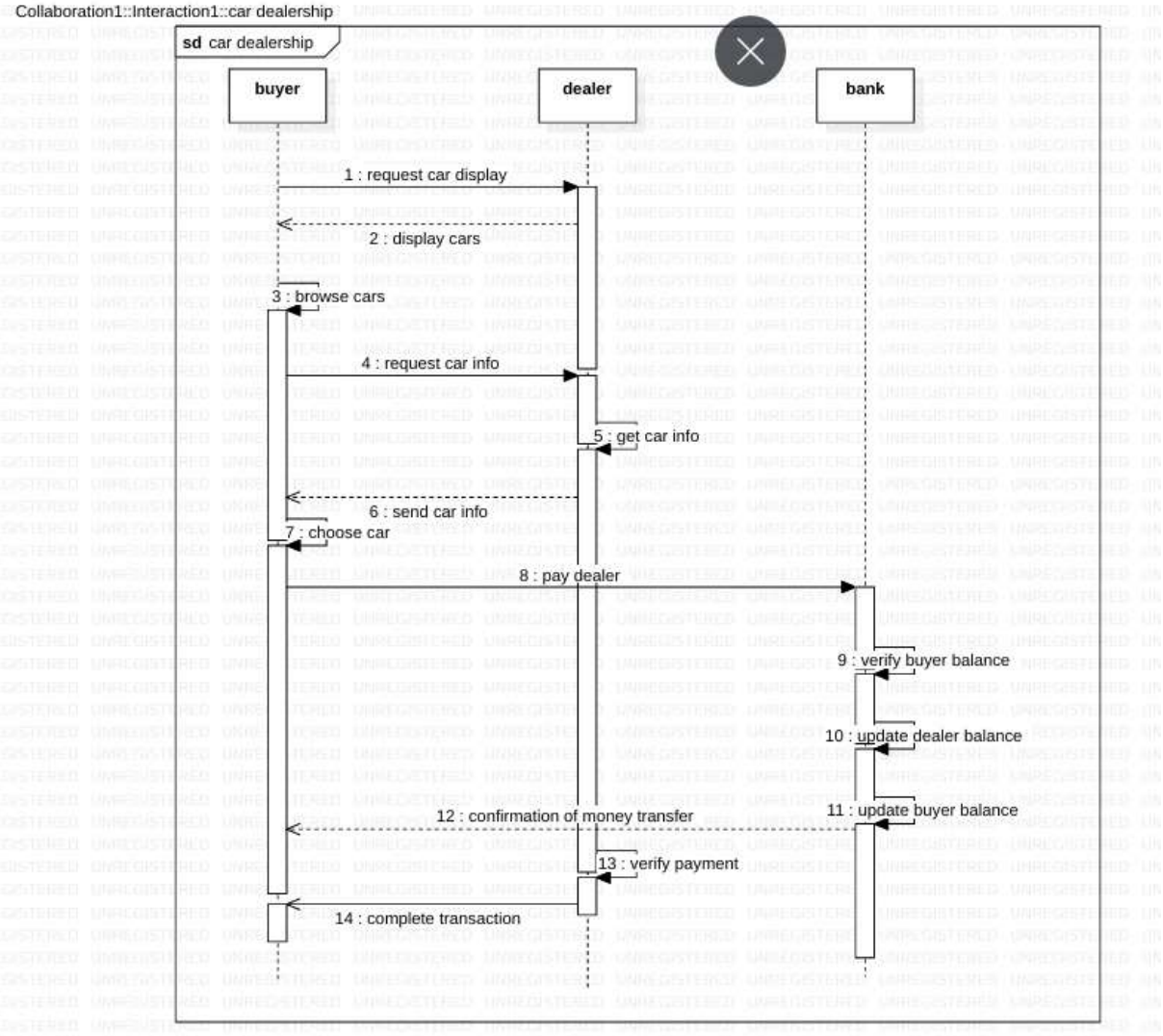
2.a) Use Case Diagram:



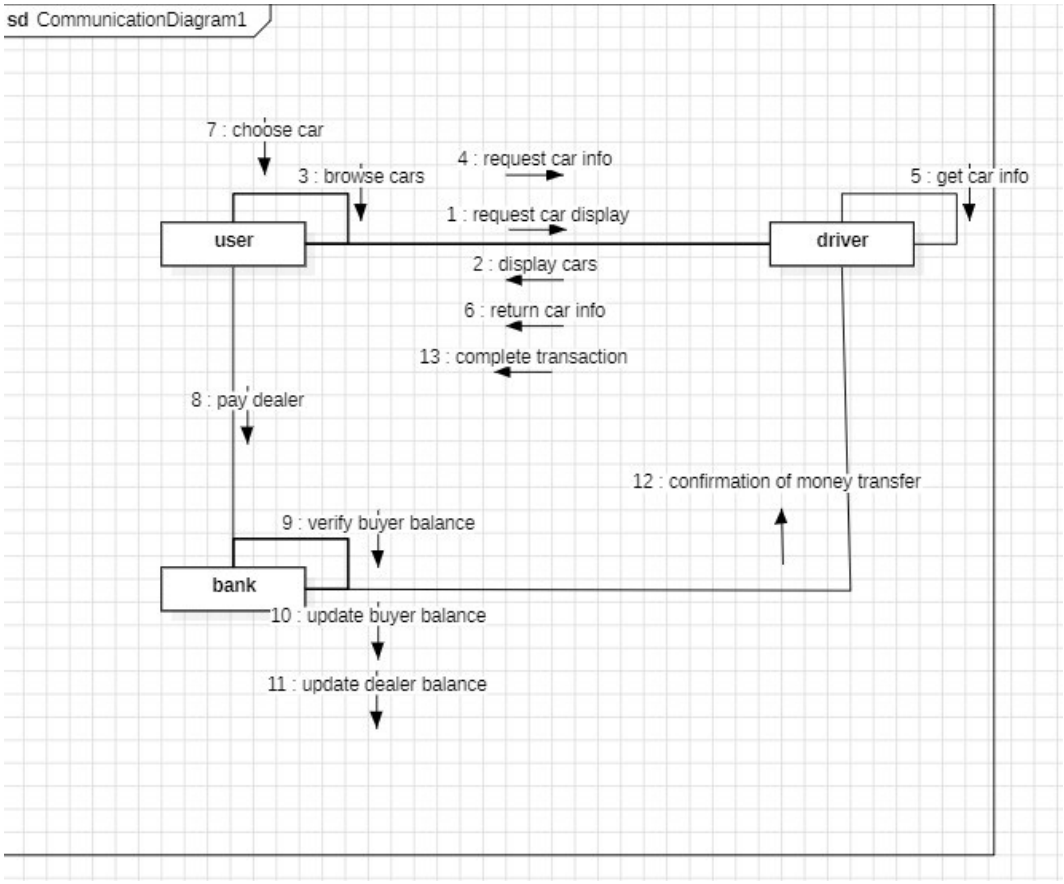
2.b) Class Diagram:



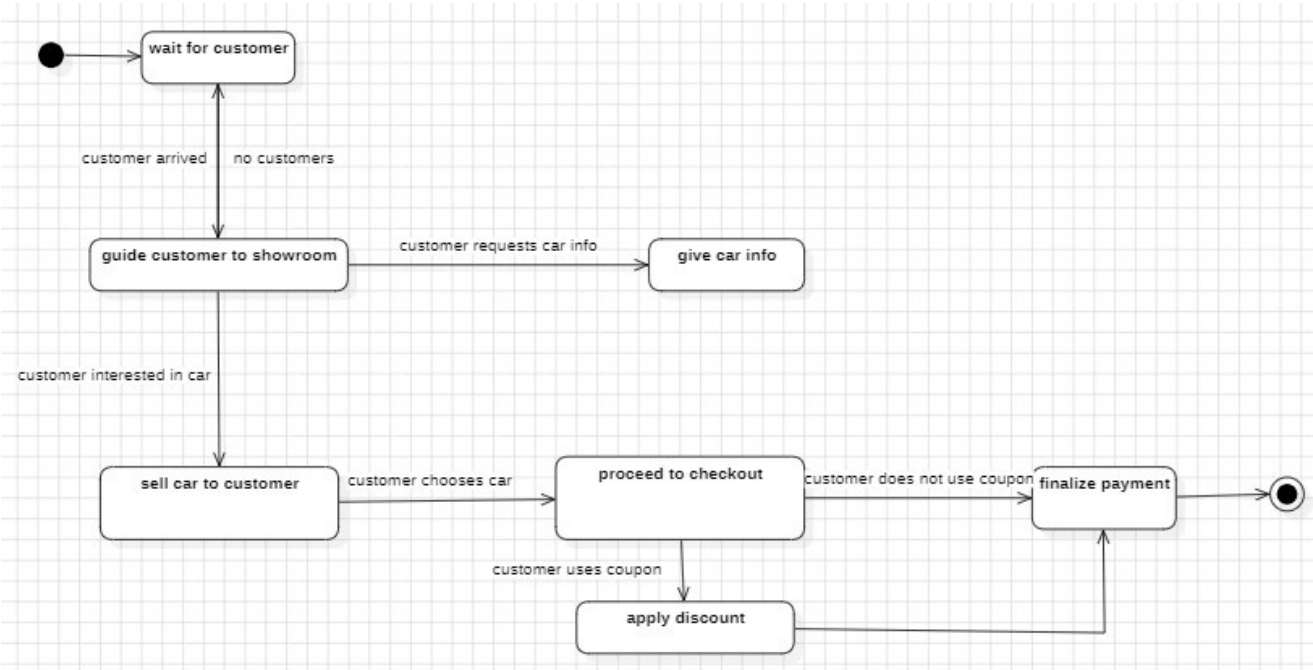
2.c) Sequence Diagram:



2.d) Collaboration Diagram:



2.e) State-Activity Diagram:



3. Basic Java Programs

3.a) CalculateArea of Rectangle, Circle, Triangle:

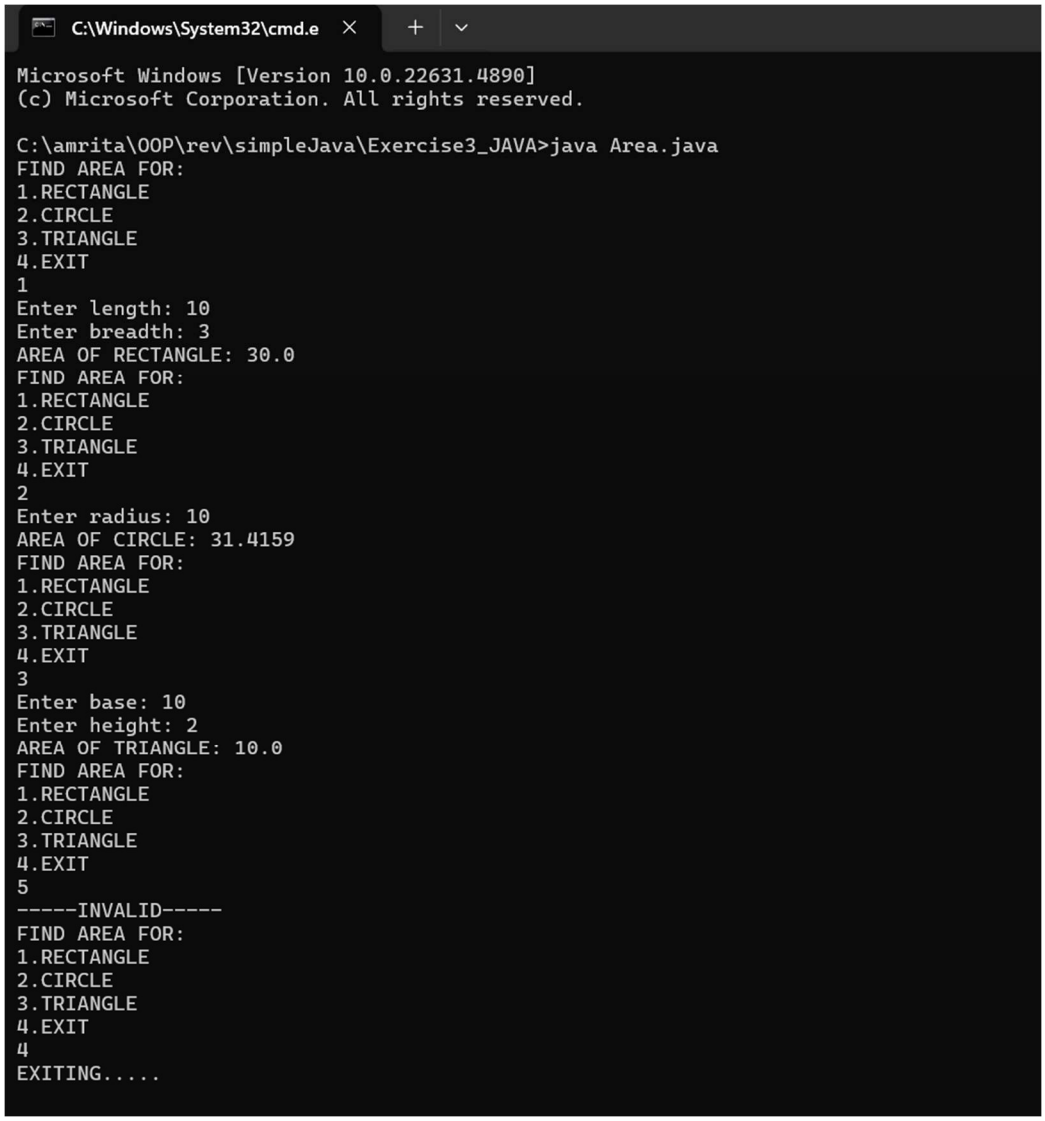
Code:

```
import java.util.Scanner;


class Area{
    public static void main(String[] args){
        int choice=0;
        double l;
        double b;
        double h;
        double r;
        findArea find=new findArea();
        Scanner scan=new Scanner(System.in);
        while(choice!=4){
            System.out.println("FIND AREA FOR:");
            System.out.println("1.RECTANGLE");
            System.out.println("2.CIRCLE");
            System.out.println("3.TRIANGLE");
            System.out.println("4.EXIT");
            choice=scan.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter length: ");
                    l=scan.nextDouble();
                    System.out.print("Enter breadth: ");
                    b=scan.nextDouble();
                    find.rectArea(l, b);
```

```
                    break;
                case 2:
                    System.out.print("Enter radius: ");
                    r=scan.nextDouble();
                    find.CircleArea(r);
                    break;
                case 3:
                    System.out.print("Enter base: ");
                    b=scan.nextDouble();
                    System.out.print("Enter height: ");
                    h=scan.nextDouble();
                    find.triArea(b, h);
                    break;
                case 4:
                    System.out.println("EXITING.....");
                    break;
                default:
                    System.out.println("-----INVALID-----");
            }

        }

    }

}

class findArea{
    public void rectArea(double l, double b){
        double c=l*b;
        System.out.println("AREA OF RECTANGLE: "+c );
```

```
    }
    public void CircleArea(double r){
        double c=3.14159*r;
        System.out.println("AREA OF CIRCLE: "+c );
    }
    public void triArea(double b, double h){
        double c=0.5*b*h;
        System.out.println("AREA OF TRIANGLE: "+c );
    }

}
```

Output:



3.b) Basic Math Operations:

Code:

```
import java.util.Scanner;
class calculate{
    public static void main(String[] args){
        int choice=0;
        Calc calc=new Calc();
        Scanner scan=new Scanner(System.in);
        System.out.print("Enter num1: ");
        int num1=scan.nextInt();
        System.out.print("Enter num2: ");
        int num2=scan.nextInt();
        while(choice!=5){
            System.out.println("PICK AN OPERATION:");
            System.out.println("1.Addition");
            System.out.println("2.Subtraction");
            System.out.println("3.Multiplication");
            System.out.println("4.Division");
            System.out.println("5.EXIT");
            choice=scan.nextInt();
            switch (choice) {
                case 1:
                    calc.add(num1, num2);
                    break;
                case 2:
                    calc.subtract(num1, num2);
                    break;
                case 3:
                    calc.multiply(num1, num2);
```

```
                    break;
                case 4:
                    calc.divide(num1, num2);
                    break;
                case 5:
                    System.out.println("EXITING.....");
                    break;
                default:
                    System.out.println("-----INVALID-----");
            }

        }

    }

}

class Calc{
    public void add(int a, int b){
        int c=a+b;
        System.out.println("ANSWER: "+a + "+"+b+"="+c+"\n");
    }
    public void subtract(int a, int b){
        int c=a-b;
        System.out.println("\nANSWER: "+a + "-" +b + "=" + c+"\n");
    }
    public void multiply(int a , int b){
        int c=a*b;
        System.out.println("\nANSWER: "+a+"x"+b+"="+c+"\n");
    }
    public void divide(double a, double b){
```

```
            double c=a/b;

            System.out.println("\nANSWER: "+a+"/"+b+"="+c+"\n");

        }

    }
```

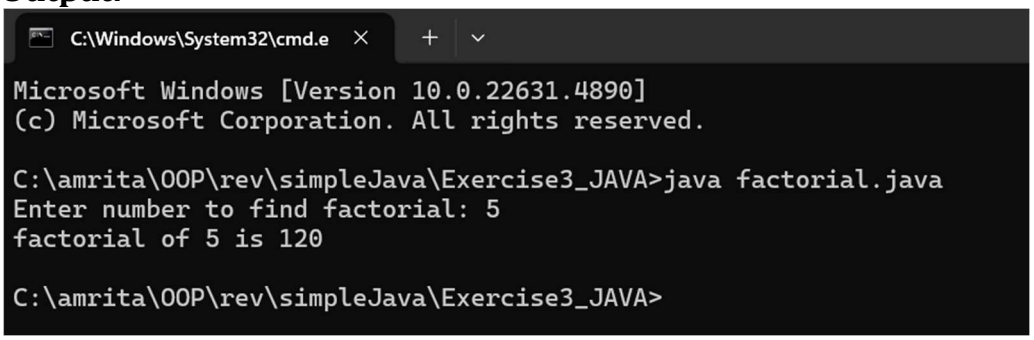
Output:

3.c) Factorial:

Code:

```
import java.util.Scanner;

class factorial{

    public static void main(String[] args){

        Scanner scan=new Scanner(System.in);

        calculate obj=new calculate();

        System.out.print("Enter number to find factorial: ");

        long num=scan.nextLong();

        long fact=obj.fac(num);

        System.out.println("factorial of "+num+" is "+fact);









    }

}

class calculate{

    public long fac(long num){

        long fac=1;

        for(long i=1; i<=num;i++){

                fac=fac*i;

            }

        return fac;

    }

}
```

Output:

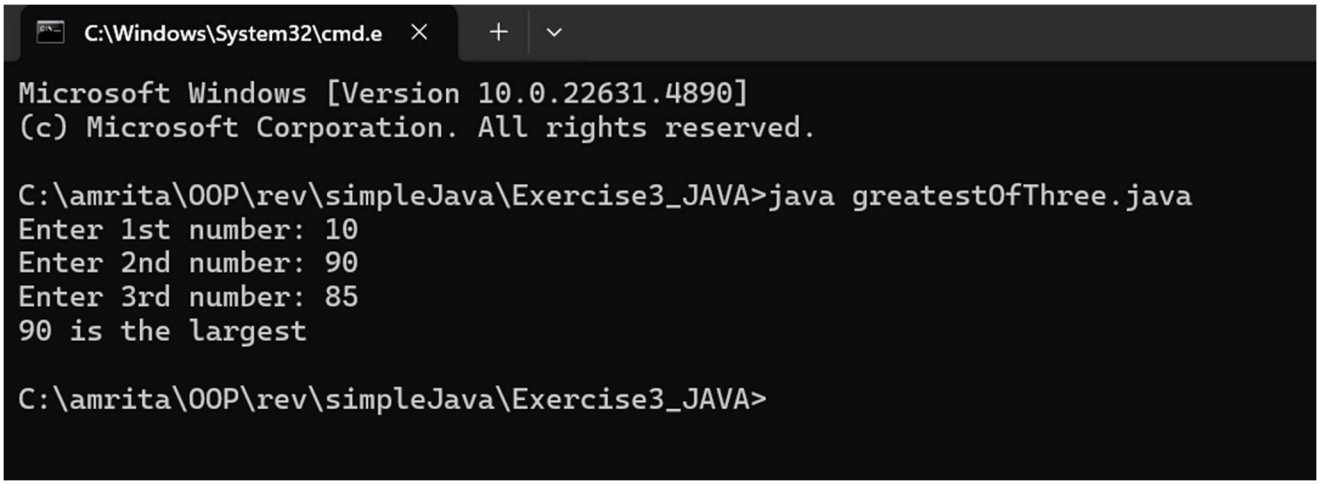


3.d) Finding the Greatest of Three Numbers:

Code:

```
import java.util.Scanner;
class greatestOfThree{
    public static void main(String[] args){
        Scanner scan=new Scanner(System.in);
        System.out.print("Enter 1st number: ");
        int a=scan.nextInt();
        System.out.print("Enter 2nd number: ");
        int b=scan.nextInt();
        System.out.print("Enter 3rd number: ");
        int c=scan.nextInt();
     if (a>b){
            if(a>c){
                System.out.println(a+" is the largest");
            }
        }
        if (b>c){
            if (b>a){
                System.out.println(b+" is the largest");
            }
        }
        if (c>a){
            if(c>b){
                System.out.println(c+" is the largest");
            }
        }
    }
}
```

Output;


C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>java greatestOfThree.java
Enter 1st number: 10
Enter 2nd number: 90
Enter 3rd number: 85
90 is the largest
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>

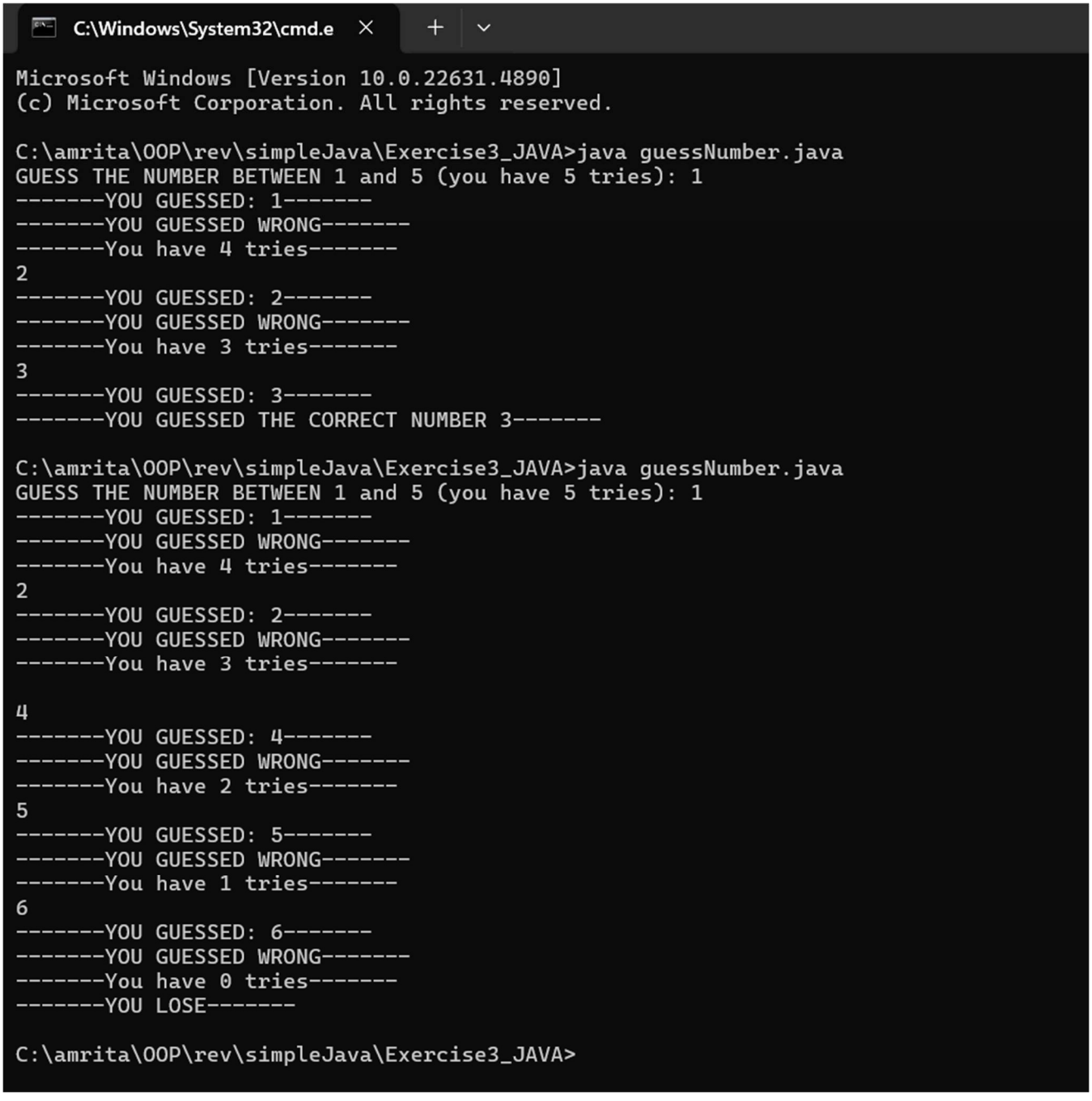
3.e) Number Guessing Game:

Code:

```
import java.util.Scanner;

class guessNumber{

    public static void main(String[] args){

        int choice=0;

        int tries=5;

        Scanner scan=new Scanner(System.in);

        System.out.print("GUESS THE NUMBER BETWEEN 1 and 5 (you have 5
tries): ");

        while(tries>0){

            choice=scan.nextInt();

            System.out.println("-------YOU GUESSED: "+choice+"-------");

            if (choice==3){

                System.out.println("-------YOU GUESSED THE CORRECT NUMBER
3-------");

                break;

            }

            else{

                System.out.println("-------YOU GUESSED WRONG-------");

                tries=tries-1;

                System.out.println("-------You have "+tries+" tries-------
");

            }

            if (tries==0){

                System.out.println("-------YOU LOSE-------");

                break;

            }

        }

    }
```

```
}
```

Output:

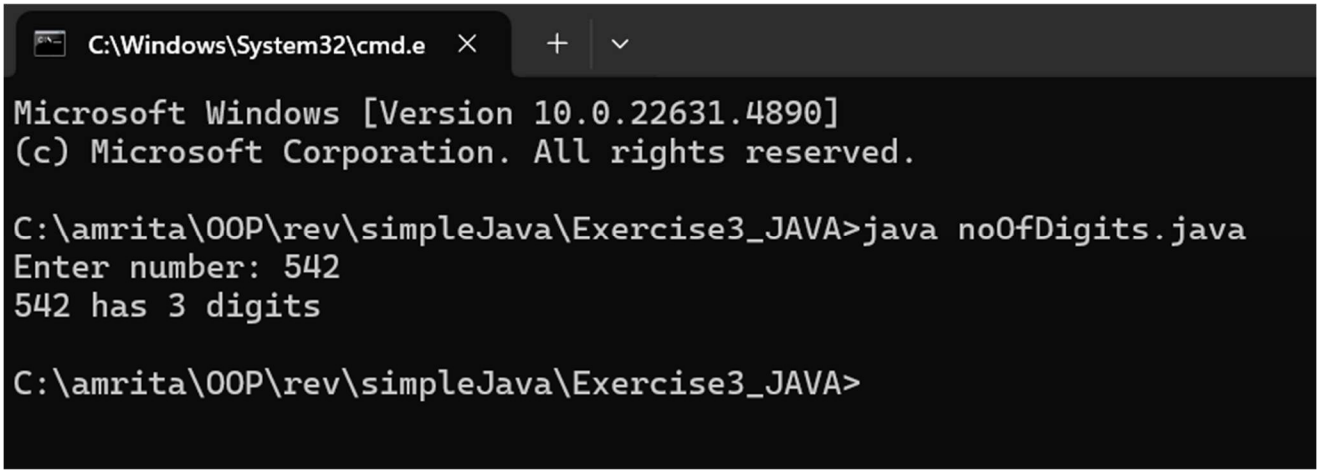

C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>java guessNumber.java
GUESS THE NUMBER BETWEEN 1 and 5 (you have 5 tries): 1
-------YOU GUESSED: 1-------
-------YOU GUESSED WRONG-------
-------You have 4 tries-------
2
-------YOU GUESSED: 2-------
-------YOU GUESSED WRONG-------
-------You have 3 tries-------
3
-------YOU GUESSED: 3-------
-------YOU GUESSED THE CORRECT NUMBER 3-------
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>java guessNumber.java
GUESS THE NUMBER BETWEEN 1 and 5 (you have 5 tries): 1
-------YOU GUESSED: 1-------
-------YOU GUESSED WRONG-------
-------You have 4 tries-------
2
-------YOU GUESSED: 2-------
-------YOU GUESSED WRONG-------
-------You have 3 tries-------
4
-------YOU GUESSED: 4-------
-------YOU GUESSED WRONG-------
-------You have 2 tries-------
5
-------YOU GUESSED: 5-------
-------YOU GUESSED WRONG-------
-------You have 1 tries-------
6
-------YOU GUESSED: 6-------
-------YOU GUESSED WRONG-------
-------You have 0 tries-------
-------YOU LOSE-------
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>

3.f) Find Number of Digits in a Number:

Code:

```
public class NumberPattern {
    import java.util.Scanner;

    class noOfDigits{

        public static void main(String[]
        args){

            Scanner scan=new
        Scanner(System.in);

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

            int num=scan.nextInt();

            int tnum=num;

            int digits=0;

            while (tnum>0){

                int digit=tnum%10;

                tnum=tnum/10;

                digits+=1;

            }

            System.out.println(num+" has
        "+digits+" digits");

        }

    }
```

Output:


C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>java noOfDigits.java
Enter number: 542
542 has 3 digits
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>

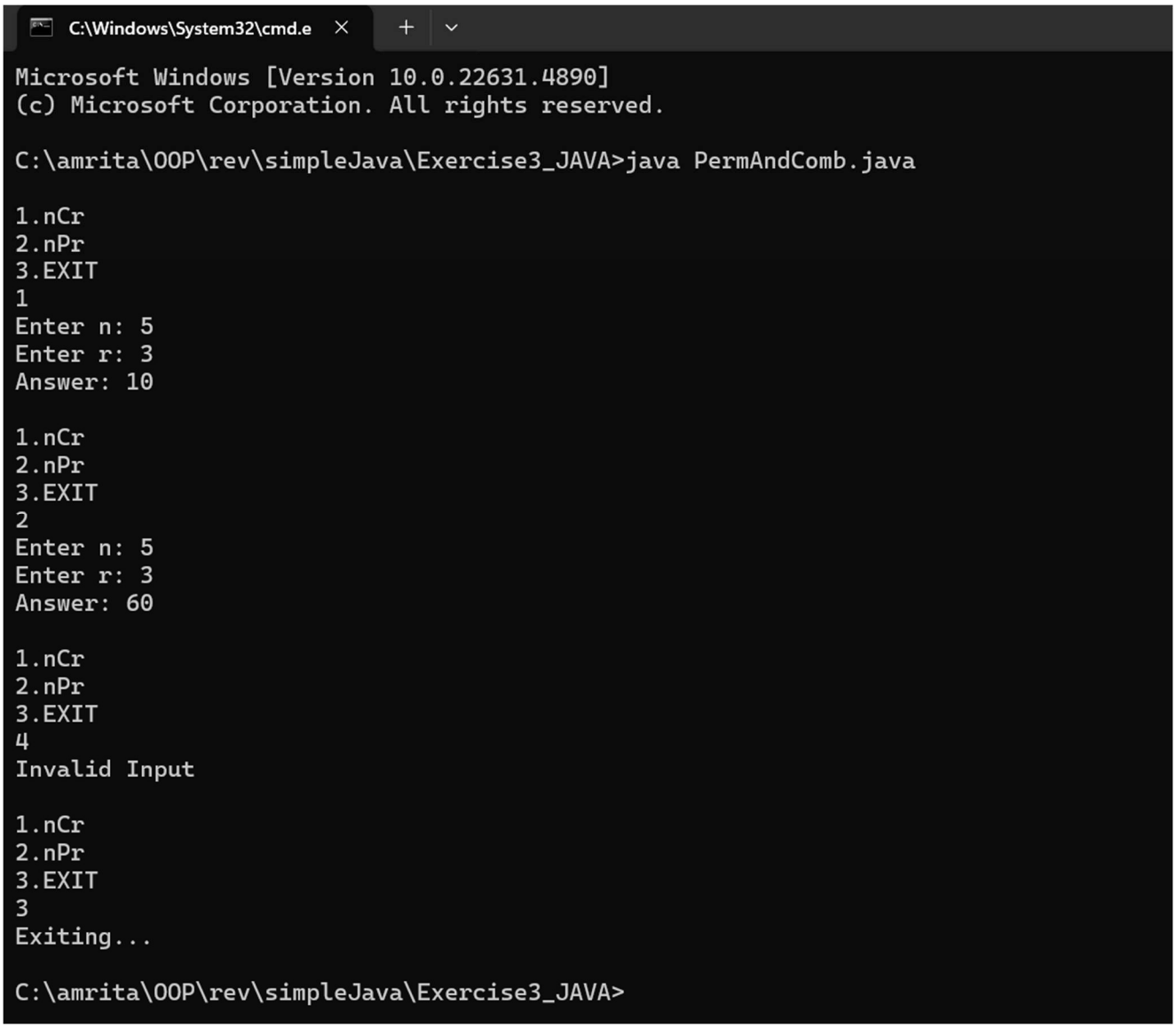
3.g) Permutaions and Combinations:

Code:

```
import java.util.Scanner;
class PermAndComb{
    public static void main(String[] args){
        int ch=0;
        while(ch!=3){
            Operations obj=new Operations();
            Scanner scan=new Scanner(System.in);
            System.out.println("\n1.nCr");
            System.out.println("2.nPr");
            System.out.println("3.EXIT");
            ch=scan.nextInt();
            if(ch==1){
                System.out.print("Enter n: ");
                int n=scan.nextInt();
                System.out.print("Enter r: ");
                int r=scan.nextInt();
                System.out.println("Answer: "+obj.C(n,r));
            }
            else if(ch==2){
                System.out.print("Enter n: ");
                int n=scan.nextInt();
                System.out.print("Enter r: ");
                int r=scan.nextInt();
                System.out.println("Answer: "+obj.P(n,r));
            }
            else if(ch==3){
                System.out.println("Exiting...");
                break;
            }
```

```
            else{
                System.out.println("Invalid Input");
            }
        }
    }
}

class factorial{
    public long factorial(long n){
        long fac=1;
        for(int i=2;i<=n;i++){
            fac=fac*i;
        }
        return fac;
    }
}

class Operations extends factorial{
    public long P(int n, int r){
        return factorial(n)/factorial(n-r);
    }
    public long C(int n, int r){
        return factorial(n)/((factorial(n-r))*factorial(r));
    }
}
```

Output:

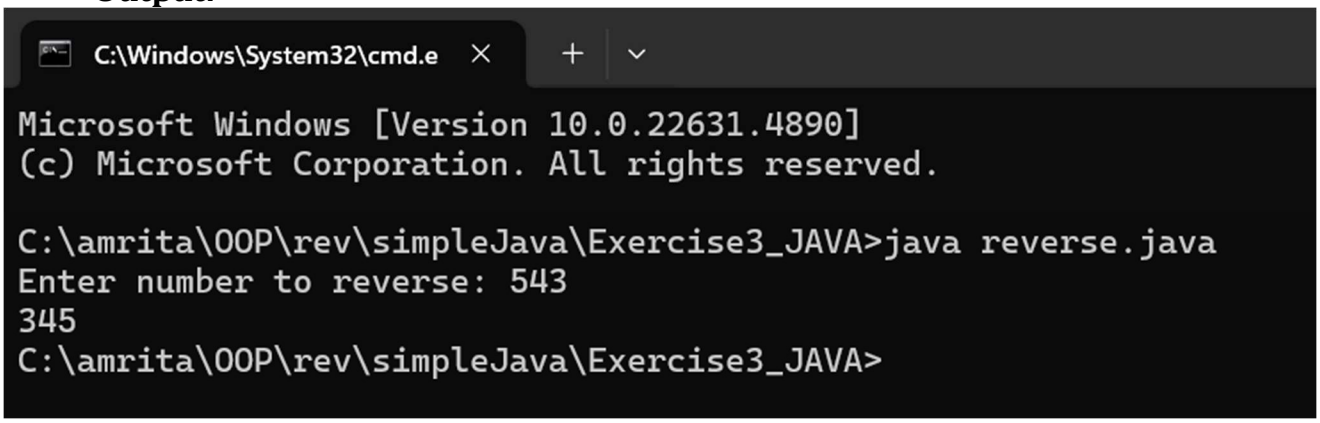

C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>java PermAndComb.java
1.nCr
2.nPr
3.EXIT
1
Enter n: 5
Enter r: 3
Answer: 10
1.nCr
2.nPr
3.EXIT
2
Enter n: 5
Enter r: 3
Answer: 60
1.nCr
2.nPr
3.EXIT
4
Invalid Input
1.nCr
2.nPr
3.EXIT
3
Exiting...
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>

3.h) Reversing an Integer:

Code:

```
import java.util.Scanner;
class reverse{
    public static void main(String[] args){
        Scanner scan=new Scanner(System.in);
        System.out.print("Enter number to reverse: ");
        int num=scan.nextInt();
        int rev=0;
        int digit=0;
        while(num>0){
            digit=num%10;
            rev=rev*10+digit;
            num=num/10;
            }
        System.out.print(rev);
    }
}
```

Output:

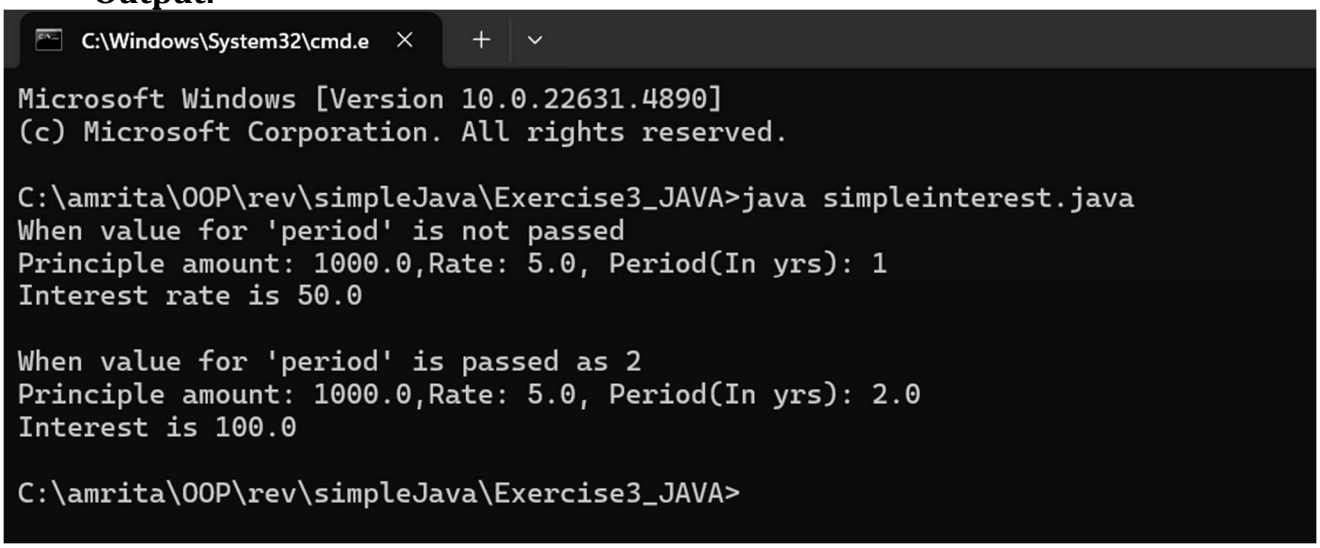


3.i) Calculate Simple Interest:

Code:

```
class simpleInterest{
    public static void main(String[] args){
    Calc obj=new Calc();
    System.out.println("When value for 'period' is not passed");
    obj.s_interest(1000,5);
    System.out.println("\nWhen value for 'period' is passed as 2");
    obj.s_interest(1000,5,2);

    }
}

class Calc{
    public static void s_interest(double principle, double rate, double
period){
        double interest=principle*(rate/100.0)*period;
        System.out.println("Principle amount: " + principle +",Rate: "+
rate +", Period(In yrs): " + period );
        System.out.println("Interest is " + interest);
    }
    public static void s_interest(double principle, double rate){
        double interest=principle*(rate/100.00)*1.00;
        System.out.println("Principle amount: "+principle+",Rate: "+rate+",
Period(In yrs): "+ 1 );
        System.out.println("Interest rate is "+interest);
    }

}
```

Output:

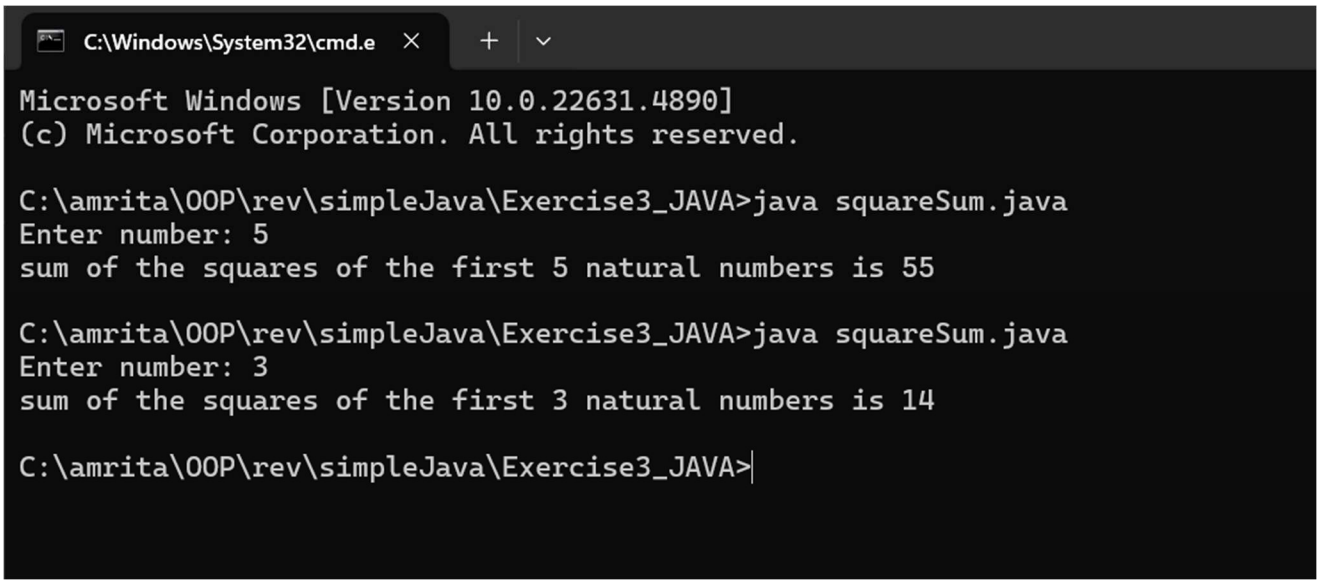

C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>java simpleinterest.java
When value for 'period' is not passed
Principle amount: 1000.0,Rate: 5.0, Period(In yrs): 1
Interest rate is 50.0
When value for 'period' is passed as 2
Principle amount: 1000.0,Rate: 5.0, Period(In yrs): 2.0
Interest is 100.0
C:\amrita\OOP\rev\simpleJava\Exercise3_JAVA>

3.j) Sum of Squares of First n Natural Numbers:

Code:

```
import java.util.Scanner;

class sqaureSum{

    public static void main(String[] args){

        Scanner scan=new Scanner(System.in);

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

        int n=scan.nextInt();

        int sum=0;

        for(int count=0;count<=n;count++){

            sum+=count*count;}

        System.out.println("sum of the squares of the first "+n+" natural
numbers is "+sum);

    }

}
```

Output:



INHERITANCE
4. Single Inheritance Programs

4 a) Demonstrating single inheritance by implementing cameras.

Code:

```
class Cam{
     public static void main(String[] args){
          DigitalCamera camOBJ=new DigitalCamera();
          camOBJ.setFL(0.3);
          camOBJ.setMan("Canon");
          camOBJ.turnOn();
          camOBJ.digitalZoom(1.5);
          camOBJ.takePhoto();
          camOBJ.turnOff();
          camOBJ.recharge();
     }
}


class Camera{
     double focalLength;
     String manufacturer;

     public void setFL(double fl){
          focalLength=fl;
          System.out.println("focal length set as "+fl);
     }
     public void setMan(String man){
          manufacturer=man;
          System.out.println("Manufacturer set as "+man);
     }
     public void takePhoto(){
          System.out.println("TAKING PHOTO.....");
     }
}

class DigitalCamera extends Camera{
     double batteryLevel;
     double digiZoom;
     public void turnOn(){
          System.out.println("turning on camera....");
```

```
    }
    public void turnOff(){
        System.out.println("turning off camera....");
    }
    public void recharge(){
        System.out.println("Recharging...");
    }
    public void digitalZoom(double zoomVal){
        digiZoom=zoomVal;
        System.out.println("Digital Zoom set at "+zoomVal);
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\SingleInhertiance>java cam.java
focal length set as 0.3
Manufacturer set as Canon
turning on camera....
Digital Zoom set at 1.5
TAKING PHOTO.....
turning off camera....
Recharging...
```

4 b) Demonstrating single inheritance using vehicles.

Code:

```
class vehicles{
	public static void main(String[] args){
		Car carOBJ=new Car();
		carOBJ.set("electric","tesla",4,4);
		carOBJ.recharge();
		carOBJ.start();
		carOBJ.stop();
	}
}

class Vehicle{
	String type;
	String manufacturer;
	int no_of_wheels;
	int max_no_of_passengers;
	public void set(String t, String man, int now, int nop){
		type=t;
		manufacturer=man;
		no_of_wheels=now;
		max_no_of_passengers=nop;
	}
	abstract public void start();
	abstract public  void stop();
}

class Car extends Vehicle{
	public void start(){
		System.out.println("STARTING THE CAR...");
	}
	public void stop(){
		System.out.println("STOPPING THE CAR...");
	}
	public void recharge(){
		System.out.println("REFUELLING THE CAR.....");
	}
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\SingleInhertiance>java vehicles.java
REFUELLING THE CAR.....
STARTING THE CAR...
STOPPING THE CAR...
```

5. Multilevel Inheritance Programs

5 a)Banking System

Code:

```
class banking {
    public static void main(String[] args) {
        SavingsAccount sa = new SavingsAccount();

        sa.displayBankName();
        sa.displayAccountDetails();
        sa.calculateInterest(2);
    }
}

class Bank {
    String bankName = "ABC Bank";

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

class Account extends Bank {
    String accHolder = "Vaish";
    double balance = 10000;

    void displayAccountDetails() {
        System.out.println("Account Holder: " + accHolder);
        System.out.println("Balance: Rs" + balance);
    }
}

class SavingsAccount extends Account {
    double interestRate = 4.5;
    void calculateInterest(int years) {
        double interest = (balance * interestRate * years) / 100;
        System.out.println("Interest for " + years + " years: Rs" + interest);
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\MultilevelInheritance>java banking.java
Bank Name: ABC Bank
Account Holder: Vaish
Balance: Rs10000.0
Interest for 2 years: Rs900.0
```

5 b)Implementing Shapes

Code:

```
public class shapeMulti {
    public static void main(String[] args) {
        Box box = new Box();

        box.printShape();
        box.calculateArea();
        box.calculateVolume();
    }
}

class Shape {
    void printShape() {
        System.out.println("This is a geometric shape.");
    }
}

class Rectangle extends Shape {
    int length = 10;
    int width = 5;

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

class Box extends Rectangle {
    int height = 4;

    void calculateVolume() {
        int volume = length * width * height;
        System.out.println("Volume of Box: " + volume);
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\MultilevelInheritance>java shapeMulti.java
This is a geometric shape.
Area of Rectangle: 50
Volume of Box: 200

C:\amrita\OOP\Exercise4_JavaConcepts\MultilevelInheritance>
```

6. Hierarchical Inheritance Programs

6 a) Implementing furniture to show hierarchical inheritance.

Code:

```
class Furniture{
	public static void main(String[] args){
		HeightAdjustTable tableOBJ=new HeightAdjustTable(5.0,20);
		FoldableTable foldable=new FoldableTable(1.2);
		foldable.Table(15.0,10,10,"metal");
		foldable.unfold();
		foldable.fold();
		tableOBJ.Table(20,50,10,"wood");
		tableOBJ.changeHeight(10);
		tableOBJ.changeHeight(50);
	}
}


class Table{
	double length;
	double breadth;
	double standHeight;
	String material;
	public void Table(double len,double bred,double sh,String mat){
		this.length=len;
		this.breadth=bred;
		this.standHeight=sh;
		this.material=mat;
	}
}

class FoldableTable extends Table{
	Double foldedHeight;
	public FoldableTable(Double fh){
		this.foldedHeight=fh;
	}
	public void fold(){
		System.out.println("FOLDING TABLE - Now you have more space");
	}
	public void unfold(){
		System.out.println("UNFOLDING TABLE - Table is ready to be used");
	}
}

class HeightAdjustTable extends Table{
	double maxHeight;
	double minHeight;
	double curHeight=standHeight;
	public HeightAdjustTable(double minH,double  maxH){
		this.maxHeight=maxH;
		this.minHeight=minH;
	}
	public void changeHeight(double height){
		if(height>maxHeight){
```

```
                System.out.println("MAXIMUM HEIGHT IS "+maxHeight);
            }
            else if(height<minHeight){
                System.out.println("MINIMUM HEIGHT IS "+minHeight);
            }
            else{
                System.out.println("adjusting height to "+height);
                curHeight=height;
            }
        }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\HierarchicalInheritance>java furniture.java
UNFOLDING TABLE - Table is ready to be used
FOLDING TABLE - Now you have more space
adjusting height to 10.0
MAXIMUM HEIGHT IS 20.0
```

6 b)Implementing Bikes

Code:

```
class bikes{
    public static void main(String[] args){
        motorbike bike=new motorbike("gas",500.0,1000.0);
        cycle cycleOBJ=new cycle();
```

```
		cycleOBJ.startPedalling();
		cycleOBJ.useBrakes(true);
		bike.startBike();
		bike.useBrakes(true);
		bike.checkFuel();
	}
}

class two_wheeler{
	int no_of_wheels=2;
	float speed;
	boolean brakes;

	public void useBrakes(boolean val){
		brakes=val;
		if(val){
			System.out.println("Engaging Brakes");
		}
		else{
			System.out.println("Disengaging Brakes");
		}
	}
}

class cycle extends two_wheeler{
	String type;
	public void startPedalling(){
		System.out.println("Pedalling....");
	}
}

class motorbike extends two_wheeler{
	String type;
	double fuelCapacity;
	double fuelLevel;
	public motorbike(String type, double fuelCapacity, double fuelLevel){
		this.type=type;
		this.fuelCapacity=fuelCapacity;
		this.fuelLevel=fuelLevel;
	}
	public void startBike(){
		System.out.println("starting Bike....");
	}
	public void checkFuel(){
		System.out.println("Amount of fuel: "+fuelLevel+" Litres");
	}

}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\HierarchicalInheritance>java bikes.java
Pedalling....
Engaging Brakes
starting Bike....
Engaging Brakes
Amount of fuel: 1000.0 Litres

C:\amrita\OOP\Exercise4_JavaConcepts\HierarchicalInheritance>
```

7. Hybrid Inheritance Programs

7 a) Implementing Devices like computer, laptop and tablet.
Code:

```
public class DeviceHybrid {
    public static void main(String[] args) {
        System.out.println("Laptop");
        Laptop dell = new Laptop();
        dell.powerOn();
        dell.runOS();
        dell.fold();

        System.out.println("\nTablet");
        Tablet ipad = new Tablet();
        ipad.powerOn();
        ipad.touchSupport();
    }
}

class Device {
    void powerOn() {
        System.out.println("Device is powered on...");
    }
}

class Computer extends Device {
    void runOS() {
        System.out.println("Computer is running its OS...");
    }
}

class Laptop extends Computer {
    void fold() {
        System.out.println("Laptop is foldable and portable...");
    }
}

class Tablet extends Device {
    void touchSupport() {
        System.out.println("Tablet supports touch input....");
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\HybridInheritance>java DeviceHybrid.java
Laptop
Device is powered on...
Computer is running its OS...
Laptop is foldable and portable...

Tablet
Device is powered on...
Tablet supports touch input....

C:\amrita\OOP\Exercise4_JavaConcepts\HybridInheritance>
```

7 b) Implementing Employees

Code:

```
public class empHybrid {
    public static void main(String[] args) {
        System.out.println("Manager Details");
        Manager m = new Manager();
        m.displayPerson();
        m.displayEmployee();
        m.showRole();

        System.out.println("\nEngineer Details");
        Engineer e = new Engineer();
        e.displayPerson();
        e.displayEmployee();
        e.showRole();
    }
}

class Person {
    String name = "Vaish";

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

class Employee extends Person {
    int empId = 101;

    void displayEmployee() {
        System.out.println("Employee ID: " + empId);
    }
}

class Manager extends Employee {
    void showRole() {
        System.out.println("Role: Manager");
    }
}

class Engineer extends Employee {
    void showRole() {
        System.out.println("Role: Engineer");
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\HybridInheritance>java empHybrid.java
Manager Details
Name: Vaish
Employee ID: 101
Role: Manager

Engineer Details
Name: Vaish
Employee ID: 101
Role: Engineer

C:\amrita\OOP\Exercise4_JavaConcepts\HybridInheritance>
```

Polymorphism
8. Constructor Programs

8 a) Implementing Students

Code:

```
public class studentConstructor{
    public static void main(String[] args) {
        Student studObj = new Student("Vaish", 18);
        studObj.displayInfo();
    }
}

class Student {
    String name;
    int age;

    Student(String studentName, int studentAge) {
        name = studentName;
        age = studentAge;
        System.out.println("Student created: " + name + ", Age: " + age);
    }

    void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism>java studentConstructor.java
Student created: Vaish, Age: 18
Name: Vaish
Age: 18

C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism>
```

9.Constructor Overloading Programs

9 a)Book Classification

Code:

```
public class bookConOverload {
    public static void main(String[] args) {
        Book b1 = new Book("The Invisible Man");
        Book b2 = new Book("Harry Potter", "J K Rowling");

        b1.display();
        b2.display();
    }
}

class Book {
    String title;
    String author;

    Book(String title) {
        this.title = title;
        this.author = "Unknown";
    }

    Book(String title, String author) {
        this.title = title;
        this.author = author;
    }

    void display() {
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism>java bookConOverload.java
Title: The Invisible Man
Author: Unknown
Title: Harry Potter
Author: J K Rowling

C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism>
```

10.Method Overloading Programs

10 a) Implementing simple search engine queries

Code:

```
import java.util.Scanner;
class search{
	public static void main(String args[]){
		Scanner scan=new Scanner(System.in);
		Browser obj=new Browser();

		System.out.print("Enter search query: ");
		String sq=scan.nextLine();

		System.out.print("would you like to use your desired search
engine?(y/n) ");
		String ans=scan.nextLine();
		if(ans.compareTo("y")==0){
			System.out.print("Enter search engine name: ");
			String engine=scan.nextLine();
			obj.search(sq,engine);
		}
		else{
			obj.search(sq);
		}
	}
}


class Browser{
	String searchEngine;
	public void search(String query){
		System.out.println("Search Engine: google");
		System.out.println("Searching for: "+query);
	}
	public void search(String query, String engine){
		searchEngine=engine;
		System.out.println("Search Engine: "+engine);
		System.out.println("Searching for: "+query);
	}
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism\methodOverload>java search.java
Enter search query: https://www.youtube.com
would you like to use your desired search engine?(y/n) y
Enter search engine name: duckduckgo
Search Engine: duckduckgo
Searching for: https://www.youtube.com

C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism\methodOverload>java search.java
Enter search query: https://www.youtube.com
would you like to use your desired search engine?(y/n) n
Search Engine: google
Searching for: https://www.youtube.com

C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism\methodOverload>
```

10 b)Calculating Simple Interest

Code:

```
class simpleInterestCalc{
	public static void main(String[] args){
	Calc obj=new Calc();
	System.out.println("When value for 'period' is not passed");
	obj.s_interest(1000,5);
	System.out.println("\nWhen value for 'period' is passed as 2");
	obj.s_interest(1000,5,2);

	}
}

class Calc{
	public static void s_interest(double principle, double rate, double
period){
		double interest=principle*(rate/100.0)*period;
		System.out.println("Principle amount: " + principle +",Rate: "+ rate
+", Period(In yrs): " + period );
		System.out.println("Interest is " + interest);
	}
	public static void s_interest(double principle, double rate){
		double interest=principle*(rate/100.00)*1.00;
		System.out.println("Principle amount: "+principle+",Rate: "+rate+",
Period(In yrs): "+ 1 );
		System.out.println("Interest rate is "+interest);
	}

}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism\methodOverload>java simpleInterestCalc.java
When value for 'period' is not passed
Principle amount: 1000.0,Rate: 5.0, Period(In yrs): 1
Interest rate is 50.0

When value for 'period' is passed as 2
Principle amount: 1000.0,Rate: 5.0, Period(In yrs): 2.0
Interest is 100.0

C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism\methodOverload>
```

11.Method Overriding

11 a)Implementing Discount System

Code:

```
class DiscountOverride {
    public static void main(String[] args) {
        Discount normal = new Discount();
        Discount special = new SpecialDiscount();

        normal.applyDiscount(1000, 10);
        special.applyDiscount(1000, 10);
    }
}

class Discount {
    void applyDiscount(double price, double discountPercent) {
        double finalPrice = price - (price * discountPercent / 100);
        System.out.println("Final Price after normal discount: Rs" + finalPrice);
    }
}

class SpecialDiscount extends Discount {
    void applyDiscount(double price, double discountPercent) {
        double specialAmount = 5;
        double finalPrice = price - (price * (discountPercent + specialAmount) /
100);
        System.out.println("Final Price with special discount: Rs" + finalPrice);
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism\MethodOverriding>java DiscountOverride.java
Final Price after normal discount: Rs900.0
Final Price with special discount: Rs850.0

C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism\MethodOverriding>
```

11 b)Implement Laptops

Code:

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

```
        Laptop normal = new Laptop();
        Laptop better = new GamingLaptop();

        normal.features();
        better.features();
    }
}

class Laptop {
    void features() {
        System.out.println("Laptop features: Browsing, Docs, Videos");
    }
}

class GamingLaptop extends Laptop {
    void features() {
        System.out.println("Gaming Laptop features: High-performance GPU,
Cooling, RGB");
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism\MethodOverriding>java laptopOverride.java
Laptop features: Browsing, Docs, Videos
Gaming Laptop features: High-performance GPU, Cooling, RGB

C:\amrita\OOP\Exercise4_JavaConcepts\polymorphism\MethodOverriding>
```

Abstraction
12. Interface Programs

12 a)Implementing Amphibian features

Code:

```
public class amphibians {
    public static void main(String[] args) {
        Amphibian frog = new Amphibian();
        frog.walkOnLand();
        frog.swim();
    }
}

interface Terestrial {
    void walkOnLand();
}

interface Aquatic {
    void swim();
}

class Amphibian implements Terestrial, Aquatic {
    public void walkOnLand() {
        System.out.println("Walking on land");
    }

    public void swim() {
        System.out.println("Swimming in water");
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Abstraction\interface>java amphibians.java
Walking on land
Swimming in water
```

12 b) implementing birds

Code:

```
public class birds {
    public static void main(String[] args) {
        Bird bird = new Bird();
```

```
        bird.fly();
        bird.sing();
    }
}
interface Aerial{
    void fly();
}

interface Singer {
    void sing();
}

class Bird implements Aerial, Singer {
    public void fly() {
        System.out.println("The bird is flying.");
    }

    public void sing() {
        System.out.println("The bird is singing.");
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Abstraction\interface>java birds.java
The bird is flying.
The bird is singing.
```

12 c)Implementing Phones

Code:

```
public class phone_interface {
    public static void main(String[] args) {
        Phone phone = new Phone();
        phone.makeCall();
```

```
        phone.sendMessage();
    }
}
interface Caller {
    void makeCall();
}

interface Messenger {
    void sendMessage();
}

class Phone implements Caller, Messenger {
    public void makeCall() {
        System.out.println("Making a phone call...");
    }

    public void sendMessage() {
        System.out.println("Sending a text message...");
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Abstraction\interface>java phone_interface.java
Making a phone call...
Sending a text message...
```

12 d) Implementing a recording and playing device

Code:

```
public class recorder_player {
    public static void main(String[] args) {
        MediaDevice device = new MediaDevice();
        device.play();
        device.record();
    }
```

```
}

interface Player {
    void play();
}

interface Recorder {
    void record();
}

class MediaDevice implements Player, Recorder {
    public void play() {
        System.out.println("Playing audio...");
    }

    public void record() {
        System.out.println("Recording audio...");
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Abstraction\interface>java recorder_player.java
Playing audio...
Recording audio...
```

13. Abstract Class Programs

13 a)Implementing Employees

Code:

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

```
        Employee emp1 = new FullTimeEmployee("Vaish", 4000);
        Employee emp2 = new PartTimeEmployee("Peter Parker", 20, 80);

        emp1.displayInfo();
        System.out.println("Monthly Salary: $" + emp1.calculateSalary());

        emp2.displayInfo();
        System.out.println("Monthly Salary: $" + emp2.calculateSalary());
    }
}

abstract class Employee {
    String name;
    Employee(String name) {
        this.name = name;
    }

    abstract double calculateSalary();

    void displayInfo() {
        System.out.println("Employee Name: " + name);
    }
}

class FullTimeEmployee extends Employee {
    double salary;

    FullTimeEmployee(String name, double salary) {
        super(name);
        this.salary = salary;
    }

    double calculateSalary() {
        return salary;
    }
}

class PartTimeEmployee extends Employee {
    double hourlyRate;
    int hoursWorked;

    PartTimeEmployee(String name, double hourlyRate, int hoursWorked) {
        super(name);
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }

    double calculateSalary() {
        return hourlyRate * hoursWorked;
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Abstraction\Abstract>java employees_abstract.java
Employee Name: Vaish
Monthly Salary: $4000.0
Employee Name: Peter Parker
Monthly Salary: $1600.0
```

13 b)Implementing Payment System

Code:

```
public class payment_abstract {
    public static void main(String[] args) {
        Payment credit = new CreditCardPayment(120.50, "1234567812345678");
        Payment UPI = new UPIPayment(89.99, "1234567890@bankname");

        credit.paymentInfo();
        credit.makePayment();

        System.out.println();
```

```
        UPI.paymentInfo();
        UPI.makePayment();
    }
}

abstract class Payment {
    double amount;

    Payment(double amount) {
        this.amount = amount;
    }

    abstract void makePayment();

    void paymentInfo() {
        System.out.println("Processing payment of Rs" + amount);
    }
}

class CreditCardPayment extends Payment {
    String cardNumber;

    CreditCardPayment(double amount, String cardNumber) {
        super(amount);
        this.cardNumber = cardNumber;
    }

    void makePayment() {
        System.out.println("Charged Rs" + amount + " to credit card ending in " +
cardNumber.substring(cardNumber.length() - 4));
    }
}

class UPIPayment extends Payment {
    String upiID;

    UPIPayment(double amount, String upiID) {
        super(amount);
        this.upiID = upiID;
    }

    void makePayment() {
        System.out.println("Paid Rs" + amount + " via UPI account: " + upiID);
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Abstraction\Abstract>java payment_abstract.java
Processing payment of Rs120.5
Charged Rs120.5 to credit card ending in 5678

Processing payment of Rs89.99
Paid Rs89.99 via UPI account: 1234567890@bankname
```

13 c)Implementing Phones

Code:

```
class phone_abstract{
	public static void main(String[] args){
		SmartPhone obj=new SmartPhone();
		obj.turnOn();
		obj.setNum("1234567890");
		obj.call("1234567891");
		obj.download("YouTube");
		obj.turnOff();
	}
}
```

```
abstract class Phone{
	String ph_no;
	public void setNum(String num){
		if(num.length() ==10){
		ph_no=num;}
		else{
		System.out.println("NUMBER MUST ONLY BE 10 DIGITS");
		}
	}

	public void call(String num){
		if(num.length()== 10 && num!=ph_no){
		System.out.println("calling "+ num);}
		else if(num==ph_no){
		System.out.println("YOU CANNOT DIAL YOURSELF");
		}
		else{
		System.out.println("NUMBER MUST ONLY BE 10 DIGITS");
		}
	}
	public void turnOn(){
		System.out.println("Turning on...");
	}
	public void turnOff(){
		System.out.println("Turning off...");
	}
}

class SmartPhone extends Phone{
	public  void download(String app){
		System.out.println("Downloading "+app+"...");
	}
	public void playMusic(){
		System.out.println("Playing Music!");
	}
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Abstraction\Abstract>java phone_abstract.java
Turning on...
calling 1234567891
Downloading YouTube...
Turning off...
```

13 d)Implementing Vehicle

Code:

```
class transportaion_abstract{
	public static void main(String[] args){
		Car carOBJ=new Car();
		carOBJ.set("electric","tesla",4,4);
		carOBJ.recharge();
		carOBJ.start();
		carOBJ.stop();
	}
}

abstract class Vehicle{
	String type;
	String manufacturer;
```

```
	int no_of_wheels;
	int max_no_of_passengers;
	public void set(String t, String man, int now, int nop){
		type=t;
		manufacturer=man;
		no_of_wheels=now;
		max_no_of_passengers=nop;
	}
	abstract public void start();
	abstract public  void stop();
}

class Car extends Vehicle{
	public void start(){
		System.out.println("STARTING THE CAR...");
	}
	public void stop(){
		System.out.println("STOPPING THE CAR...");
	}
	public void recharge(){
		System.out.println("REFUELLING THE CAR.....");
	}
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Abstraction\Abstract>java transportation_abstract.java
REFUELLING THE CAR.....
STARTING THE CAR...
STOPPING THE CAR...
```

ENCAPSULATION

14. ENCAPSULATION PROGRAMS

14 a)Implementing Banking Account System

Code:

```
public class Account {
    public static void main(String[] args) {
        BankAccount acc = new BankAccount();
        acc.setAccountHolder("Vaish");
        acc.deposit(5000);
        acc.withdraw(1200);

        System.out.println("Account Holder: " + acc.getAccountHolder());
```

```
        System.out.println("Remaining Balance: Rs" + acc.getBalance());
    }
}

class BankAccount {
    private String accountHolder;
    private double balance;

    public void setAccountHolder(String name) {
        accountHolder = name;
    }

    public String getAccountHolder() {
        return accountHolder;
    }

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

    public double getBalance() {
        return balance;
    }

    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        } else {
            System.out.println("Insufficient funds.");
        }
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Encapsulation>java Account.java
Account Holder: Vaish
Remaining Balance: Rs3800.0
```

14 b)Implementing Login System

Code:

```
public class Login {
    public static void main(String[] args) {
        LoginSystem login = new LoginSystem();
        login.setCredentials("admin", "1234");

        if (login.validate("admin", "1234")) {
            System.out.println("Login successful!");
        } else {
            System.out.println("Invalid credentials.");
```

```
        }
    }
}

class LoginSystem {
    private String username;
    private String password;

    public void setCredentials(String u, String p) {
        username = u;
        password = p;
    }

    public boolean validate(String inputUser, String inputPass) {
        return inputUser.equals(username) && inputPass.equals(password);
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Encapsulation>java Login.java
Login successful!
```

14 c)Implementing Product Tracking System

Code:

```
public class Products {
    public static void main(String[] args) {
        Product item = new Product();
        item.setName("Laptop");
        item.setPrice(55000);
        item.setQuantity(2);
        System.out.println("Product: " + item.getName());
        System.out.println("Total Value: ₹" + item.getTotalValue());
    }
}

class Product {
    private String name;
    private double price;
    private int quantity;
    public void setName(String n) {
        name = n;
    }
```

```
    public String getName() {
        return name;
    }

    public void setPrice(double p) {
            price = p;
    }

    public double getPrice() {
        return price;
    }

    public void setQuantity(int q) {
            quantity = q;
    }

    public int getQuantity() {
        return quantity;
    }

    public double getTotalValue() {
        return price * quantity;
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Encapsulation>java Products.java
Product: Laptop
Total Value: Rs110000.0
```

14 d)Implementing Calculator For Rectangle

Code:

```
public class RectangleCalc{
    public static void main(String[] args) {
        Rectangle r = new Rectangle();
        r.setDimensions(5.5, 3.0);

        System.out.println("Area: " + r.getArea());
        System.out.println("Perimeter: " + r.getPerimeter());
    }
}

class Rectangle {
    private double length;
    private double width;

    public void setDimensions(double l, double w){
            length=l;
            width=w;
    }
```

```
    public double getArea() {
        return length*width;
    }

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

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Encapsulation>java RectangleCalc.java
Area: 16.5
Perimeter: 17.0
```

15. PACKAGES PROGRAMS

15 a)User Defined Packages (Implementing Basic Calculator)

Code:

User Defined Package:

```
import  userPacks.Calc;

class simpleCalc{
    public static void main(String[] args) {
        Calc calcOBJ =new Calc();
        int sum=calcOBJ.add(5,10);
        int dif=calcOBJ.subtract(5, 10);
        int prod=calcOBJ.multiply(5, 10);
        double quo=calcOBJ.divide(5, 10);
        System.out.println("SUM: "+sum);
        System.out.println("DIFFERENCE: "+dif);
        System.out.println("PRODUCT: "+prod);
        System.out.println("QUOTIENT: "+quo);

    }
}
```

Main Program:

```
import  userPacks.Calc;

class simpleCalc{
    public static void main(String[] args) {
        Calc calcOBJ =new Calc();
        int sum=calcOBJ.add(5,10);
        int dif=calcOBJ.subtract(5, 10);
        int prod=calcOBJ.multiply(5, 10);
        double quo=calcOBJ.divide(5, 10);
        System.out.println("SUM: "+sum);
        System.out.println("DIFFERENCE: "+dif);
        System.out.println("PRODUCT: "+prod);
        System.out.println("QUOTIENT: "+quo);

    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Packages\UserDefined>java simpleCalc.java
SUM: 15
DIFFERENCE: -5
PRODUCT: 50
QUOTIENT: 0.5
```

15 b)User Defined Packages (Implementing a Square and Cube Calculator)

Code:

User Defined Package:

```
package userPacks;

public class MoreMath{
    public double square(double num){
        return num*num;
    }
    public double cube(double num){
        return num*num*num;
    }
}
```

Main Program:

```
import userPacks.MoreMath;

class squareCubes{
    public static void main(String[] args) {
        MoreMath obj=new MoreMath();
        System.out.println("Square of 2 is "+obj.square(2));
        System.out.println("Cube of 3 is "+obj.cube(3));
    }
```

```
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Packages\UserDefined>java SquareCubes.java
Square of 2 is 4.0
Cube of 3 is 27.0
```

15 c)Built-In Packages (Using util and lang)

Code:

```
import java.util.Scanner;
import java.lang.Math;
class findSquare{
    public static void main(String[] args){
        Scanner scan= new Scanner(System.in);
        System.out.print("Enter number: ");
        double num=scan.nextDouble();
        System.out.print("Enter power");
        double power=scan.nextDouble();
        double answer= Math.pow(num, power);
        System.out.println("ANSWER: "+answer);
    }
}
```

Output:

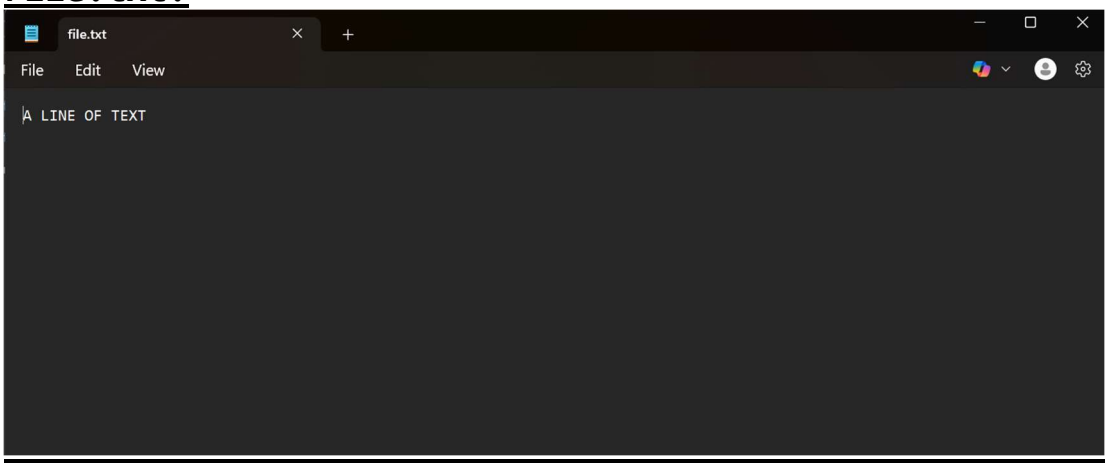
```
C:\amrita\OOP\Exercise4_JavaConcepts\Packages\BuiltIn>java findSquare.java
Enter number: 5
Enter power2
ANSWER: 25.0
```

15 d)Built-In Packages(Using util and io)

Code:

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

public class noOfLetters {
    public static void main(String[] args) {
        int length=0;
        try {
            File file = new File("file.txt");
            Scanner reader = new Scanner(file);
            System.out.println("Reading file content:\n");
            while (reader.hasNextLine()) {
                String line = reader.nextLine().replace(" ","");
                length=length+line.length();
            }
            reader.close();
        }
        catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
            e.printStackTrace();
        }
        System.out.println("There are "+length+" letters");
    }
}
```

File.txt:



Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\Packages\BuiltIn>java noOfLetters.java
Reading file content:

There are 11 letters
```

16. EXCEPTION HANDLING PROGRAMS

16 a)Finding square root of non-negative numbers

Code:

```
import java.util.*;
public class SquareRoot{
    public static void main(String[] args) {
        Scanner scan=new Scanner(System.in);
        System.out.println("Enter a number to find square root");
        double num=scan.nextDouble();
            try {
                if (num < 0) {
                    throw new IllegalArgumentException("Cannot take square root
of negative number");
                }
                double root = Math.sqrt(num);
                System.out.println("Square root of" + num + " = " + root);
            } catch (IllegalArgumentException e) {
                System.out.println("Imaginary Numbers Not Allowed");
            }
        }
    }
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\ExceptionHandling>java SquareRoot.java
Enter a number to find square root
4
Square root of4.0 = 2.0

C:\amrita\OOP\Exercise4_JavaConcepts\ExceptionHandling>java SquareRoot.java
Enter a number to find square root
-3
Imaginary Numbers Not Allowed
```

16 b)Detecting index out of range

Code:

```
public class indexOut {
    public static void main(String[] args) {
        int[] arr = {10, 20, 30};

        try {
            System.out.println(arr[5]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Array index out of bounds.");
        }
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\ExceptionHandling>java indexOut.java
Error: Array index out of bounds.
```

16 c)Detecting invalid input

Code:

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

```
        Scanner scan=new Scanner(System.in);
        System.out.println("Enter a number");
      try {
        int num=scan.nextInt();
        System.out.println("Number: " + num);
      } catch (InputMismatchException e) {
        System.out.println("Error: Invalid number.");
      }
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\ExceptionHandling>java invalidNum.java
Enter a number
55
Number: 55

C:\amrita\OOP\Exercise4_JavaConcepts\ExceptionHandling>java invalidNum.java
Enter a number
abcd
Error: Invalid number.
```

16 d)Catching Zero Division

Code:

```
import java.util.Scanner;

public class zeroDiv {
    public static void main(String[] args) {
      Scanner scan=new Scanner(System.in);
        try {
          System.out.println("Enter Numerator");
            int a = scan.nextInt();
          System.out.println("Enter Denominator");
          int b = scan.nextInt();
            int result = a / b;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed.");
        }
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\ExceptionHandling>java zeroDiv.java
Enter Numerator
10
Enter Denominator
2
Result: 5

C:\amrita\OOP\Exercise4_JavaConcepts\ExceptionHandling>java zeroDiv.java
Enter Numerator
10
Enter Denominator
0
Error: Division by zero is not allowed.
```

17. FILE HANDLING PROGRAMS

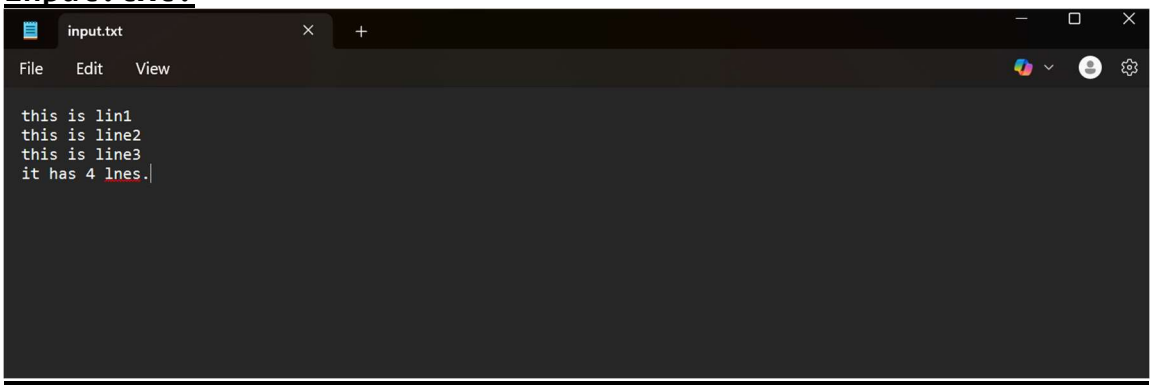
17 a)Copying a file into another file

Code:

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

public class copyFile {
    public static void main(String[] args) {
        File inFile = new File("input.txt");
        File outFile = new File("output.txt");
```

```
        try {
            Scanner reader = new Scanner(inFile);
            FileWriter writer = new FileWriter(outFile);
            while (reader.hasNextLine()) {
                String line = reader.nextLine();
                writer.write(line + "\n");
            }
            reader.close();
            writer.close();
            System.out.println("File copied");

        } catch (FileNotFoundException e) {
            System.out.println("Error file not found.");
        } catch (IOException e) {
            System.out.println("Error writing to the target file.");
            e.printStackTrace();
        }
    }
}
```

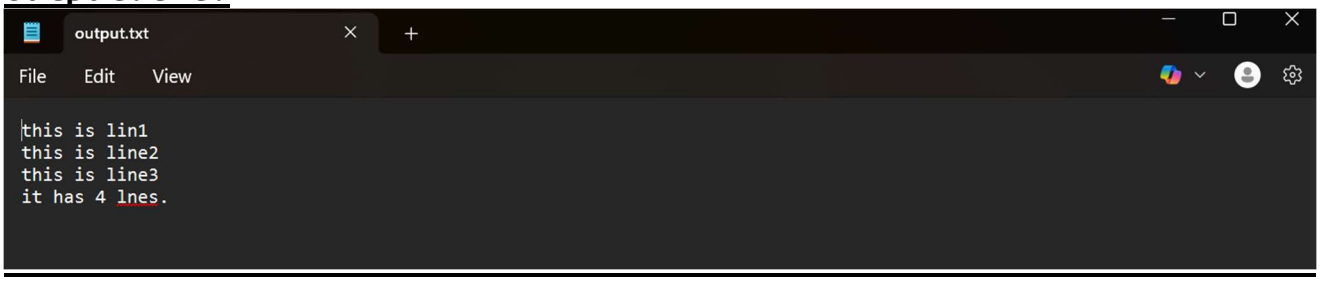
Input.txt:



Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\FileHandling>java copyFile.java
File copied
```

output.txt:



17 b)Creating a file and writing to it

Code:

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

public class createFile {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("output.txt");
            writer.write("THIS THE CONTENT THAT WILL BE WRITTEN IN THE FILE");
            writer.close();
            System.out.println("File created and written to");
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\FileHandling>java createFile.java
File created and written to
```

output.txt:

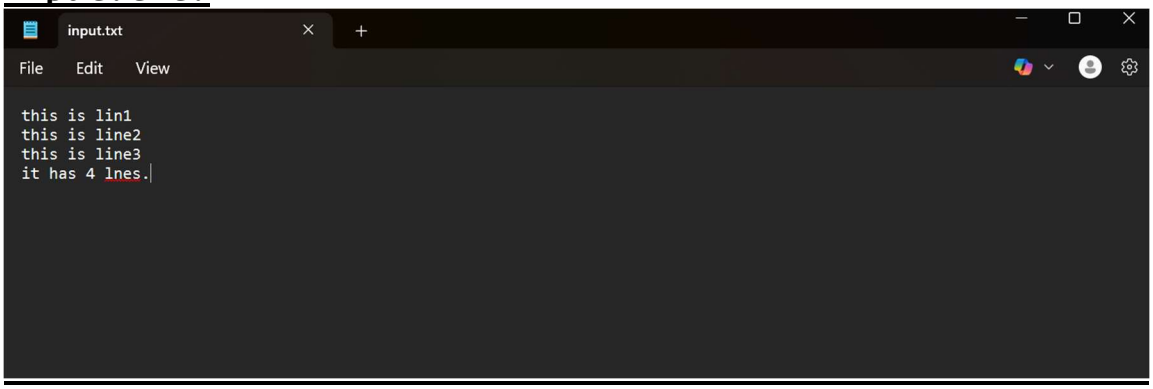
17 c)Finding number of lines in a file

Code:

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

public class noOfLines {
    public static void main(String[] args) {
        int c=0;
        try {
            File file = new File("input.txt");
            Scanner reader = new Scanner(file);
            System.out.println("Reading file content:\n");
            while (reader.hasNextLine()) {
                String line = reader.nextLine();
                c+=1;
            }
            reader.close();
        }
        catch (FileNotFoundException e) {
```

```
            System.out.println("Error: File not found.");
            e.printStackTrace();
        }
        System.out.println("There are "+c+" lines");
    }
}
```

input.txt:



Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\FileHandling>java noOfLines.java
Reading file content:

There are 4 lines
```

17 d)Reading the contents of a file

Code:

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

public class readFile {
    public static void main(String[] args) {
        try {
            File file = new File("input.txt");
            Scanner reader = new Scanner(file);
            System.out.println("Reading file content:\n");
            while (reader.hasNextLine()) {
                String line = reader.nextLine();
                System.out.println(line);
            }
            reader.close();
        }
        catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
            e.printStackTrace();
        }
```

```
    }
}
```

Output:

```
C:\amrita\OOP\Exercise4_JavaConcepts\FileHandling>java readFile.java
Reading file content:

this is lin1
this is line2
this is line3
it has 4 lnes.
```