LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24140 – Rohith S

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING

CHENNAI

March - 2025

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.U4CSE24140 – ROHITH S*** 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 / /2025

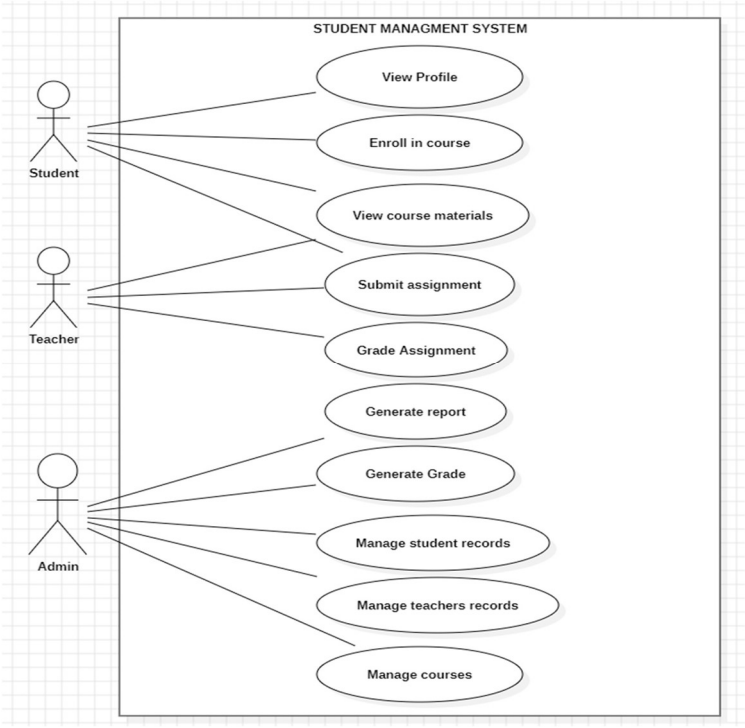
Internal Examiner 1 Internal Examiner 2

INDEX

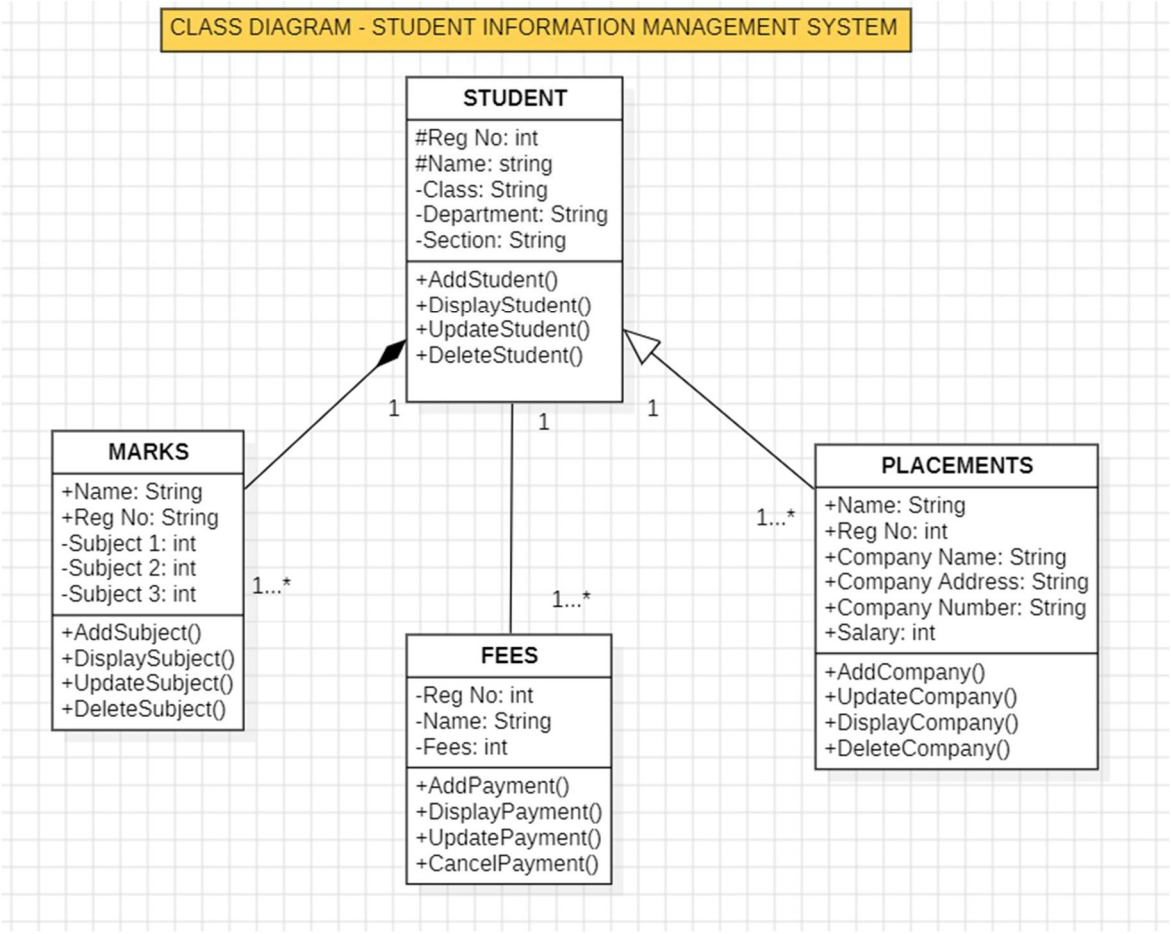
UML DIAGRAMS

1. STUDENT MANAGEMENT

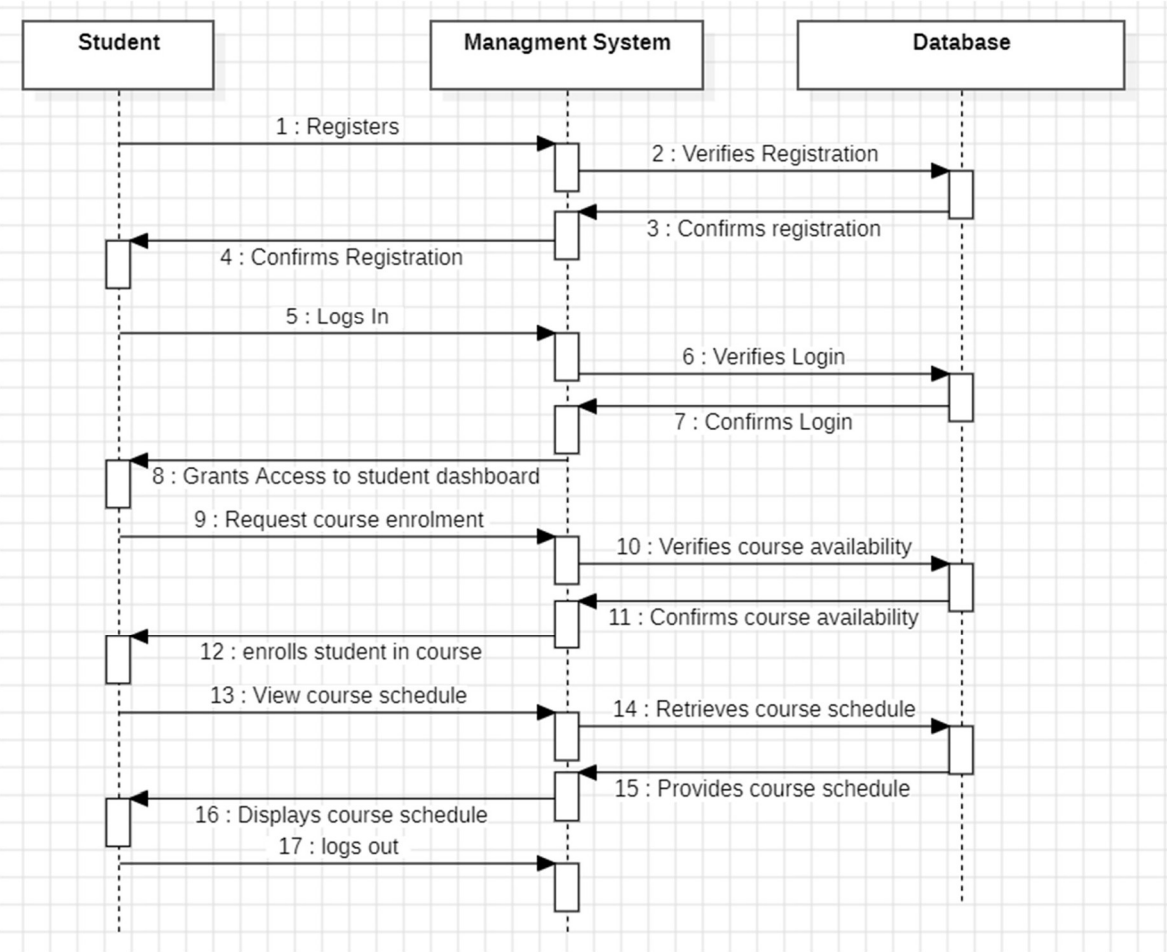
1.a) Use Case Diagram:



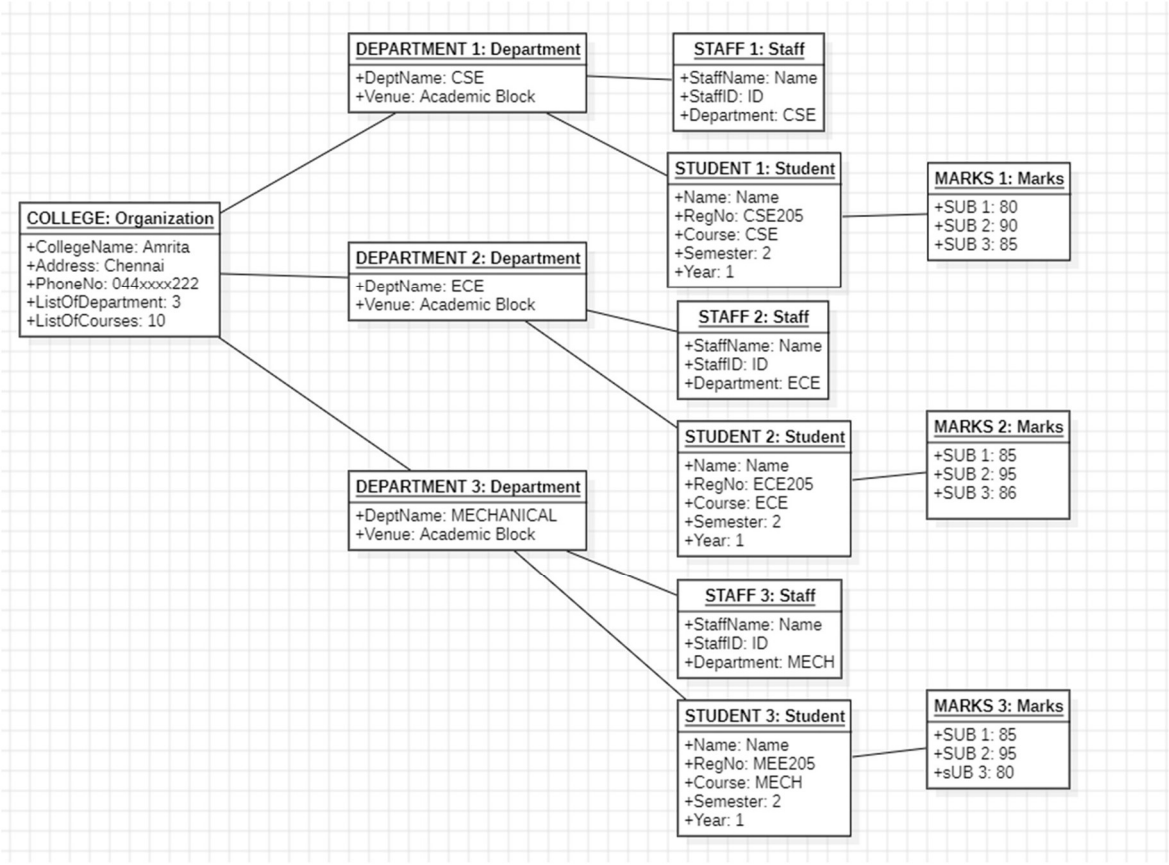
1.b) Class Diagram:



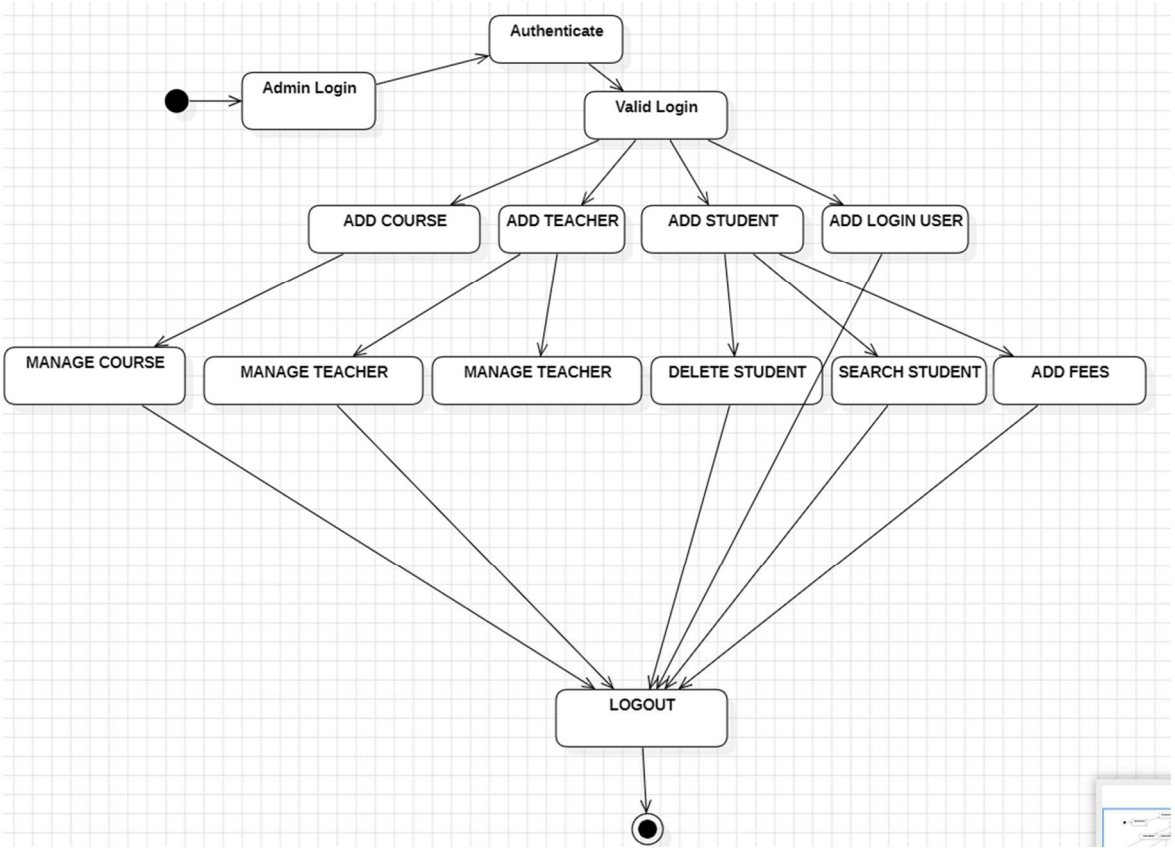
1.c) Sequence Diagram:



1.d) Object Diagram:

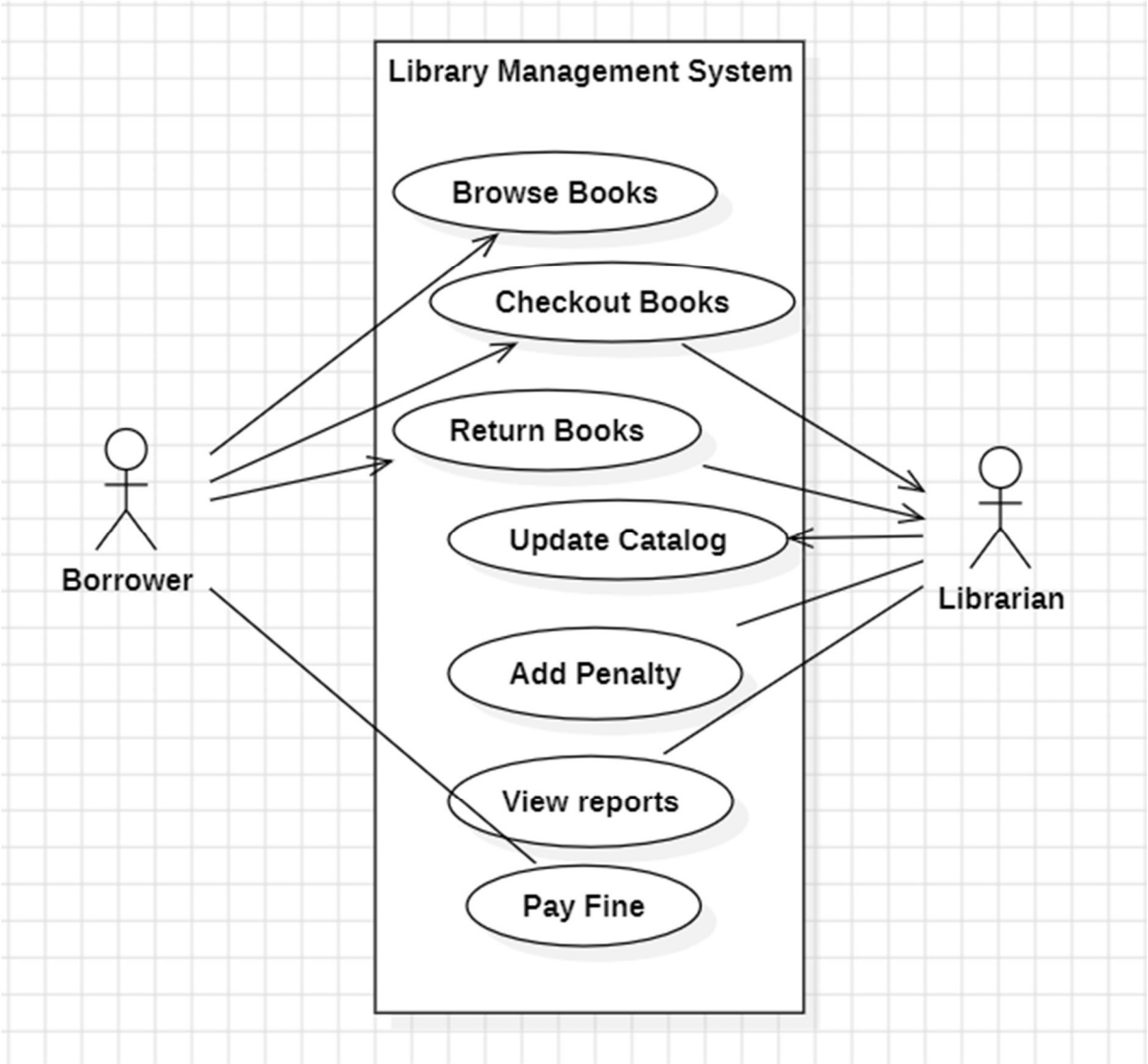


1.e) State-Activity Diagram:

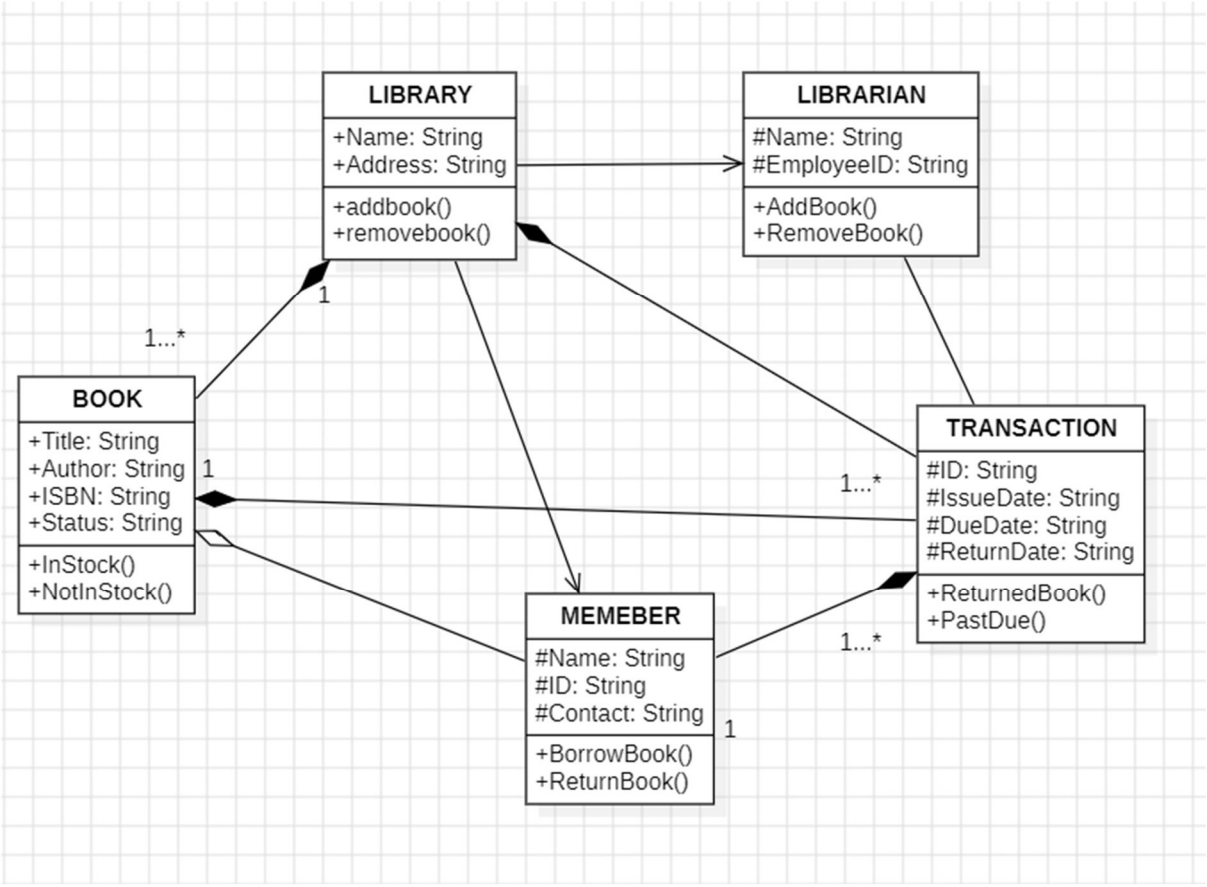


2. LIBRARY MANAGEMENT

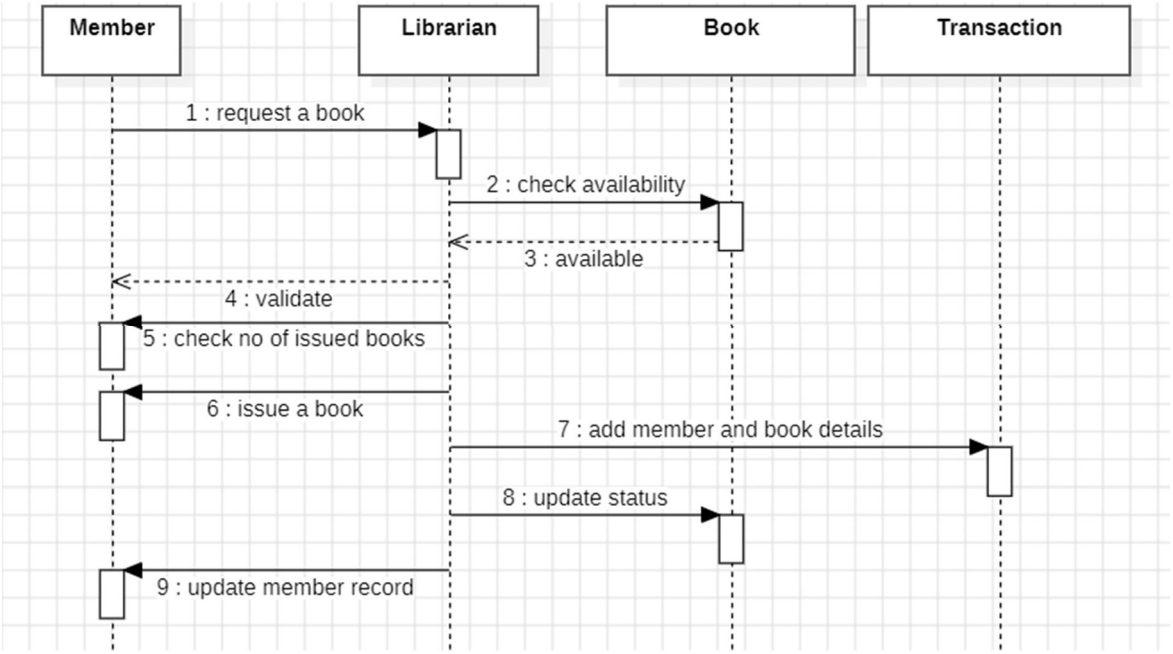
2.a) Use Case Diagram:



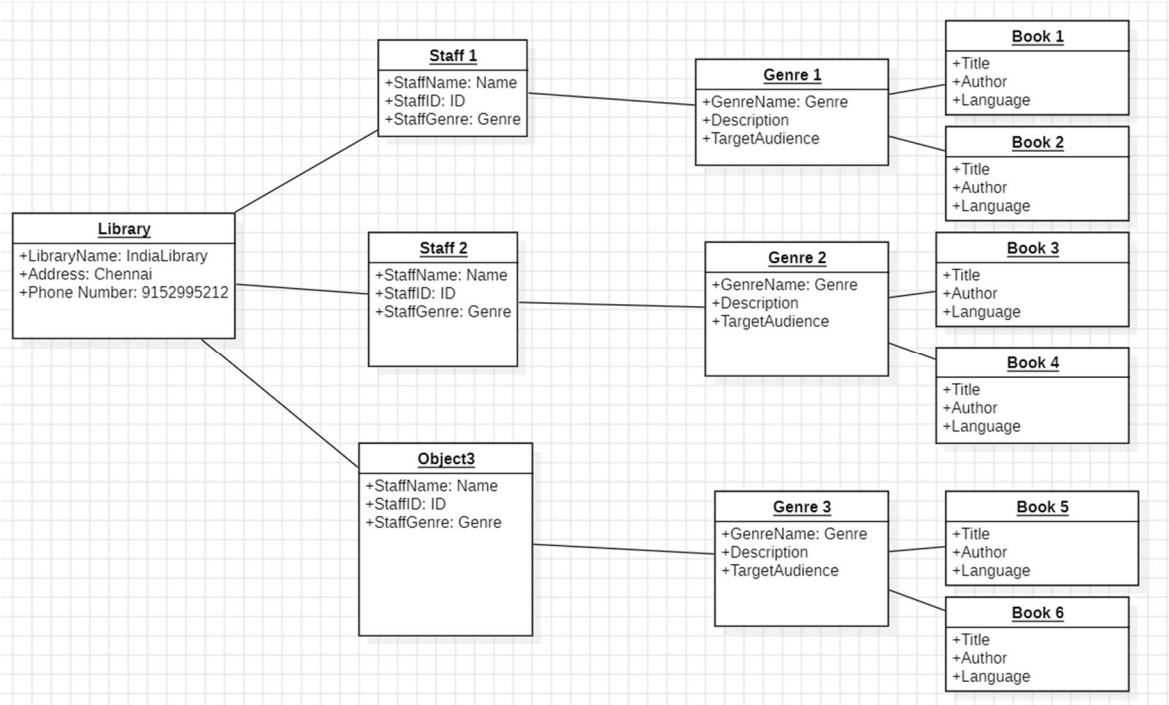
2.b) Class Diagram:



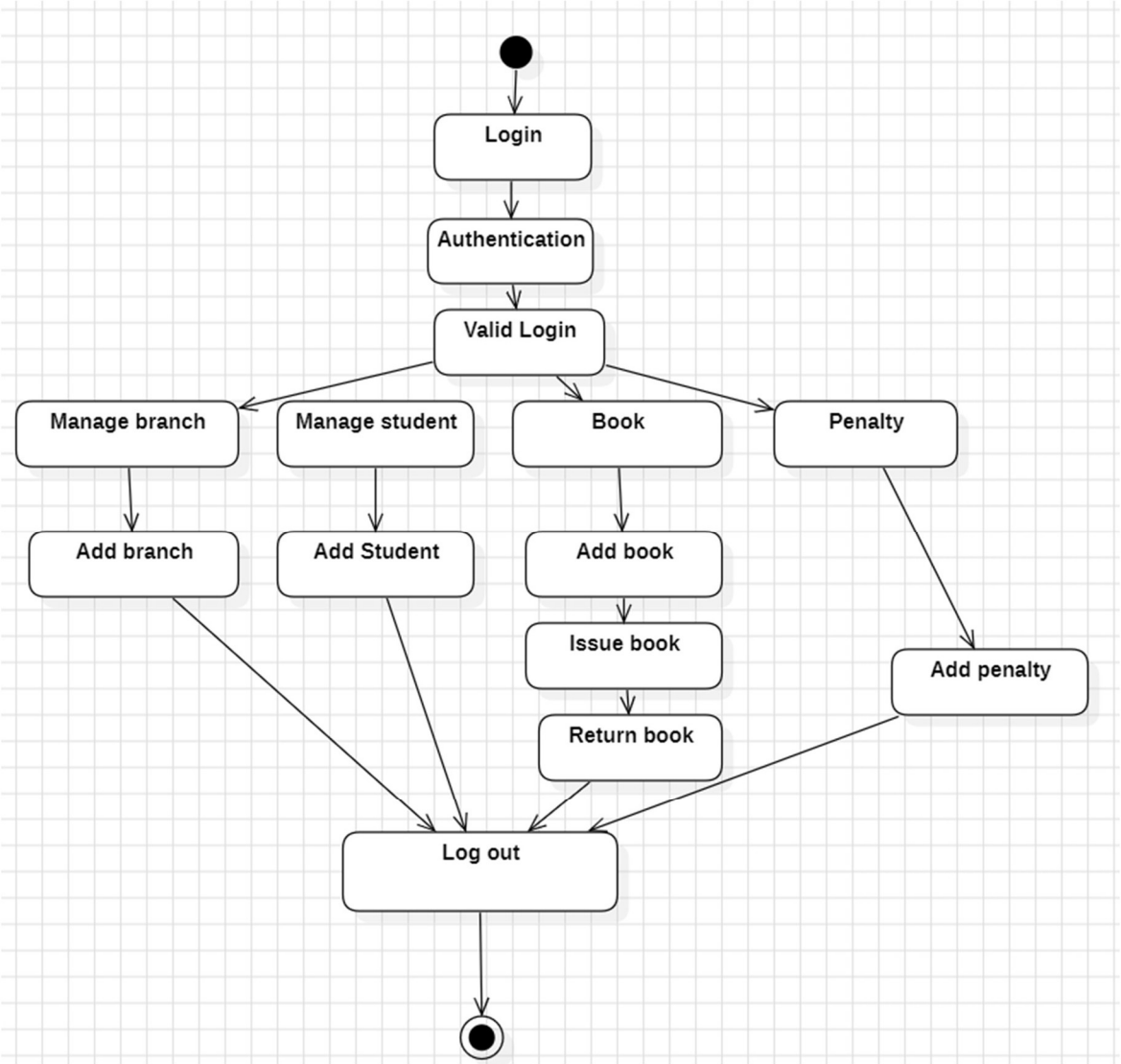
2.c) Sequence Diagram:



2.d) Object Diagram:



2.e) State-Activity Diagram:



3. Basic Java Programs

3.a) Simple Calculator:

Code:

```
import java.util.*;
public class calc {
    int add(int num1, int num2) {
        return num1 + num2;
    }

    int sub(int num1, int num2) {
        return num1 - num2;
    }

    int multiply(int num1, int num2) {
        return num1 * num2;
    }

    double divide(int num1, int num2) {
        if (num2 == 0) {
            System.out.println("Error! Can't divide by zero!");
            return 0;
        }
        return (double) num1 / num2;
    }

    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        calc obj = new calc();

        System.out.print("Enter your first number: ");
        int num1 = sc.nextInt();

        System.out.print("Enter your second number: ");
        int num2 = sc.nextInt();

        sc.nextLine();

        System.out.print("Choose your operator (+ - * /): ");
        String oper = sc.nextLine();

        if (oper.equals("+")) {
            System.out.println("Addition: " + obj.add(num1, num2));
        } else if (oper.equals("-")) {
            System.out.println("Subtraction: " + obj.sub(num1, num2));
        } else if (oper.equals("*")) {
```

```
            System.out.println("Multiplication: " + obj.multiply(num1,
num2));
        } else if (oper.equals("/")) {
            System.out.println("Division: " + obj.divide(num1, num2));
        } else {
            System.out.println("Invalid operator!");
        }
    }
}
```

Output:

```
Enter your first number: 200
Enter your second number: 3
Choose your operator (+ - * /): *
Multiplication: 600
PS C:\Users\123sr\OneDrive\Desktop\Basic java programs>
```

3.b) Even or Odd:

Code:

```
import java.util.*;
public class evenorodd{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a number: ");
        int a = sc.nextInt();
        if(a%2==0){
            System.out.println("It is a even number");
        }
        else{
            System.out.println("It is a odd number");
        }
    }
}
```

Output:

```
Enter a number:
25253
It is a odd number
PS C:\Users\123sr\OneDrive\Desktop\Basic java programs>
```

3.c) Factorial:

Code:

```
import java.util.*;
public class factorial{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your number: ");
        int a = sc.nextInt();
        sc.close();
        int fact=1;
        if(a>0){
            for(int i =1;i<=a;i++){
            fact = fact*i;
            }
            System.out.println("The factorial is "+fact);
        }
        else if(a==0){
            System.out.println("The factorial is 1");
        }
        else{
            System.out.println("Invalid input");
        }

        }
    }
```

Output:

```
Enter your number: 6
The factorial is 720
PS C:\Users\123sr\OneDrive\Desktop\Basic java programs>
```

3.d) Fibonacci Series:

Code:

```
import java.util.Scanner;
public class fibonacci {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n = sc.nextInt(), a = 0, b = 1;
        System.out.print("Fibonacci Series For first "+n+" Numbers " + a +
" " + b);
        for (int i = 2; i < n; i++) {
            int next = a + b;
            System.out.print(" " + next);
            a = b; b = next;
        }
        sc.close();
    }
}
```

Output;

```
Enter n: 5
Fibonacci Series For first 5 Numbers 0 1 1 2 3
```

3.e) Largest of three numbers :

Code:

```
import java.util.*;
public class largest3numbers{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your first number: ");
        int a = sc.nextInt();
        System.out.println("Enter your second number: ");
        int b = sc.nextInt();
        System.out.println("Enter your third number: ");
        int c = sc.nextInt();
        if(a>b && a>c){
            System.out.println(a+" is the largest number ");
        }
        else if(b>a && b>c){
            System.out.println(b+" is the largest number ");

        }
        else if(c>a && c>b){
         System.out.println(a+" is the largest number ");
        }

    }
}
```

Output:

```
Enter your first number:
4
Enter your second number:
2
Enter your third number:
5
5 is the largest number
PS C:\Users\123sr\OneDrive\Desktop\Basic java programs>
```

3.f) Number of Digits:

Code:

```
import java.util.*;
public class numdigits{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your number: ");
        int number =sc.nextInt();
String numberString = String.valueOf(number);
int numberOfDigits = numberString.length();
System.out.println("Number of digits is: "+numberOfDigits);
    }
}
```

Output:

```
Enter your number: 95295125
Number of digits is: 8
PS C:\Users\123sr\OneDrive\Desktop\Basic java programs>
```

3.g) Palindrome Check:

Code:

```
import java.util.*;
public class palindrome{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter your number: ");
        int num = sc.nextInt();
        sc.close();
        int rev = 0;
        int r =0;
        int num2=num;
        while(num>0){
            r = num % 10;
            rev = rev*10+r;
            num=num/10;
        }
        if(rev==num2){
            System.out.println("It is a palindrome! ");
```

```
        }
        else{
            System.out.println("It is not a palindrome! ");
        }
    }
}
```

Output:

```
Enter your number: 212
It is a palindrome!
PS C:\Users\123sr\OneDrive\Desktop\Basic java programs>
```

3.h) Prime Checker:

Code:

```
import java.util.*;
public class primenumber{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        sc.close();
        boolean isPrime = num>1;
        for (int i=2;i*i<=num;i++){

        if (num%i==0){
            isPrime=false;
            break;
        }
    }
        if(isPrime){
            System.out.println("Prime");
        }
        else{
            System.out.println("Not prime");
        }
}
}
```

Output:

```
Enter a number: 429
Not prime
PS C:\Users\123sr\OneDrive\Desktop\Basic java programs>
```

3.i) Reverse Number:

Code:

```
import java.util.*;

public class numreverse {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = sc.nextInt();
        int r = 0;
        int rev = 0;
        while(num > 0) {
            r = num % 10;
            rev = rev * 10 + r;
            num = num / 10;
        }
        System.out.println("Reversed number is: "+rev);
    }
}
```

Output:

```
Enter a number: 42141
Reversed number is: 14124
PS C:\Users\123sr\OneDrive\Desktop\Basic java programs>
```

3.j) String Reversal:

Code:

```
import java.util.*;
public class reversestr{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your string");
        String str =sc.nextLine();
        String rstr = " ";
        char ch;
        for(int i = 0;i<str.length();i++){
            ch=str.charAt(i);
            rstr=ch+rstr;

        }
        System.out.println(rstr);
    }
}
```

Output:

```
Enter your string
hello
olleh
PS C:\Users\123sr\OneDrive\Desktop\Basic java programs>
```

INHERITANCE PROGRAMS

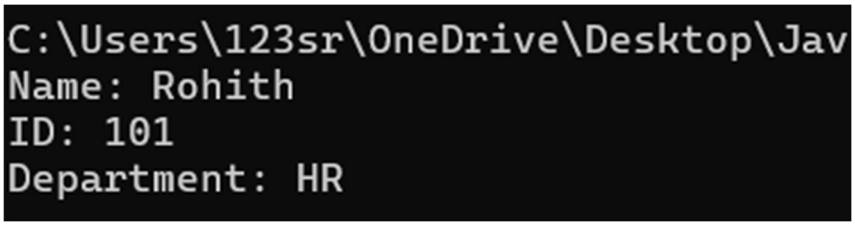
SINGLE INHERITANCE PROGRAMS

4.a)Employee System

CODE:

```
class Employee {
    String name = "Rohith";
    int id = 101;

    void showDetails() {
        System.out.println("Name: " + name);
        System.out.println("ID: " + id);
    }
}

class Manager extends Employee {
    void department() {
        System.out.println("Department: HR");
    }
}

public class SingleInheritance1 {
    public static void main(String[] args) {
        Manager m = new Manager();
        m.showDetails();
        m.department();
    }
}
```

OUTPUT:

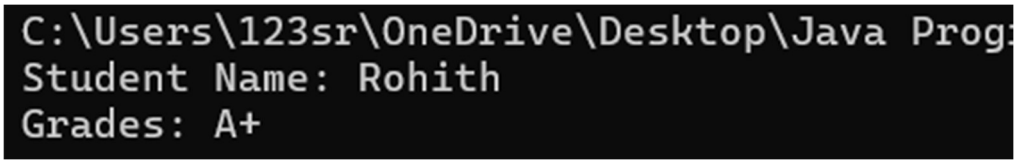


4.b)Grading System

CODE:

```
class Student {
    String name = "Rohith";

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

class Grades extends Student {
    void showGrades() {
        System.out.println("Grades: A+");
    }
}

public class SingleInheritance2 {
    public static void main(String[] args) {
        Grades g = new Grades();
        g.displayName();
        g.showGrades();
    }
}
```

OUTPUT:



MULTILEVEL INHERITANCE PROGRAMS

5.a)University System

CODE:

```
class University {
    void uniName() {
        System.out.println("Amrita Vishwa Vidyapeetham");
    }
}

class Department extends University {
    void deptName() {
        System.out.println("Department: CSE");
    }
}

class Student extends Department {
    void studentName() {
        System.out.println("Student: Rohith");
    }
}

public class MultilevelInheritance1 {
    public static void main(String[] args) {
        Student s = new Student();
        s.uniName();
        s.deptName();
        s.studentName();
    }
}
```

OUTPUT:

```
Amrita Vishwa Vidyapeetham
Department: CSE
Student: Rohith
PS C:\Users\123sr>
```

5.b)Employee System

CODE:

```
class Person {
    public Person() {
        System.out.println("Class Person");
    }
    public void showDetails() {
        System.out.println("Name: Rohith S");
    }
}

class Employee extends Person {
    public Employee() {
        System.out.println("Class Employee");
    }
    public void showSalary() {
        System.out.println("Salary: $60,000");
    }
}

class Manager extends Employee {
    public Manager() {
        System.out.println("Class Manager");
    }
    public void showDepartment() {
        System.out.println("Department: Operations");
    }
}

public class MultilevelInheritance2 {
    public static void main(String[] args) {
        Manager m = new Manager();
        m.showDetails();
        m.showSalary();
        m.showDepartment();
    }
}
```

OUTPUT:

```
Class Person
Class Employee
Class Manager
Name: Rohith S
Salary: $60,000
Department: Operations
PS C:\Users\123sr>
```

HIERARCHICAL INHERITANCE PROGRAMS

7.a)Animals

CODE:

```
class Animal {
    public Animal() {
        System.out.println("Class Animal");
    }
    public void sound() {
        System.out.println("Animals make sounds");
    }
}

class Dog extends Animal {
    public Dog() {
        System.out.println("Class Dog");
    }
    public void bark() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {
    public Cat() {
        System.out.println("Class Cat");
    }
    public void meow() {
        System.out.println("Cat meows");
    }
}
public class Hierarchial1 {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.sound();
        d.bark();
        System.out.println();
        Cat c = new Cat();
        c.sound();
        c.meow();
    }
}
```

OUTPUT:

```
Class Animal
Class Dog
Animals make sounds
Dog barks

Class Animal
Class Cat
Animals make sounds
Cat meows
PS C:\Users\123sr>
```

7.b)Vehicle System

CODE:

```
class Vehicle {
    public Vehicle() {
        System.out.println("Class Vehicle");
    }
    public void start() {
        System.out.println("Vehicle is starting");
    }
}

class Car extends Vehicle {
    public Car() {
        System.out.println("Class Car");
    }
    public void drive() {
        System.out.println("Driving the car");
    }
}

class Bike extends Vehicle {
    public Bike() {
        System.out.println("Class Bike");
    }
    public void ride() {
        System.out.println("Riding the bike");
    }
}

public class Hierarchial2 {
    public static void main(String[] args) {
        Car car = new Car();
        car.start();
        car.drive();
        System.out.println();
        Bike bike = new Bike();
        bike.start();
        bike.ride();
    }
}
```

OUTPUT:

```
Class Vehicle
Class Car
Vehicle is starting
Driving the car

Class Vehicle
Class Bike
Vehicle is starting
Riding the bike
PS C:\Users\123sr>
```

HYBRID INHERITANCE PROGRAMS

6.a)Animals

CODE:

```
interface Walkable {
    void walk();
}

interface Swimmable {
    void swim();
}

class Animal {
    public Animal() {
        System.out.println("Animal created");
    }

    public void eat() {
        System.out.println("Animal is eating");
    }
}

class Duck extends Animal implements Walkable, Swimmable {
    public Duck() {
        System.out.println("Duck created");
    }

    public void walk() {
        System.out.println("Duck is walking");
    }

    public void swim() {
        System.out.println("Duck is swimming");
    }
}

public class HybridInheritance1 {
    public static void main(String[] args) {
        Duck d = new Duck();
        d.eat();
        d.walk();
        d.swim();
    }
}
```

OUTPUT:

```
Animal created
Duck created
Animal is eating
Duck is walking
Duck is swimming
PS C:\Users\123sr>
```

6.b) Vehicles

CODE:

```
interface Engine {
    void startEngine();
    void stopEngine();
}

interface EntertainmentSystem {
    void playMusic();
}

class Vehicle {
    String model;
    int year;

    Vehicle(String model, int year) {
        this.model = model;
        this.year = year;
    }

    void displayInfo() {
        System.out.println("Model: " + model + ", Year: " + year);
    }
}

class Car extends Vehicle implements Engine, EntertainmentSystem {
    Car(String model, int year) {
        super(model, year);
    }

    public void startEngine() {
        System.out.println(model + " car engine started");
    }

    public void stopEngine() {
        System.out.println(model + " car engine stopped");
    }

    public void playMusic() {
        System.out.println("Playing music in " + model);
    }

    void drive() {
        System.out.println(model + " is driving");
    }
}

class Bicycle extends Vehicle {
    Bicycle(String model, int year) {
        super(model, year);
    }

    void pedal() {
        System.out.println(model + " bicycle is pedaling");
    }
```

```
}

class ElectricBike extends Bicycle implements Engine {
    ElectricBike(String model, int year) {
        super(model, year);
    }

    public void startEngine() {
        System.out.println(model + " electric bike motor started");
    }

    public void stopEngine() {
        System.out.println(model + " electric bike motor stopped");
    }
}

public class HybridInheritance2 {
    public static void main(String[] args) {
        Car myCar = new Car("Toyota", 2022);
        myCar.displayInfo();
        myCar.startEngine();
        myCar.drive();
        myCar.playMusic();

        ElectricBike myBike = new ElectricBike("EcoRide", 2023);
        myBike.displayInfo();
        myBike.startEngine();
        myBike.pedal();
    }
}
```

OUTPUT:

```
Model: Toyota, Year: 2022
Toyota car engine started
Toyota is driving
Playing music in Toyota
Model: EcoRide, Year: 2023
EcoRide electric bike motor started
EcoRide bicycle is pedaling
PS C:\Users\123sr>
```

POLYMORPHISM
CONSTRUCTOR PROGRAMS

8.a)Car

CODE:

```
class Car {
    String brand;
    Car(String carBrand) {
        brand = carBrand;
        System.out.println("Car brand: " + brand);
    }
    public static void main(String[] args) {
        Car myCar = new Car("Toyota");
    }
}
```

OUTPUT:

```
Car brand: Toyota
```

CONSTRUCTOR OVERLOADING PROGRAMS

9.a)Student

CODE:

```
class Student{
    String name;
    int rollNo;
    int age;
    Student(String studentName){
        name = studentName;
        System.out.println("Student Name: " + name);
    }

    Student(String studentName, int studentRollNo){
        name = studentName;
        rollNo = studentRollNo;
        System.out.println("Student Name: "+name+", Student rollNo: "+rollNo);
    }

    Student(String studentName, int studentRollNo, int studentAge){
        name = studentName;
        rollNo=studentRollNo;
        age=studentAge;
        System.out.println("Student Name: "+name+", Student rollNo: "+rollNo+",
Student age: "+age);
    }
    public static void main(String[] args){
        Student s1 = new Student("Preetham");
        Student s2 = new Student("Raghav",155);
        Student s3 = new Student("Srini",146, 18);
    }
}
```

OUTPUT:

```
Student Name: Preetham
Student Name: Raghav, Student rollNo: 155
Student Name: Srini, Student rollNo: 146, Student age: 18
PS C:\Users\123sr>
```

METHOD OVERLOADING PROGRAMS

10.a)Payment Processor

CODE:

```
class PaymentProcessor {
    void makePayment(int amount) {
        System.out.println("Payment of Rupees " + amount + " made using cash.");
    }
    void makePayment(long cardNumber, int amount) {
        System.out.println("Payment of Rupees " + amount + " made using Credit Card
(Card Number: " + cardNumber + ").");
    }
    void makePayment(String upiID, double amount) {
        System.out.println("Payment of Rupees " + amount + " made using UPI (UPI ID:
" + upiID + ").");
    }
    public static void main(String[] args) {
        PaymentProcessor pp = new PaymentProcessor();
        pp.makePayment(500);
        pp.makePayment(1234567890123456L, 1500);
        pp.makePayment("user@upi", 999.99);
    }
}
```

OUTPUT:

```
Payment of Rupees 500 made using cash.
Payment of Rupees 1500 made using Credit Card (Card Number: 1234567890123456).
Payment of Rupees 999.99 made using UPI (UPI ID: user@upi).
PS C:\Users\123sr>
```

10.b)Calculate Volume

CODE:

```
class volume {

    double calculateVolume(int cubeSide) {
        return cubeSide * cubeSide * cubeSide;
    }

    double calculateVolume(int radius, int height) {
        return Math.PI * radius * radius * height;
    }

    double calculateVolume(int length, int width, int height) {
        return length * width * height;
    }

    public static void main(String args[]) {
        volume vc = new volume();
        System.out.println("Volume of Cube: " + vc.calculateVolume(5));
        System.out.println("Volume of Cylinder: " + vc.calculateVolume(3, 7));
        System.out.println("Volume of Cuboid: " + vc.calculateVolume(4, 5, 6));
    }
}
```

OUTPUT:

```
Volume of Cube: 125.0
Volume of Cylinder: 197.92033717615698
Volume of Cuboid: 120.0
PS C:\Users\123sr>
```

METHOD OVERRIDING PROGRAMS

11.a)Animal Sounds

CODE:

```
class Animal {
    void makeSound() {
        System.out.println("Some generic animal sound");
    }
}

class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Dog barks: Woof woof!");
    }
}

class Cat extends Animal {
    @Override
    void makeSound() {
        System.out.println("Cat meows: Meow meow!");
    }
}

class Cow extends Animal {
    @Override
    void makeSound() {
        System.out.println("Cow moos: Moo moo!");
    }
}

public class AnimalSounds {
    public static void main(String[] args) {
        Animal myAnimal;

        myAnimal = new Dog();
        myAnimal.makeSound();

        myAnimal = new Cat();
        myAnimal.makeSound();

        myAnimal = new Cow();
        myAnimal.makeSound();
    }
}
```

OUTPUT:

```
Dog barks: Woof woof!
Cat meows: Meow meow!
Cow moos: Moo moo!
PS C:\Users\123sr>
```

11.b)Employee System

CODE:

```
class Employee{
    void getSalary(){
        System.out.println("Employee Salary");
    }
}
class Manager extends Employee{
    @Override
    void getSalary(){
        System.out.println("Manager Salary: $80,000 per year");
    }
}
class Developer extends Employee{
    @Override
    void getSalary(){
        System.out.println("Developer Salary: $60,000 per year");
    }
}
class Intern extends Employee{
    @Override
    void getSalary(){
        System.out.println("Intern Salary: $15,000 per year");
    }
}

public class EmployeeTest{
    public static void main(String args[]){
        Employee emp;
        emp = new Manager();
        emp.getSalary();

        emp = new Developer();
        emp.getSalary();

        emp = new Intern();
        emp.getSalary();
        }
}
```

OUTPUT:

```
Manager Salary: $80,000 per year
Developer Salary: $60,000 per year
Intern Salary: $15,000 per year
PS C:\Users\123sr>
```

ABSTRACTION
INTERFACE PROGRAMS

12.a) Banking System

CODE:

```
interface Banking {
    void deposit(double amount);
    void withdraw(double amount);
}

class SavingsAccount implements Banking {
    private double balance;

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

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: $" + amount + " | New Balance: $" + balance);
    }

    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: $" + amount + " | Remaining Balance: $" +
balance);
        } else {
            System.out.println("Insufficient balance!");
        }
    }
}

class CurrentAccount implements Banking {
    private double balance;

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

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: $" + amount + " | New Balance: $" + balance);
    }

    public void withdraw(double amount) {
        balance -= amount;
        System.out.println("Withdrawn: $" + amount + " | Remaining Balance: $" +
balance);
    }
}
```

```
public class Bank {
    public static void main(String[] args) {
        Banking savings = new SavingsAccount(1000);
        Banking current = new CurrentAccount(2000);

        savings.deposit(500);
        savings.withdraw(2000);

        current.deposit(1000);
        current.withdraw(2500);
    }
}
```

OUTPUT:

```
Deposited: $500.0 | New Balance: $1500.0
Insufficient balance!
Deposited: $1000.0 | New Balance: $3000.0
Withdrawn: $2500.0 | Remaining Balance: $500.0
PS C:\Users\123sr>
```

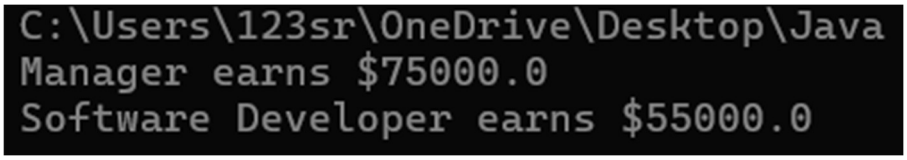
12.b) Employee System

CODE:

```
interface Employee {
    double getSalary();
    String getRole();
}

class Manager implements Employee {
    public double getSalary() {
        return 75000;
    }

    public String getRole() {
        return "Manager";
    }
}

class Developer implements Employee {
    public double getSalary() {
        return 55000;
    }

    public String getRole() {
        return "Software Developer";
    }
}

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

```
        Employee manager = new Manager();
        Employee developer = new Developer();

        System.out.println(manager.getRole() + " earns $" +
manager.getSalary());
        System.out.println(developer.getRole() + " earns $" +
developer.getSalary());
    }
}
```

OUTPUT:



12.c) Shapes

CODE:

```
interface Shape {
    double calculateArea();
}

class Circle implements Shape {
    private double radius;

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

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

class Rectangle implements Shape {
    private double length, width;

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

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

public class Shapes{
    public static void main(String[] args) {
        Shape circle = new Circle(5);
        Shape rectangle = new Rectangle(4, 6);

        System.out.println("Circle Area: " + circle.calculateArea());
        System.out.println("Rectangle Area: " + rectangle.calculateArea());
    }
}
```

OUTPUT:

```
Circle Area: 78.53981633974483
Rectangle Area: 24.0
PS C:\Users\123sr>
```

12.d) Vehicles

CODE:

```
interface Vehicle {
    void start();
    void stop();
}

class Car implements Vehicle {
    public void start() {
        System.out.println("Car is starting...");
    }

    public void stop() {
        System.out.println("Car is stopping...");
    }
}

class Bike implements Vehicle {
    public void start() {
        System.out.println("Bike is starting...");
    }

    public void stop() {
        System.out.println("Bike is stopping...");
    }
}

public class Vehicles {
    public static void main(String[] args) {
        Vehicle car = new Car();
        Vehicle bike = new Bike();

        car.start();
        car.stop();

        bike.start();
        bike.stop();
    }
}
```

OUTPUT:

```
Car is starting...
Car is stopping...
Bike is starting...
Bike is stopping...
PS C:\Users\123sr>
```

ABSTRACT CLASS PROGRAMS

13.a)Ordering System

CODE:

```
abstract class Order {
    abstract double calculateTotal();
}

class OnlineOrder extends Order {
    private double itemPrice, shippingFee;

    OnlineOrder(double itemPrice, double shippingFee) {
        this.itemPrice = itemPrice;
        this.shippingFee = shippingFee;
    }

    double calculateTotal() {
        return itemPrice + shippingFee;
    }
}

class InStoreOrder extends Order {
    private double itemPrice, discount;

    InStoreOrder(double itemPrice, double discount) {
        this.itemPrice = itemPrice;
        this.discount = discount;
    }

    double calculateTotal() {
        return itemPrice - discount;
    }
}

public class Orders {
    public static void main(String[] args) {
        Order online = new OnlineOrder(1000, 50);
        Order inStore = new InStoreOrder(1000, 100);

        System.out.println("Online Order Total: $" + online.calculateTotal());
        System.out.println("In-Store Order Total: $" + inStore.calculateTotal());
    }
}
```

OUTPUT:

```
Online Order Total: $1050.0
In-Store Order Total: $900.0
PS C:\Users\123sr>
```

13.b)Restaurant System

CODE:

```
abstract class Restaurant {
    abstract void serveFood();
}

class VegRestaurant extends Restaurant {
    void serveFood() {
        System.out.println("Serving vegetarian dishes.");
    }
}

class NonVegRestaurant extends Restaurant {
    void serveFood() {
        System.out.println("Serving non-vegetarian dishes.");
    }
}

public class Restaurants {
    public static void main(String[] args) {
        Restaurant veg = new VegRestaurant();
        Restaurant nonVeg = new NonVegRestaurant();

        veg.serveFood();
        nonVeg.serveFood();
    }
}
```

OUTPUT:

```
Serving vegetarian dishes.
Serving non-vegetarian dishes.
PS C:\Users\123sr>
```

13.c)Ticketing System

CODE:

```
abstract class Ticket {
    abstract double getPrice();
}

class MovieTicket extends Ticket {
    double getPrice() {
        return 250;
    }
}

class BusTicket extends Ticket {
    double getPrice() {
        return 50;
    }
}

public class Tickets {
    public static void main(String[] args) {
        Ticket movie = new MovieTicket();
```

```
        Ticket bus = new BusTicket();

        System.out.println("Movie Ticket Price: $" + movie.getPrice());
        System.out.println("Bus Ticket Price: $" + bus.getPrice());
    }
}
```

OUTPUT:

```
Movie Ticket Price: $250.0
Bus Ticket Price: $50.0
PS C:\Users\123sr>
```

13.d)Vehicles

CODE:

```
abstract class Vehicle {
    abstract void start();

    void stop() {
        System.out.println("Vehicle is stopping.");
    }
}

class Car extends Vehicle {
    void start() {
        System.out.println("Car is starting with ignition.");
    }
}

class Bike extends Vehicle {
    void start() {
        System.out.println("Bike is starting with a kick.");
    }
}

public class Vehicles {
    public static void main(String[] args) {
        Vehicle car = new Car();
        Vehicle bike = new Bike();

        car.start();
        car.stop();

        bike.start();
        bike.stop();
    }
}
```

OUTPUT:

```
Car is starting with ignition.
Vehicle is stopping.
Bike is starting with a kick.
Vehicle is stopping.
PS C:\Users\123sr>
```

ENCAPSULATION
ENCAPSULATION PROGRAMS

14.a)Hospital System
CODE:

```
class Patient {
    private String patientID;
    private String name;
    private int age;
    private String disease;

    public Patient(String patientID, String name, int age, String disease) {
        this.patientID = patientID;
        this.name = name;
        setAge(age);
        this.disease = disease;
    }

    public String getPatientID() {
        return patientID;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public String getDisease() {
        return disease;
    }

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

    public void setDisease(String disease) {
        this.disease = disease;
    }

    public void displayPatientInfo() {
        System.out.println("Patient ID: " + patientID);
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Disease: " + disease);
    }
}
```

```
public class Hospital{
    public static void main(String[] args) {
        Patient p = new Patient("P123", "Kailash", 18, "Fever");
        p.displayPatientInfo();
        p.setAge(-5);
        p.setDisease("Flu");
        p.displayPatientInfo();
    }
}
```

OUTPUT:

```
Patient ID: P123
Name: Kailash
Age: 18
Disease: Fever
Age must be positive!
Patient ID: P123
Name: Kailash
Age: 18
Disease: Flu
PS C:\Users\123sr>
```

14.b)Student Info

CODE:

```
class Student {
    private String studentID;
    private String name;
    private int age;
    private double grade;

    public Student(String studentID, String name, int age, double grade) {
        this.studentID = studentID;
        this.name = name;
        setAge(age);
        setGrade(grade);
    }

    public String getStudentID() {
        return studentID;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public double getGrade() {
        return grade;
    }

    public void setAge(int age) {
        if (age >= 5 && age <= 100) {
```

```
            this.age = age;
        } else {
            System.out.println("Invalid age! Age must be between 5 and 100.");
        }
    }

    public void setGrade(double grade) {
        if (grade >= 0 && grade <= 100) {
            this.grade = grade;
        } else {
            System.out.println("Grade must be between 0 and 100!");
        }
    }

    public void displayStudentInfo() {
        System.out.println("Student ID: " + studentID);
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Grade: " + grade);
    }
}

public class StudentInfo {
    public static void main(String[] args) {
        Student s = new Student("S101", "Rohith", 18, 92);
        s.displayStudentInfo();
        s.setAge(3);
        s.setGrade(105);
    s.setAge(19);
    s.setGrade(99);
        s.displayStudentInfo();
    }
}
```

OUTPUT:

```
Student ID: S101
Name: Rohith
Age: 18
Grade: 92.0
Invalid age! Age must be between 5 and 100.
Grade must be between 0 and 100!
Student ID: S101
Name: Rohith
Age: 19
Grade: 99.0
PS C:\Users\123sr>
```

14.c)Book Info

CODE:

```
class Book {
    private String title;
    private String author;
    private double price;

    public Book(String title, String author, double price) {
        this.title = title;
        this.author = author;
        setPrice(price);
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        if (price > 0) {
            this.price = price;
        } else {
            System.out.println("Price cannot be negative!");
        }
    }

    public void displayBookInfo() {
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Price: $" + price);
    }
}

public class BookInfo {
    public static void main(String[] args) {
        Book b = new Book("Fight Club", "Chuck Palahniuk", 499.99);
        b.displayBookInfo();
        b.setPrice(-100);
        b.setPrice(599.99);
        b.displayBookInfo();
    }
}
```

OUTPUT:

```
Title: Fight Club
Author: Chuck Palahniuk
Price: $499.99
Price cannot be negative!
Title: Fight Club
Author: Chuck Palahniuk
Price: $599.99
PS C:\Users\123sr>
```

14.d)Banking System

CODE:

```
class BankAccount {
    private String accountNumber;
    private double balance;
    private String ownerName;

    public BankAccount(String accountNumber, String ownerName, double balance) {
        this.accountNumber = accountNumber;
        this.ownerName = ownerName;
        this.balance = balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public String getOwnerName() {
        return ownerName;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: $" + amount);
        } else {
            System.out.println("Deposit amount must be positive!");
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: $" + amount);
        } else {
            System.out.println("Insufficient balance or invalid amount!");
        }
    }
```

```
    public void displayAccountInfo() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Owner Name: " + ownerName);
        System.out.println("Current Balance: $" + balance);
    }
}

public class Bank {
    public static void main(String[] args) {
        BankAccount myAccount = new BankAccount("123456789", "Rohith", 5000.0);
        myAccount.displayAccountInfo();
        myAccount.deposit(2000);
        myAccount.withdraw(1500);
        myAccount.withdraw(7000);
        myAccount.displayAccountInfo();
    }
}
```

OUTPUT:

```
Deposited: $500.0 | New Balance: $1500.0
Insufficient balance!
Deposited: $1000.0 | New Balance: $3000.0
Withdrawn: $2500.0 | Remaining Balance: $500.0
PS C:\Users\123sr>
```

PACKAGES PROGRAMS

15.a)Calculates Area

CODE:

Area.java

```
import shapes.Circle;
import shapes.Rectangle;

public class Area {
    public static void main(String[] args) {
        Circle c = new Circle(5);
        Rectangle r = new Rectangle(4, 6);

        System.out.println("Circle Area: " + c.getArea());
        System.out.println("Rectangle Area: " + r.getArea());
    }
}
```

Circle.java

```
package shapes;

public class Circle {
    private double radius;

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

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

Rectangle.java

```
package shapes;

public class Rectangle {
    private double length, width;

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

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

OUTPUT:

```
Circle Area: 78.53981633974483
Rectangle Area: 24.0
PS C:\Users\123sr>
```

15.b)Banking System

CODE:

Bank.java

```
import banking.BankAccount;

public class Bank {
    public static void main(String[] args) {
        BankAccount acc = new BankAccount("Rohith S", 1000);

        acc.deposit(500);
        acc.withdraw(300);
        acc.displayBalance();
    }
}
```

BankAccount.java

```
package banking;

public class BankAccount {
    private String accountHolder;
    private double balance;

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

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: $" + amount);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: $" + amount);
        } else {
            System.out.println("Insufficient balance or invalid amount.");
        }
    }

    public void displayBalance() {
```

```
        System.out.println(accountHolder + "'s Account Balance: $" + balance);
    }
}
```

OUTPUT:

```
Deposited: $500.0
Withdrawn: $300.0
Rohith S's Account Balance: $1200.0
PS C:\Users\123sr>
```

15.c)Task Logger

CODE:

```
import java.util.Scanner;
import java.time.LocalDateTime;

public class TaskLogger {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        StringBuilder taskLog = new StringBuilder();
        System.out.print("Enter the number of tasks: ");
        int numTasks = scanner.nextInt();
        scanner.nextLine();
        for (int i = 1; i <= numTasks; i++) {
            System.out.print("Enter Task " + i + ": ");
            String task = scanner.nextLine();
            LocalDateTime timestamp = LocalDateTime.now();
            taskLog.append("Task: ").append(task).append(" | Time:
").append(timestamp).append("\n");
        }
        System.out.println("\n--- Task Log ---");
        System.out.println(taskLog.toString());
        scanner.close();
    }
}
```

OUTPUT:

```
Enter the number of tasks: 2
Enter Task 1: Java Lab Exercises
Enter Task 2: UID CAPSTONE

--- Task Log ---
Task: Java Lab Exercises | Time: 2025-04-05T02:22:06.694167600
Task: UID CAPSTONE | Time: 2025-04-05T02:22:10.187225500

PS C:\Users\123sr>
```

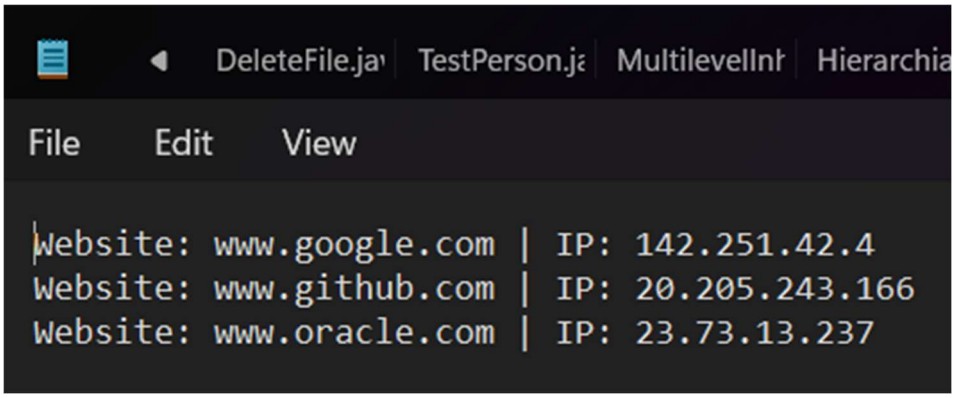
15.d)Website IP

CODE:

```
import java.util.ArrayList;
import java.net.InetAddress;
import java.io.FileWriter;
import java.io.IOException;

public class BuiltIn {
    public static void main(String[] args) {
        ArrayList<String> websites = new ArrayList<>();
        websites.add("www.google.com");
        websites.add("www.github.com");
        websites.add("www.oracle.com");

        try {
            FileWriter writer = new FileWriter("WebsiteIPs.txt");

            for (String site : websites) {
                InetAddress address = InetAddress.getByName(site);
                String ip = address.getHostAddress();
                System.out.println("Website: " + site + " | IP: " + ip);

                writer.write("Website: " + site + " | IP: " + ip + "\n");
            }

            writer.close();
            System.out.println("Data written to WebsiteIPs.txt");

        } catch (IOException e) {
            System.out.println("File writing error: " + e.getMessage());
        } catch (Exception e) {
            System.out.println("Network error: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
Website: www.google.com | IP: 142.251.42.100
Website: www.github.com | IP: 20.205.243.166
Website: www.oracle.com | IP: 23.73.13.237
Data written to WebsiteIPs.txt
PS C:\Users\123sr>
```



EXCEPTION HANDLING PROGRAMS

16.a)Bank Transaction

CODE:

```
import java.util.Scanner;
class InsufficientFundsException extends Exception {
    public InsufficientFundsException(String message) {
        super(message);
    }
}

public class BankTransaction {
    private double balance;

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

    public void withdraw(double amount) throws InsufficientFundsException {
        if (amount > balance) {
            throw new InsufficientFundsException("Insufficient balance! Available: " +
balance);
        }
        balance -= amount;
        System.out.println("Withdrawal successful! Remaining balance: " + balance);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        BankTransaction account = new BankTransaction(5000);

        try {
            System.out.print("Enter amount to withdraw: ");
            double amount = scanner.nextDouble();
            account.withdraw(amount);

        } catch (InsufficientFundsException e) {
            System.out.println("Transaction Failed: " + e.getMessage());
        } finally {
            System.out.println("Thank you for using our banking service.");
        }

        scanner.close();
    }
}
```

OUTPUT:

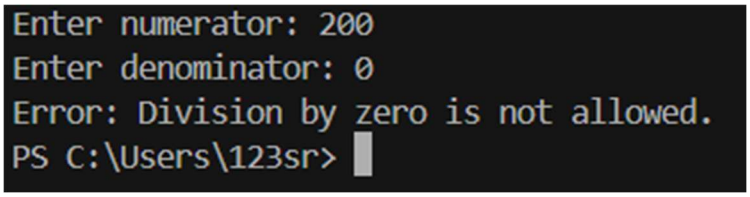
```
Enter amount to withdraw: 10000
Transaction Failed: Insufficient balance! Available: 5000.0
Thank you for using our banking service.
PS C:\Users\123sr>
```

16.b)Divide By Zero

CODE:

```
import java.util.Scanner;

public class DivideByZero {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter numerator: ");
            int num = scanner.nextInt();

            System.out.print("Enter denominator: ");
            int den = scanner.nextInt();

            int result = num / den;
            System.out.println("Result: " + result);

        } catch (ArithmeticException e) {
            System.out.println("Error: Division by zero is not allowed.");
        }

        scanner.close();
    }
}
```

OUTPUT:

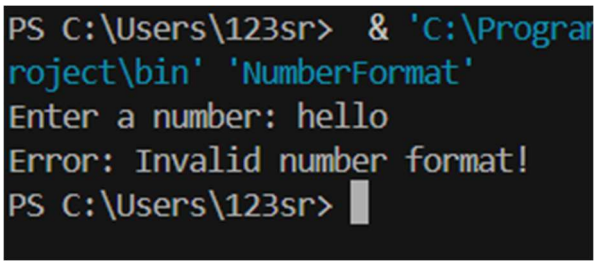


16.c)Number Format

CODE:

```
import java.util.Scanner;

public class NumberFormat {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter a number: ");
            String input = scanner.nextLine();
            int num = Integer.parseInt(input);
            System.out.println("You entered: " + num);

        } catch (NumberFormatException e) {
            System.out.println("Error: Invalid number format!");
        }

        scanner.close();
    }
}
```

OUTPUT:



16.d)Password Validation

CODE:

```
import java.util.Scanner;
class WeakPasswordException extends Exception {
    public WeakPasswordException(String message) {
        super(message);
    }
}
public class PasswordValidation {
    public static void validatePassword(String password) throws WeakPasswordException {
        if (password.length() < 8 || !password.matches(".*\\d.*")) {
            throw new WeakPasswordException("Password must be at least 8 characters long and contain a
number!");
        }
        System.out.println("Password is strong!");
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter a password: ");
            String password = scanner.next();
            validatePassword(password);
        } catch (WeakPasswordException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            System.out.println("Password validation completed.");
            scanner.close();
        }
    }
}
```

OUTPUT:

```
Enter a password: hello
Error: Password must be at least 8 characters long and contain a number!
Password validation completed.
PS C:\Users\123sr>
```

FILE HANDLING PROGRAMS

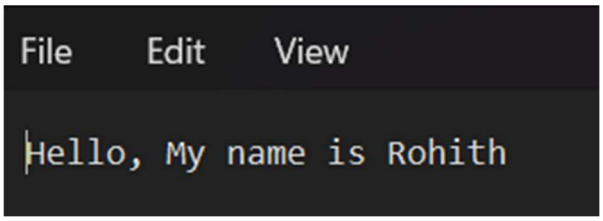
17.a) Write To file

CODE:

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

public class WriteToFile {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("example.txt");
            writer.write("Hello, My name is Rohith");
            writer.close();
            System.out.println("Successfully written to the file.");
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
Successfully written to the file.
PS C:\Users\123sr>
```



17.b)Read from file

CODE:

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

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

OUTPUT:

```
C:\Users\123sr\OneDrive\Desktop\Java Programs\FILE HANDLING>java ReadFromFile.java
Hello, My name is Rohith
```

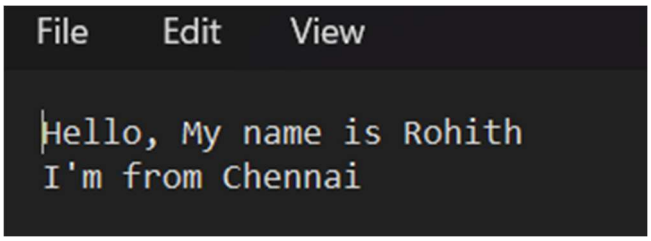
17.c)Append to file

CODE:

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

public class AppendToFile {
    public static void main(String[] args) {
        try {
            FileWriter writer = new FileWriter("example.txt", true);
            writer.write("\nI'm from Chennai");
            writer.close();
            System.out.println("Successfully appended to the file.");
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

OUTPUT:

```
C:\Users\123sr\OneDrive\Desktop\Java Progr
Successfully appended to the file.
```



17.d)Delete file

CODE:

```
import java.io.File;

public class DeleteFile {
    public static void main(String[] args) {
        File file = new File("example.txt");
        if (file.delete()) {
            System.out.println("File deleted successfully.");
        } else {
            System.out.println("Failed to delete the file.");
        }
    }
}
```

OUTPUT: