RAAGHAV VEL P

CH.SC.U4CSE24155

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.U4CSE24155 – RAAGHAV VEL P*** 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

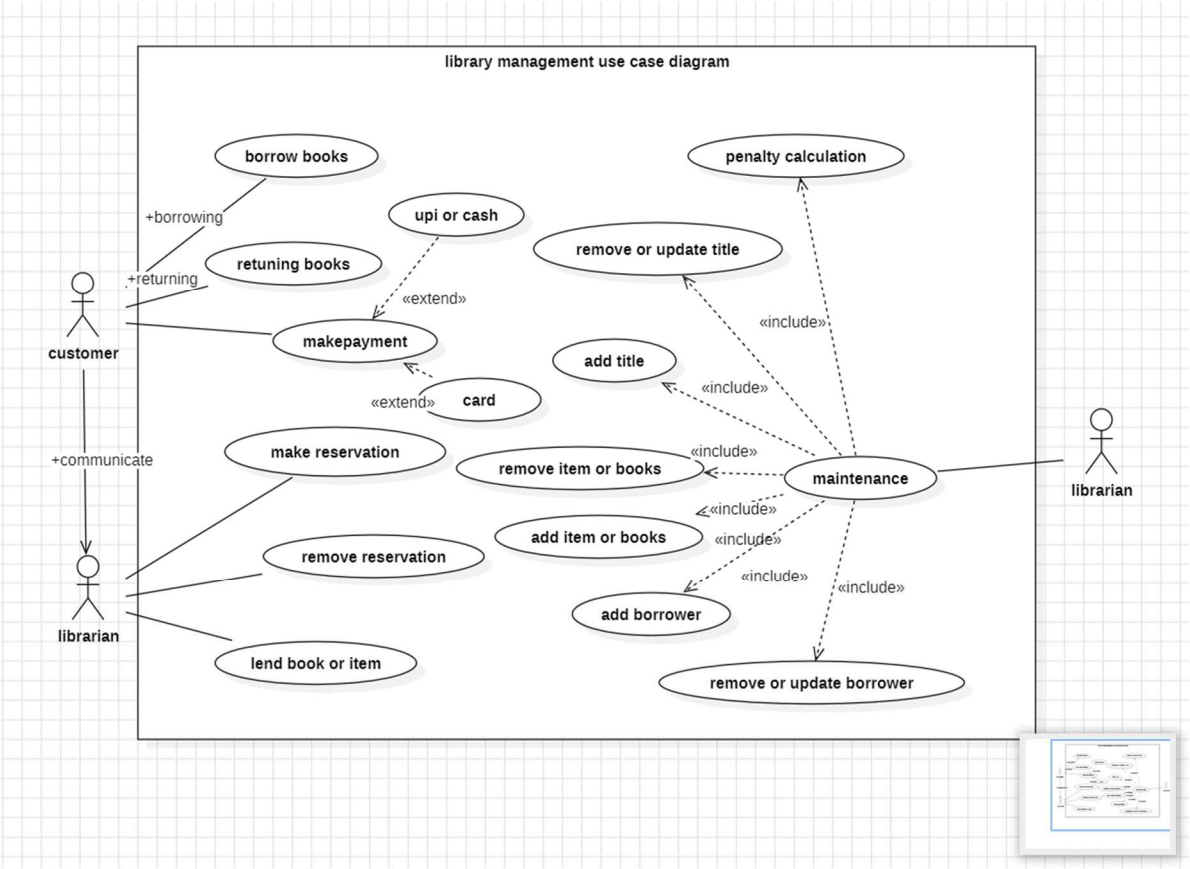
Internal Examiner 1 Internal Examiner 2

INDEX

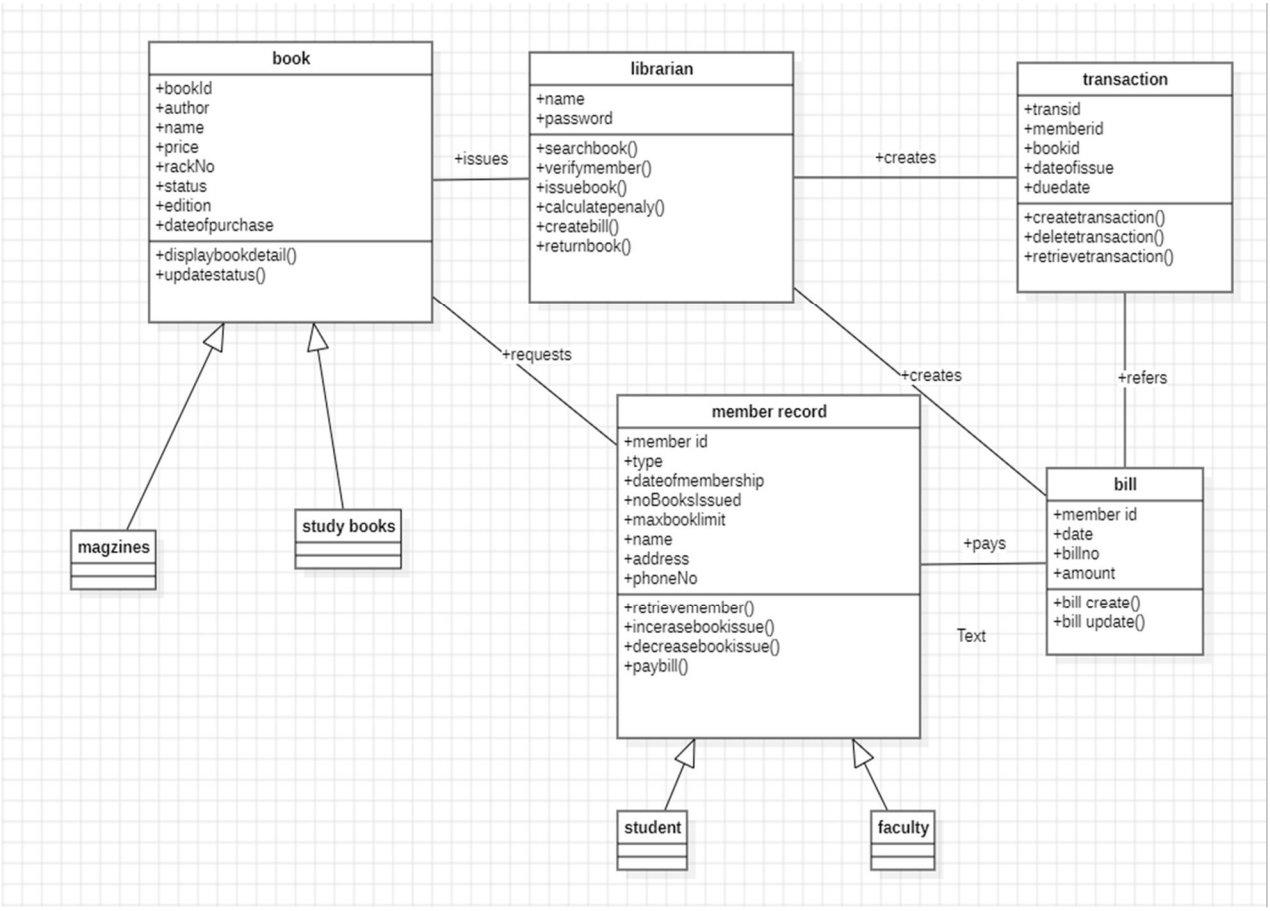
UML DIAGRAMS

1. LIBRARY MANAGEMENT

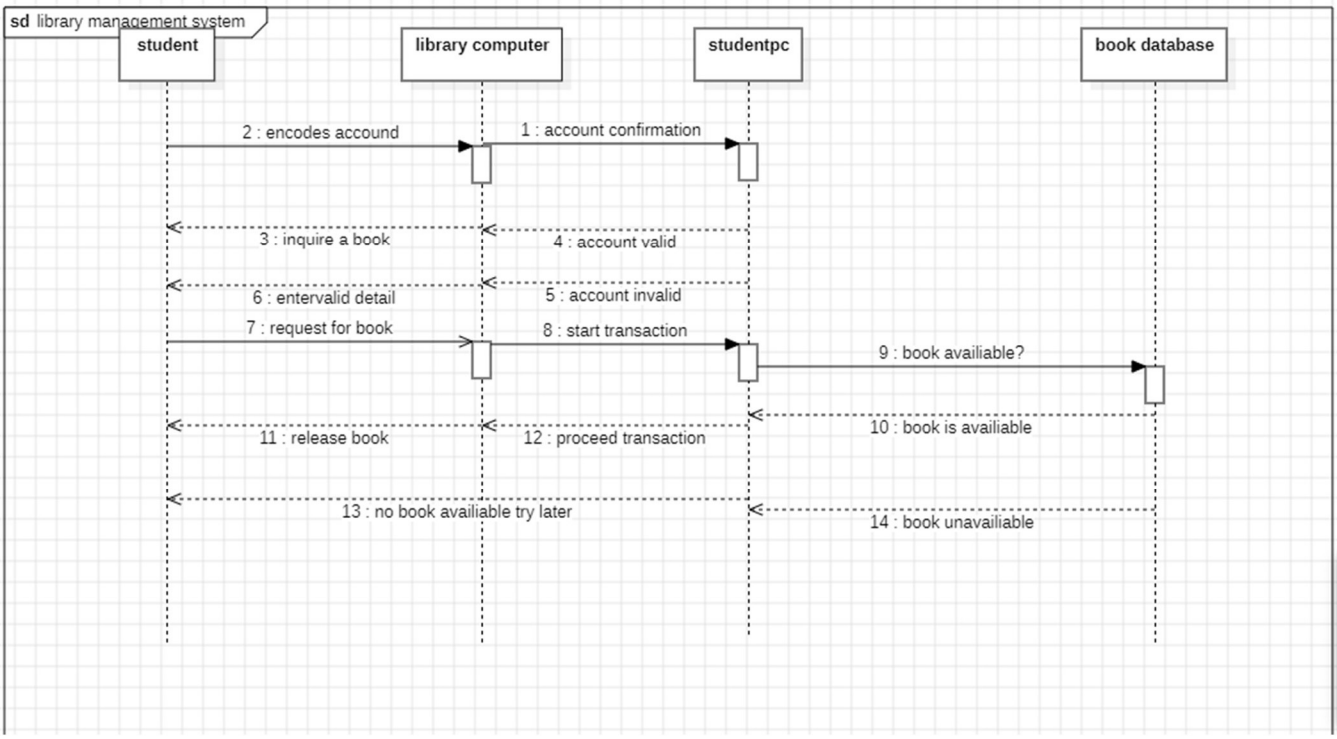
1.a) Use Case Diagram:



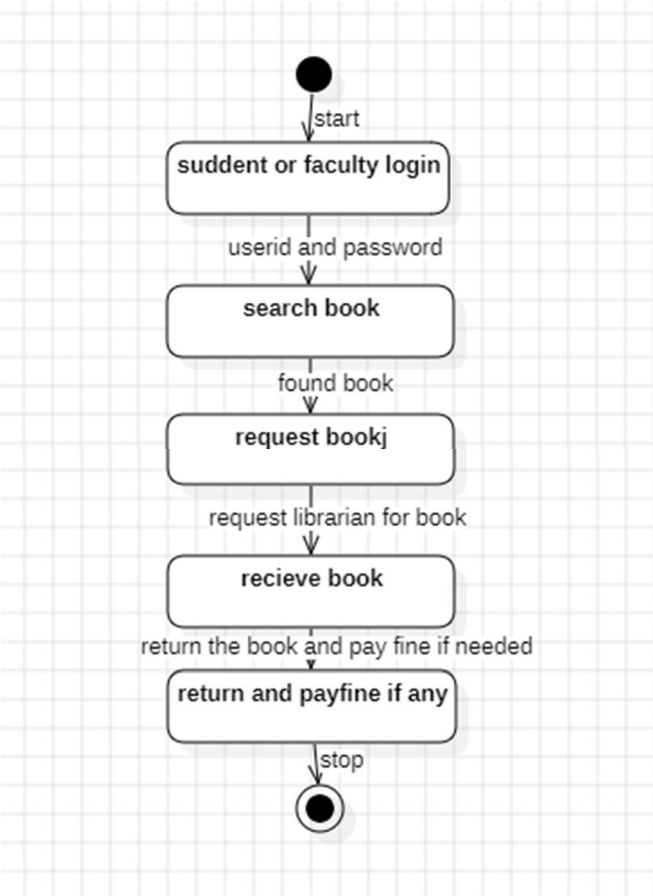
1.b) Class Diagram:



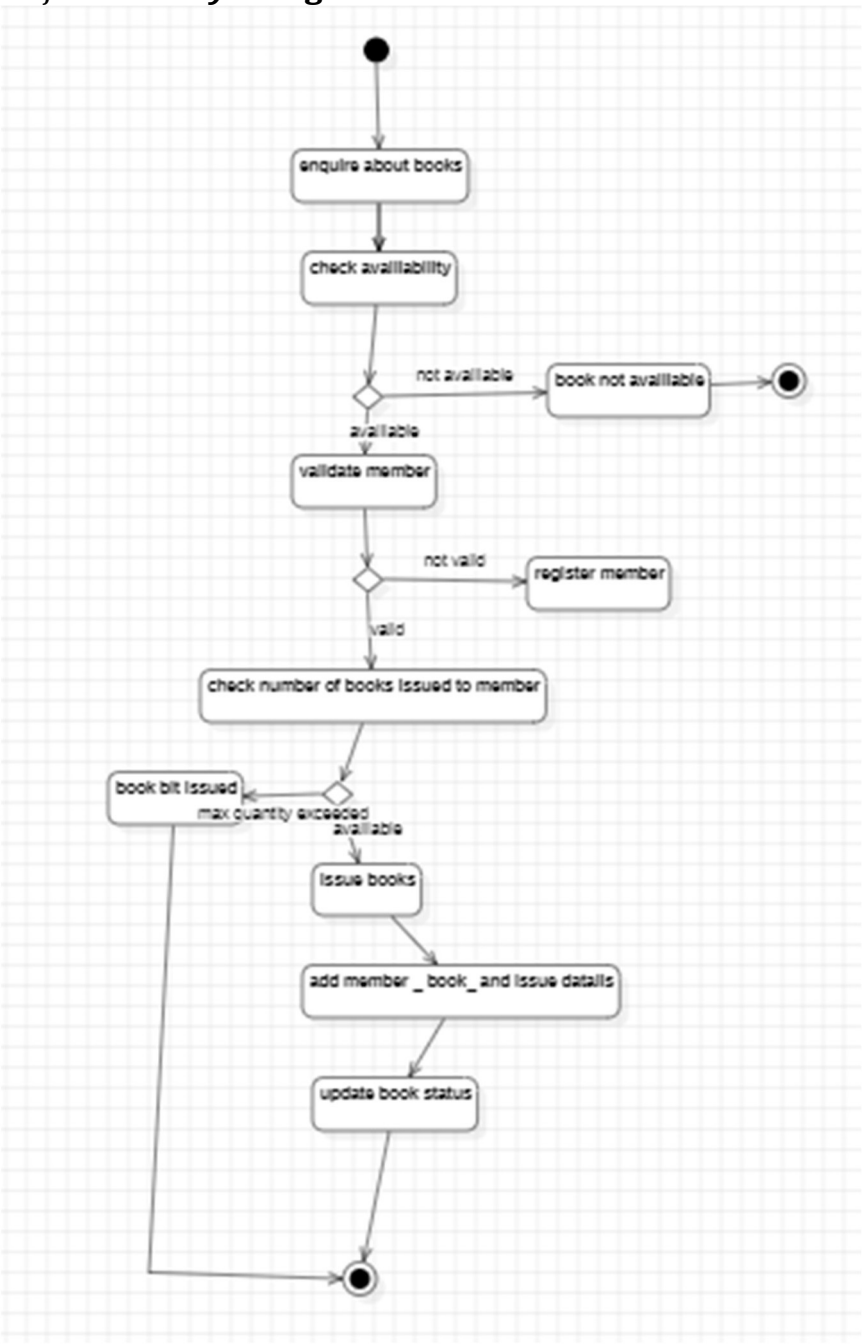
1.c) Sequence Diagram:



1.d) **Statechart diagram:**

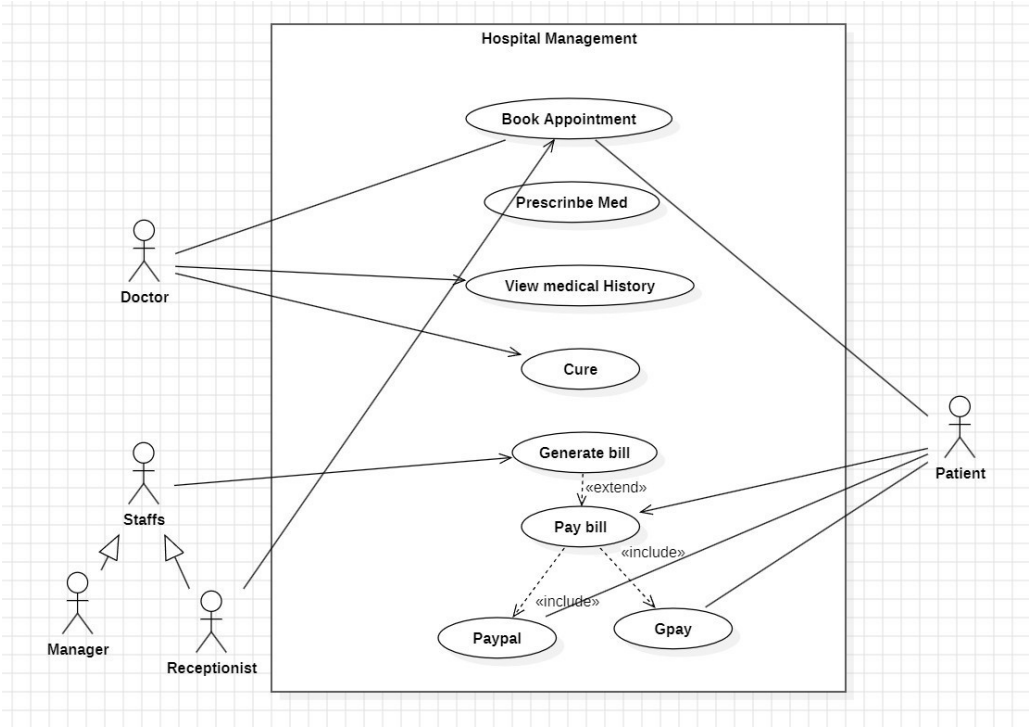


1.e) **Activity Diagram:**

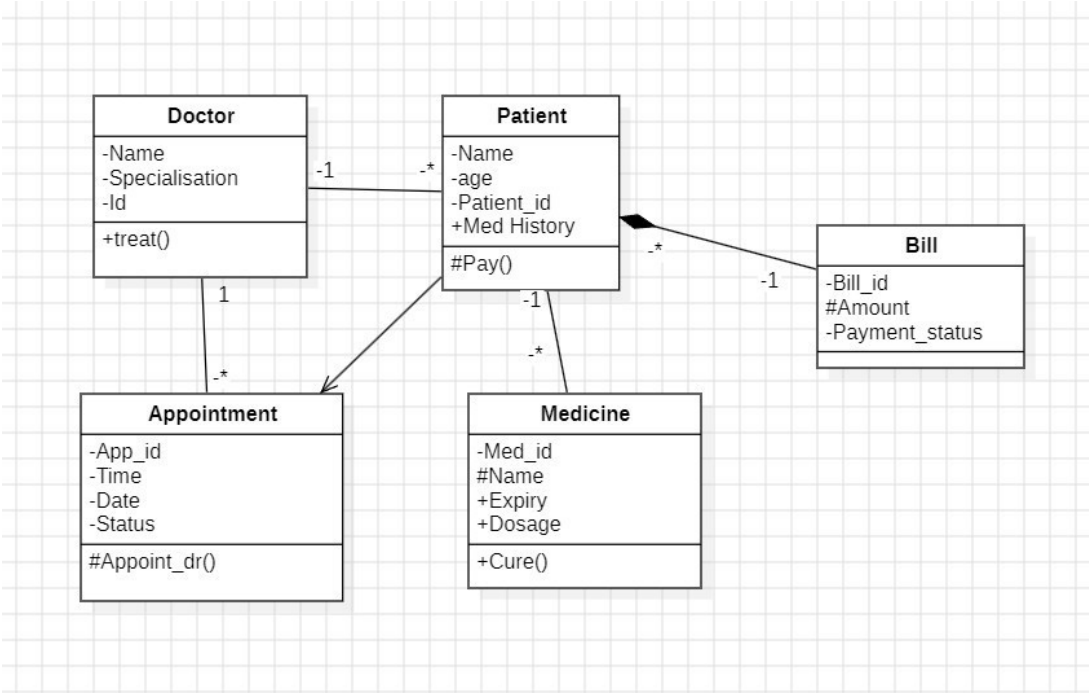


2. Hospital Management

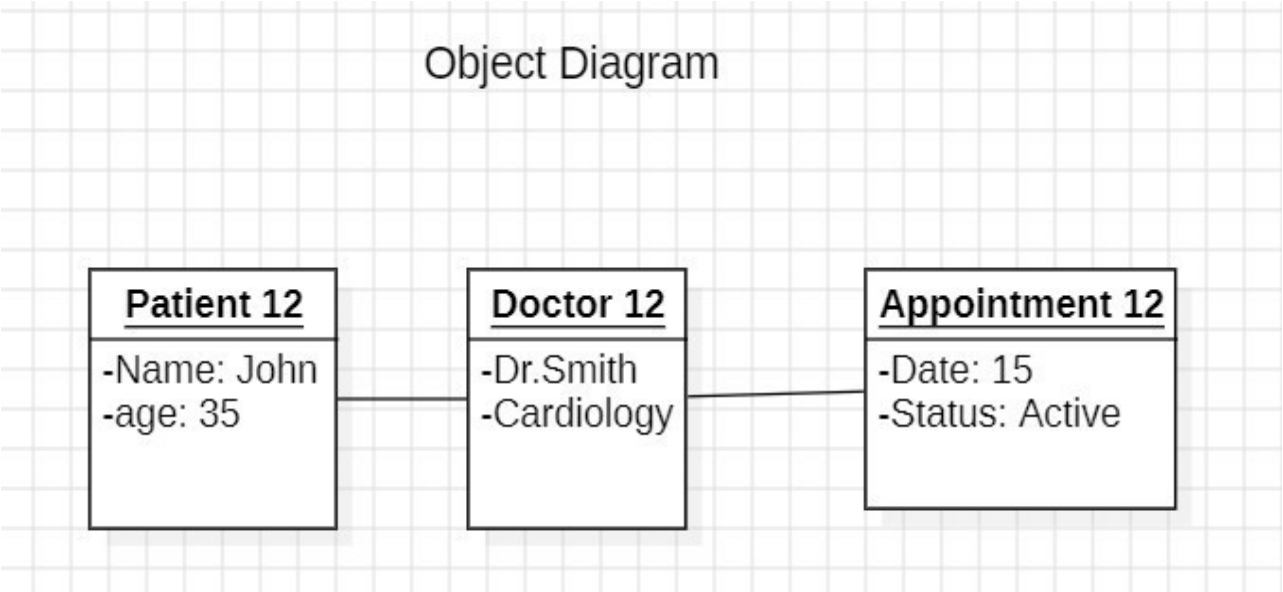
2.a) Use Case Diagram:



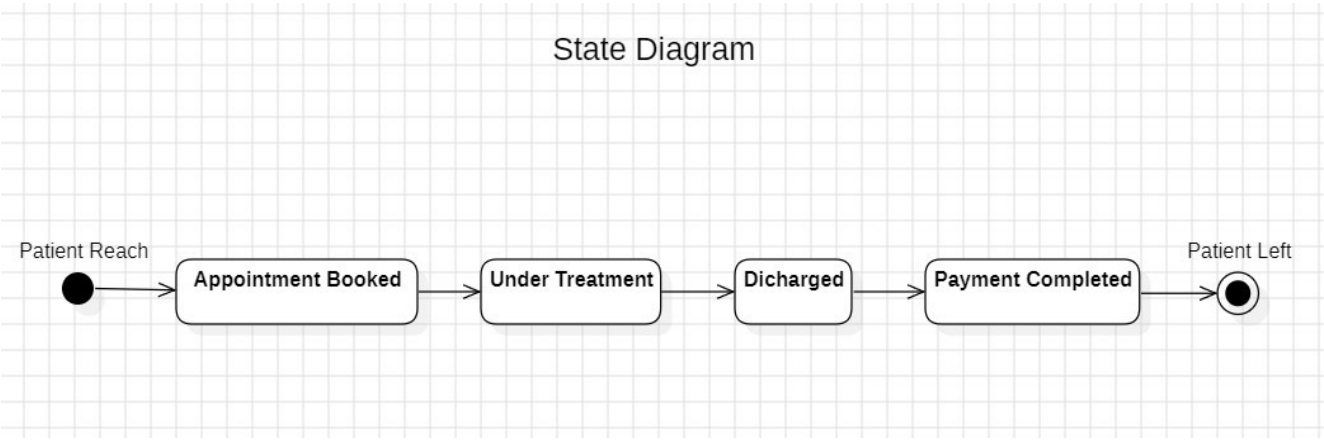
2.b) Class Diagram:



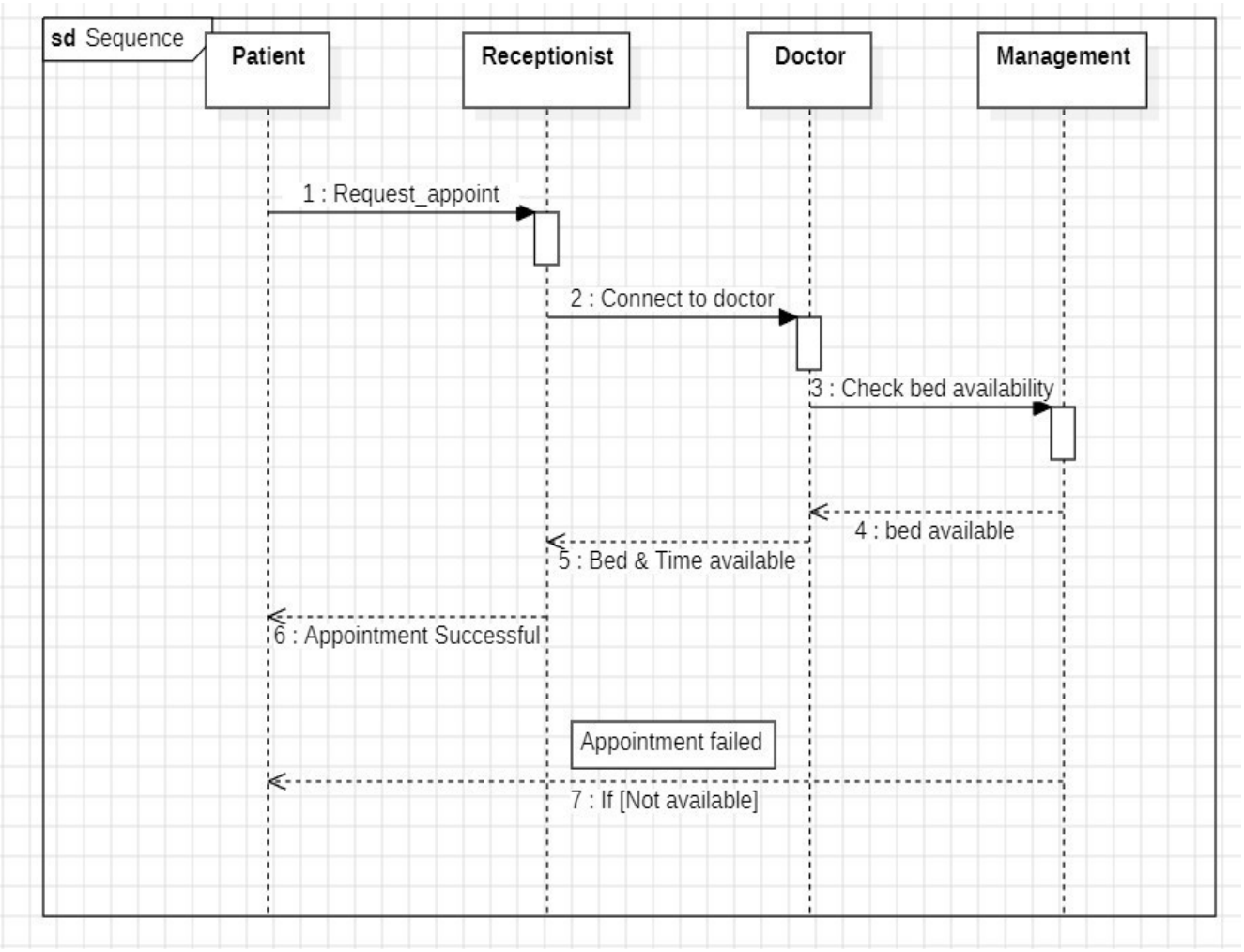
2c) Object Diagram:



2d)State Diagram:



2e)Sequence Diagram:



Basic Java Questions

3a) Even Or Odd with Scanner:

Code:

```
import java.util.Scanner;
public class even{
    public void find(int a){
        if(a>=0){
            if (a%2==0){
                System.out.println("It is even!");
            }
            else{
                System.out.println("It is odd!");
            }}
        else{
            System.out.println("Enter a number greater than or equalt to
0!!");
        }
    }
    public static void main(String[]args){
        Scanner ip = new Scanner(System.in);
        even ob1=new even();
        System.out.print("Enter no to check: ");
        int a=ip.nextInt();
        ob1.find(a);
    }}
```

Output:

```
C:\2nd sem\oop>java Even.java
Enter no to check: 23
It is odd!
```

3b) Count Number Of Digits :

Code:

```
import java.util.Scanner;

public class CountDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        int count = 0;
        while (num != 0) {
            num /= 10;
            count++;
        }

        System.out.println("Number of digits: " + count);
        scanner.close();
    }
}
```

Output:

```
C:\2nd sem\oop>java CountDigits.java
Enter a number: 1234
Number of digits: 4
```

3c) Factorial:

Code:

```
import java.util.Scanner;

public class FactorialLoop {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        int factorial = 1;
        for (int i = 1; i <= num; i++) {
            factorial *= i;
        }

        System.out.println("Factorial of " + num + " is " + factorial);
        scanner.close();
    }
}
```

Output:

```
C:\2nd sem\oop>java FactorialLoop.java
Enter a number: 5
Factorial of 5 is 120
```

3d) Fibonacci Series:

Code:

Code:

```
import java.util.Scanner;

public class FibonacciSeries{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        int factorial = 1;
        for (int i = 1; i <= num; i++) {
            factorial *= i;
        }

        System.out.println("Factorial of " + num + " is " + factorial);
        scanner.close();
    }
}
```

Output:

```
Enter a number: 10
Fibonacci Series for 10 terms:
0,1,1,2,3,5,8,13,21,34,55
```

3e) Largest Number Calculator:

Code:

```
import java.util.Scanner;

public class Largest{
    int a,b,c;
        void lar(int a,int b, int c){
            if(a>b && a>c){
                System.out.println(a + "Is the largest among 3");
            }
            else if(b>a && b>c){
                System.out.println(b + "Is the largest among 3");
            }
            else if(c>a && c>b){
                System.out.println(c + "Is the largest among 3");
            }
            else{
                System.out.println("All are equal no larger number");
            }
        }
    }

class call{
    public static void main(String[]args){
        Largest l1= new Largest();
        Scanner ip=new Scanner(System.in);
        System.out.println("Enter Number 1: ");
        int a=ip.nextInt();
        System.out.println("Enter Number 2: ");
        int b=ip.nextInt();
        System.out.println("Enter Number 3: ");
        int c=ip.nextInt();
        l1.lar(a,b,c);
    }
}
```

Output:

```
C:\2nd sem\oop>java Call.java
Enter Number 1:
5
Enter Number 2:
4
Enter Number 3:
8
8Is the largest among 3
```

3f) Multiplication Table :

Code:

```
import java.util.Scanner;

public class MultiplicationTable {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        for (int i = 1; i <= 10; i++) {
            System.out.println(num + " x " + i + " = " + (num * i));
        }

        scanner.close();
    }
}
```

Output:

```
C:\2nd sem\oop>java MultiplicationTable.java
Enter a number: 6
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
```

3g) Prime Check:

Code:

```
import java.util.Scanner;

public class PrimeCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();
        boolean isPrime = true;

        if (num <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i <= num / 2; i++) {
                if (num % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }

        if (isPrime)
            System.out.println(num + " is a prime number.");
        else
            System.out.println(num + " is not a prime number.");

        scanner.close();
    }
}
```

Output:

```
C:\2nd sem\oop>java PrimeCheck.java
Enter a number: 6
6 is not a prime number.
```

3h) Reverse Number:

Code:

```
import java.util.Scanner;
public class ReverseNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        int reversed = 0;
        while (num != 0) {
            int digit = num % 10;
            reversed = reversed * 10 + digit;
            num /= 10;
        }

        System.out.println("Reversed Number: " + reversed);
        scanner.close();
    }
}
```

Output:

```
C:\2nd sem\oop>java ReverseNumber.java
Enter a number: 12456
Reversed Number: 65421
```

3i) Sum Of N Natural Numbers:

Code:

```
import java.util.Scanner;

public class SumNaturalNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = scanner.nextInt();

        int sum = 0, i = 1;
        while (i <= n) {
            sum += i;
            i++;
        }

        System.out.println("Sum of first " + n + " natural numbers is " +
sum);
        scanner.close();
    }
}
```

Output:

```
C:\2nd sem\oop>java SumNaturalNumbers.java
Enter a number: 6
Sum of first 6 natural numbers is 21
```

3j) Sum of Digits:

Code:

```
import java.util.Scanner;

public class SumOfDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int num = scanner.nextInt();

        int sum = 0;
        while (num != 0) {
            sum += num % 10;
            num /= 10;
        }

        System.out.println("Sum of digits: " + sum);
        scanner.close();
    }
}
```

Output:

```
C:\2nd sem\oop>java SumOfDigits.java
Enter a number: 12203
Sum of digits: 8
```

INHERITANCE

SINGLE INHERITANCE PROGRAMS:

4a) Employee and Manager

Code:

```
class Employee {
    String name;
    double salary;

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

    void displayDetails() {
        System.out.println("Employee Name: " + name);
        System.out.println("Salary: $" + salary);
    }
}

class Manager extends Employee {
    double bonus;

    Manager(String name, double salary, double bonus) {
        super(name, salary);
        this.bonus = bonus;
    }
    void displayDetails() {
        super.displayDetails();
        System.out.println("Bonus: $" + bonus);
    }
}

public class Work {
    public static void main(String[] args) {
        Manager mgr = new Manager("Alice", 60000, 5000);
        mgr.displayDetails();
    }
}
```

OUTPUT:

```
C:\2nd sem\java prc>javac Work.java

C:\2nd sem\java prc>java Work
Employee Name: Alice
Salary: $60000.0
Bonus: $5000.0
```

4b) Vehicle and Car

Code:

```
class Vehicle {
    String brand;
    int speed;

    Vehicle(String brand, int speed) {
        this.brand = brand;
        this.speed = speed;
    }

    void displayInfo() {
        System.out.println("Brand: " + brand);
        System.out.println("Speed: " + speed + " km/h");
    }
}

class Car extends Vehicle {
    String fuelType;

    Car(String brand, int speed, String fuelType) {
        super(brand, speed);
        this.fuelType = fuelType;
    }


    void displayInfo() {
        super.displayInfo();
        System.out.println("Fuel Type: " + fuelType);
    }
}

public class Automobile {
    public static void main(String[] args) {
        Car myCar = new Car("Toyota", 180, "Petrol");
        myCar.displayInfo();
    }
}
```

OUTPUT:

```
C:\2nd sem\java prc>javac Automobile.java

C:\2nd sem\java prc>java Automobile
Brand: Toyota
Speed: 180 km/h
Fuel Type: Petrol
```

MULTILEVEL INHERITANCE PROGRAMS:

5a) Person → Employee → Manager

Code:

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

    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

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

class Employee extends Person {
    double salary;

    Employee(String name, int age, double salary) {
        super(name, age);
        this.salary = salary;
    }


    void displayDetails() {
        super.displayDetails();
        System.out.println("Salary: $" + salary);
    }
}

class Manager extends Employee {
    String department;

    Manager(String name, int age, double salary, String department) {
        super(name, age, salary);
        this.department = department;
    }


    void displayDetails() {
        super.displayDetails();
        System.out.println("Department: " + department);
    }
}

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

```
    Manager mgr = new Manager("Alice", 35, 75000, "HR");
        mgr.displayDetails();
    }
}
```

OUTPUT:

```
C:\2nd sem\java prc>javac Mwork.java

C:\2nd sem\java prc>java Mwork
Name: Alice
Age: 35
Salary: $75000.0
Department: HR
```

5b) Animal → Mammal → Dog

Code:

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

class Mammal extends Animal {
    void walk() {
        System.out.println("I can walk on land");
    }
}

class Dog extends Mammal {
    void makeSound() {
        System.out.println("Bark! Bark!");
    }
}

public class Manimal {
    public static void main(String[] args) {
        Dog myDog = new Dog();
        myDog.makeSound(); // Calls overridden method in Dog
        myDog.walk();      // Calls method from Mammal
    }
}
```

OUTPUT:

```
C:\2nd sem\java prc>javac Manimal.java

C:\2nd sem\java prc>java Manimal
Bark! Bark!
I can walk on land
```

HIERARCHICAL INHERITANCE PROGRAMS

6a) vehicle functions and property

Code:

```
class Vehicle {
    private String brand;
    private String model;
     public Vehicle(String brand, String model) {
        this.brand = brand;
        this.model = model;
    }
    public void start() {
        System.out.println("Vehicle is starting.");
    }
    public void stop() {
        System.out.println("Vehicle is stopping.");
    }
    public String getBrand() {
        return brand;
    }
  public String getModel() {
        return model;
    }
}
class Car extends Vehicle {
    private int numberOfDoors ;
   public Car(String brand, String model, int numberOfDoors) {
        super(brand, model);
        this.numberOfDoors = numberOfDoors;
    }
  public void drive() {
        System.out.println("Car is driving.");
    }
    public int getNumberOfDoors() {
        return numberOfDoors;
    }
}
```

```
class ElectricCar extends Car {
    private int batteryCapacity;
   public ElectricCar(String brand, String model, int numberOfDoors, int
batteryCapacity) {
        super(brand, model, numberOfDoors);
        this.batteryCapacity = batteryCapacity;
    }
  public void charge() {
        System.out.println("Electric car is charging.");
    }
 public int getBatteryCapacity() {
        return batteryCapacity;
    }
}
class Truck extends Vehicle {
    private double cargoCapacity;
    public Truck(String brand, String model, double cargoCapacity) {
        super(brand, model);
        this.cargoCapacity = cargoCapacity;
    }
 public void loadCargo() {
        System.out.println("Truck is loading cargo.");
    }
  public double getCargoCapacity() {
        return cargoCapacity;
    }
}
public class Main {
    public static void main(String[] args) {
        Car car = new Car("Toyota", "Corolla", 4);
        car.start();
        car.drive();
        car.stop();
        System.out.println("Car doors: " + car.getNumberOfDoors());
        ElectricCar electricCar = new ElectricCar("Tesla", "Model S", 4, 100);
        electricCar.start();
        electricCar.drive();
        electricCar.charge();
        System.out.println("Battery capacity: " +
electricCar.getBatteryCapacity());
        Truck truck = new Truck("Ford", "F-150", 2000.5);
        truck.start();
        truck.loadCargo();
        truck.stop();
        System.out.println("Cargo capacity: " + truck.getCargoCapacity());
    }
}
```

Output:

```
C:\2nd sem\java prc\HIERARCHICAL INHERITANCE PROGRAMS>javac Hvehicle.java

C:\2nd sem\java prc\HIERARCHICAL INHERITANCE PROGRAMS>java Hvehicle
Vehicle is starting.
Car is driving.
Vehicle is stopping.
Car doors: 4
Vehicle is starting.
Car is driving.
Electric car is charging.
Battery capacity: 100
Vehicle is starting.
Truck is loading cargo.
Vehicle is stopping.
Cargo capacity: 2000.5
```

6b) university

Code:

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

class Department extends University {
    void departmentInfo() {
        System.out.println("Department: Computer Science and Engineering");
    }
}

class Library extends University {
    void libraryInfo() {
        System.out.println("Library: Central Library - 9AM to 8PM");
    }
}

public class Huniversity {
    public static void main(String[] args) {
        Department dept = new Department();
        dept.universityInfo();
        dept.departmentInfo();

        Library lib = new Library();
        lib.universityInfo();
        lib.libraryInfo();
```

```
 }
}
```

Output:

```
C:\2nd sem\java prc\HIERARCHICAL INHERITANCE PROGRAMS>javac Huniversity.java

C:\2nd sem\java prc\HIERARCHICAL INHERITANCE PROGRAMS>java Huniversity
University: Amrita Vishwa Vidyapeetham
Department: Computer Science and Engineering
University: Amrita Vishwa Vidyapeetham
Library: Central Library - 9AM to 8PM
```

HYBRID INHERITANCE PROGRAMS

7a) Single and Hierarchical

Code:

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


class Dog extends Animal {
    void bark() {
        System.out.println("Dog is barking");
    }
}


class Cat extends Animal {
    void meow() {
        System.out.println("Cat is meowing");
    }
}


class Puppy extends Dog {
    void weep() {
        System.out.println("Puppy is weeping");
    }
}

public class HybridInheritanceExample {
    public static void main(String[] args) {
        Puppy puppy = new Puppy();
```

```
        puppy.eat();
        puppy.bark();
        puppy.weep();

        Cat cat = new Cat();
        cat.eat();
        cat.meow();
    }
}
```

Output:

```
C:\2nd sem\java prc\HYBRID INHERITANCE PROGRAMS>javac Single_Hierarchical.java

C:\2nd sem\java prc\HYBRID INHERITANCE PROGRAMS>java Single_Hierarchical
Animal is eating
Dog is barking
Puppy is weeping
Animal is eating
Cat is meowing
```

7b) Multilevel and Hierarchical

Code:

```
class Person {
    void displayPerson() {
        System.out.println("I am a person");
    }
}

// Derived class 1 (Multilevel inheritance)
class Student extends Person {
    void displayStudent() {
        System.out.println("I am a student");
    }
}

class Teacher extends Person {
    void displayTeacher() {
        System.out.println("I am a teacher");
    }
}

class CollegeStudent extends Student {
    void displayCollegeStudent() {
        System.out.println("I am a college student");
    }
}

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

```
        CollegeStudent cs = new CollegeStudent();
        cs.displayPerson();
        cs.displayStudent();
        cs.displayCollegeStudent();

        Teacher teacher = new Teacher();
        teacher.displayPerson();
        teacher.displayTeacher();
    }
}
```

Output

```
C:\2nd sem\java prc\HYBRID INHERITANCE PROGRAMS>javac Multilevel_Hierarchical.java

C:\2nd sem\java prc\HYBRID INHERITANCE PROGRAMS>java Multilevel_Hierarchical
I am a person
I am a student
I am a college student
I am a person
I am a teacher
```

CONSTRUCTOR PROGRAMS

8a) Student

Code:

```
class Student {
    String name;
    int age;

    Student(String n, int a) {
        name = n;
        age = a;
    }

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

public class ConstructorDemo {
    public static void main(String[] args) {
        Student s1 = new Student("Alice", 20);
        Student s2 = new Student("Bob", 22);

        s1.display();
        s2.display();
    }
}
```

Output:

```
C:\2nd sem\java prc\POLYMORPHISM>javac ConstructorDemo.java

C:\2nd sem\java prc\POLYMORPHISM>java ConstructorDemo
Name: Alice, Age: 20
Name: Bob, Age: 22
```

9a) Rectangle

Code:

```
class Rectangle {
    int length, width;
```

```
    Rectangle() {
        length = 0;
        width = 0;
    }


    Rectangle(int side) {
        length = side;
        width = side;
    }


    Rectangle(int l, int w) {
        length = l;
        width = w;
    }

    int area() {
        return length * width;
    }
}

public class ConstructorOverloading {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle();
        Rectangle r2 = new Rectangle(5);
        Rectangle r3 = new Rectangle(4, 6);

        System.out.println("Area 1: " + r1.area());
        System.out.println("Area 2: " + r2.area());
        System.out.println("Area 3: " + r3.area());
    }
}
```

Output:

```
C:\2nd sem\java prc\POLYMORPHISM>javac ConstructorOverloading.java

C:\2nd sem\java prc\POLYMORPHISM>java ConstructorOverloading
Area 1: 0
Area 2: 25
Area 3: 24
```

10a) addition

Code:

```
class Calculator {
    Calculator() {
        System.out.println("Calculator initialized");
    }

    int add(int a, int b) {
        return a + b;
    }

    double add(double a, double b) {
        return a + b;
    }

    int add(int a, int b, int c) {
        return a + b + c;
    }
}

public class Method_Overloading {
    public static void main(String[] args) {
        Calculator calc = new Calculator();

        System.out.println(calc.add(5, 3));
        System.out.println(calc.add(4.2, 3.8));
        System.out.println(calc.add(1, 2, 3));
    }
}
```

Output:

```
C:\2nd sem\java prc\POLYMORPHISM>javac Method_Overloading.java

C:\2nd sem\java prc\POLYMORPHISM>java Method_Overloading
Calculator initialized
8
8.0
6
```

10b) Area Calculation

Code:

```
class Calculation {
    int area(int side) {
        return side * side;
    }

    int area(int length, int width) {
        return length * width;
    }

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

public class AreaCalculator{
    public static void main(String[] args) {
        AreaCalculator ac = new AreaCalculator();

        System.out.println("Area of Square (side=5): " + ac.area(5));
        System.out.println("Area of Rectangle (4x6): " + ac.area(4, 6));
        System.out.println("Area of Circle (radius=3.5): " + ac.area(3.5));
    }
}
```

Output:

```
C:\2nd sem\java prc\POLYMORPHISM>java AreaCalculator
Area of Square (side=5): 25
Area of Rectangle (4x6): 24
Area of Circle (radius=3.5): 38.484510006474 96
```

11a) Bank Interest Calculation

Code:

```
class Bank {
    double getInterestRate() {
        return 0.0;
    }
}

class SBI extends Bank {

    double getInterestRate() {
        return 7.5;
```

```
   }
}

class HDFC extends Bank {

    double getInterestRate() {
        return 8.0;
    }
}

class ICICI extends Bank {
    double getInterestRate() {
        return 8.5;
    }
}

public class BankInterestDemo {
    public static void main(String[] args) {
        Bank bank1 = new SBI();
        Bank bank2 = new HDFC();
        Bank bank3 = new ICICI();

        System.out.println("SBI Interest Rate: " + bank1.getInterestRate() + "%");
        System.out.println("HDFC Interest Rate: " + bank2.getInterestRate() + "%");
        System.out.println("ICICI Interest Rate: " + bank3.getInterestRate() +
"%");
    }
}
```

Output:

```
C:\2nd sem\java prc\POLYMORPHISM>javac BankInterestDemo.java

C:\2nd sem\java prc\POLYMORPHISM>java BankInterestDemo
SBI Interest Rate: 7.5%
HDFC Interest Rate: 8.0%
ICICI Interest Rate: 8.5%
```

11b)

Code:

```
class Calculator {
    int add(int a, int b) {
        return a + b;
    }
}

class AdvancedCalc extends Calculator {
    int add(int a, int b) {  // Overriding add()
```

```
        System.out.println("Adding two numbers:");
        return a + b;
    }
}

public class Main {
    public static void main(String[] args) {
        Calculator c = new AdvancedCalc();
        System.out.println(c.add(5, 3));
    }
}
```

Output:

```
Adding two numbers:
8
```

ABSTRACTION

INTERFACE PROGRAMS

12a) circle

Code:

```
interface Drawable {
    void draw();
}

class Circle implements Drawable {
    public void draw() {
        System.out.println("Drawing Circle");
    }
}

public class Circle1 {
    public static void main(String[] args) {
        Drawable d = new Circle();
        d.draw();
    }
}
```

Output:

```
C:\2nd sem\java prc\ABSTRACTION\INTERFACE PROGRAMS>javac Circle1.java

C:\2nd sem\java prc\ABSTRACTION\INTERFACE PROGRAMS>java Circle1
Drawing Circle
```

12b) Smartphone Inheriting from Multiple Interfaces

Code:

```
interface Camera {
   void takePhoto();
   default void recordVideo() {
      System.out.println("Recording video in 1080p");
   }
}


interface MusicPlayer {
```

```
    void playMusic();
    default void pauseMusic() {
        System.out.println("Music paused");
    }
}


class Smartphone implements Camera, MusicPlayer {

    public void takePhoto() {
        System.out.println("Taking a photo");
    }


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



    public void recordVideo() {
        System.out.println("Recording video in 4K");
    }
}

public class Main {
    public static void main(String[] args) {
        Smartphone myPhone = new Smartphone();

        myPhone.takePhoto();
        myPhone.recordVideo();

        myPhone.playMusic();
        myPhone.pauseMusic();
    }
}
```

Output:

```
C:\2nd sem\java prc\ABSTRACTION\INTERFACE PROGRAMS>javac Main.java

C:\2nd sem\java prc\ABSTRACTION\INTERFACE PROGRAMS>java Main
Taking a photo
Recording video in 4K
Playing music
Music paused
```

12c) XeroMachine

Code:

```
interface Printer {
   default void print() {
      System.out.println("Printing document");
   }
}

interface Scanner {
   default void scan() {
      System.out.println("Scanning document");
   }
}

class AllInOneMachine implements Printer, Scanner {
}

public class XeroMachine {
   public static void main(String[] args) {
      AllInOneMachine machine = new AllInOneMachine();
      machine.print();
      machine.scan();
   }
}
```

Output:

```
C:\2nd sem\java prc\ABSTRACTION\INTERFACE PROGRAMS>javac XeroMachine.java

C:\2nd sem\java prc\ABSTRACTION\INTERFACE PROGRAMS>java XeroMachine
Printing document
Scanning document
```

12d)calculator

Code:

```
interface Calculator {
   default void show() {
      System.out.println("Default Calculator");
   }
   int add(int a, int b);
}

class BasicCalc implements Calculator {
   public int add(int a, int b) {
      return a + b;
```

```
  }
}

public class Icalculator {
   public static void main(String[] args) {
      BasicCalc c = new BasicCalc();
      System.out.println("Sum: " + c.add(5, 3));
      c.show();
   }
}
```

Output:

```
C:\2nd sem\java prc\ABSTRACTION\INTERFACE PROGRAMS>javac Icalculator.java

C:\2nd sem\java prc\ABSTRACTION\INTERFACE PROGRAMS>java Icalculator
Sum: 8
Default Calculator
```

ABSTRACT CLASS PROGRAMS

13a) area of circle

Code:

```
abstract class Shape {
   abstract double area();

   void display() {
      System.out.println("This is a shape");
   }
}

class Circle extends Shape {
   double radius;
   Circle(double r) { radius = r; }

   double area() {
      return 3.14 * radius * radius;
   }
}

public class Ashape {
   public static void main(String[] args) {
      Shape s = new Circle(5);
      s.display();
      System.out.println("Area: " + s.area());
   }
}
```

Output:

```
C:\2nd sem\java prc\ABSTRACTION\ABSTRACT CLASS PROGRAMS>javac Ashape.java

C:\2nd sem\java prc\ABSTRACTION\ABSTRACT CLASS PROGRAMS>java Ashape
This is a shape
Area: 78.5
```

12b) abstraction in bank

Code:

```
abstract class BankAccount {
   protected double balance;
   public void deposit(double amount) {
      balance += amount;
      System.out.println("Deposited: " + amount);
   }


   public abstract void withdraw(double amount);
   public abstract double calculateInterest();


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

class SavingsAccount extends BankAccount {
   private static final double INTEREST_RATE = 0.05;


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


   public double calculateInterest() {
      double interest = balance * INTEREST_RATE;
      balance += interest;
      return interest;
   }
}
```

```
public class Bank {
   public static void main(String[] args) {
      BankAccount account = new SavingsAccount();
      account.deposit(1000);
      account.withdraw(200);
      System.out.println("Interest earned: " + account.calculateInterest());
      account.displayBalance();
   }
}
```

Output:

```
C:\2nd sem\java prc\ABSTRACTION\ABSTRACT CLASS PROGRAMS>java Bank
Deposited: 1000.0
Withdrawn: 200.0
Interest earned: 40.0
Current Balance: 840.0
```

12c) Abstraction in Employee

Code:

```
abstract class Employee {
   protected String name;
   protected int id;


   public Employee(String name, int id) {
      this.name = name;
      this.id = id;
   }


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


   public abstract double calculateSalary();
}

class FullTimeEmployee extends Employee {
   private double monthlySalary;

   public FullTimeEmployee(String name, int id, double monthlySalary) {
      super(name, id);
      this.monthlySalary = monthlySalary;
   }


   public double calculateSalary() {
```

```
    return monthlySalary;
  }
}

public class Work {
  public static void main(String[] args) {
    Employee emp = new FullTimeEmployee("John Doe", 101, 5000);
    emp.displayInfo();
    System.out.println("Monthly Salary: $" + emp.calculateSalary());
  }
}
```

Output:

```
C:\2nd sem\java prc\ABSTRACTION\ABSTRACT CLASS PROGRAMS>javac Work.java

C:\2nd sem\java prc\ABSTRACTION\ABSTRACT CLASS PROGRAMS>java Work
ID: 101, Name: John Doe
Monthly Salary: $5000.0
```

12d)

Code:

```
abstract class ElectronicDevice {
  private String brand;
  private double price;
  private int warrantyYears;


  public ElectronicDevice(String brand, double price, int warrantyYears) {
    this.brand = brand;
    this.price = price;
    this.warrantyYears = warrantyYears;
    System.out.println("ElectronicDevice constructor called");
  }


  public String getBrand() {
    return brand;
  }

  public double getPrice() {
    return price;
  }

  public void displayBasicInfo() {
    System.out.println("Brand: " + brand);
    System.out.println("Price: $" + price);
    System.out.println("Warranty: " + warrantyYears + " years");
```

```
    }


    public abstract void powerOn();
    public abstract void powerOff();
    public abstract void performDiagnostics();
}

class SmartTV extends ElectronicDevice {
    private int screenSize;
    private String resolution;


    public SmartTV(String brand, double price, int warranty,
                   int screenSize, String resolution) {
        super(brand, price, warranty);  // Calling parent constructor
        this.screenSize = screenSize;
        this.resolution = resolution;
        System.out.println("SmartTV constructor called");
    }


    public void powerOn() {
        System.out.println("SmartTV is booting up");
        System.out.println("Showing " + "brand" + " logo");
    }


    public void powerOff() {
        System.out.println("SmartTV is shutting down");
    }


    public void performDiagnostics() {
        System.out.println("Running TV diagnostics:");
        System.out.println("- Checking HDMI ports");
        System.out.println("- Testing display pixels");
    }


    public void displayTechSpecs() {
        displayBasicInfo();  // Using inherited method
        System.out.println("Screen Size: " + screenSize + " inches");
        System.out.println("Resolution: " + resolution);
    }
}

public class Device {
    public static void main(String[] args) {
        SmartTV myTV = new SmartTV("Samsung", 899.99, 2, 55, "4K UHD");

        System.out.println("\n=== Device Information ===");
        myTV.displayTechSpecs();
```

```
        System.out.println("\n=== Power Operations ===");
        myTV.powerOn();
        myTV.powerOff();

        System.out.println("\n=== Maintenance ===");
        myTV.performDiagnostics();
    }
}
```

Output:

```
C:\2nd sem\java prc\ABSTRACTION\ABSTRACT CLASS PROGRAMS>javac Device.java

C:\2nd sem\java prc\ABSTRACTION\ABSTRACT CLASS PROGRAMS>java Device
ElectronicDevice constructor called
SmartTV constructor called

=== Device Information ===
Brand: Samsung
Price: $899.99
Warranty: 2 years
Screen Size: 55 inches
Resolution: 4K UHD

=== Power Operations ===
SmartTV is booting up
Showing brand logo
SmartTV is shutting down

=== Maintenance ===
Running TV diagnostics:
- Checking HDMI ports
- Testing display pixels
```

ENCAPSULATION

14a) Bank

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", "Raaghav vel", 5000.0);
        myAccount.displayAccountInfo();
        myAccount.deposit(2000);
        myAccount.withdraw(1500);
        myAccount.withdraw(7000);
        myAccount.displayAccountInfo();
    }
}
```

Output:

```
C:\2nd sem\java prc\ENCAPSULATION>javac Bank.java

C:\2nd sem\java prc\ENCAPSULATION>java Bank
Account Number: 123456789
Owner Name: Raaghav vel
Current Balance: $5000.0
Deposited: $2000.0
Withdrawn: $1500.0
Insufficient balance or invalid amount!
Account Number: 123456789
Owner Name: Raaghav vel
Current Balance: $5500.0
```

14b) Employee

Code:

```
class Employee {
    private String empName;
    private int empID;

    public String getEmpName() {
    return empName;
    }

    public void setEmpName(String empName) {
    this.empName = empName;
    }

    public int getEmpID() {
    return empID;
```

```
    }

    public void setEmpID(int empID) {
    if (empID > 0) {
    this.empID = empID;
    } else {
    System.out.println("Invalid employee ID!");
    }
    }
    }

    public class Ework {
    public static void main(String[] args) {
    Employee e = new Employee();
    e.setEmpName("Nina");
    e.setEmpID(1022);
    System.out.println("Employee Name: " + e.getEmpName());
    System.out.println("Employee ID: " + e.getEmpID());
    }
    }
```

Output:

```
C:\2nd sem\java prc\ENCAPSULATION>javac Ework.java

C:\2nd sem\java prc\ENCAPSULATION>java Ework
Employee Name: Nina
Employee ID: 1022
```

14c) books

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", "Chu Pal", 499.99);
        b.displayBookInfo();
        b.setPrice(-100);
        b.setPrice(599.99);
        b.displayBookInfo();
    }
}
```

Output:

```
C:\2nd sem\java prc\ENCAPSULATION>javac BookInfo.java

C:\2nd sem\java prc\ENCAPSULATION>java BookInfo
Title: Fight Club
Author: Chu Pal
Price: $499.99
Price cannot be negative!
Title: Fight Club
Author: Chu Pal
Price: $599.99
```

14d) books

Code:

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

    public void setDetails(String n, int m) {
        name = n;
        marks = m;
    }

    public String getName() {
        return name;
    }

    public int getMarks() {
        return marks;
    }
}

public class EncapsulationStudent {
    public static void main(String[] args) {
        Student s = new Student();
        s.setDetails("RAAGHAV VAL", 88);

        System.out.println("Student Name: " + s.getName());
        System.out.println("Marks Scored: " + s.getMarks());
    }
}
```

Output:

```
C:\2nd sem\java prc\ENCAPSULATION>javac EncapsulationStudent.java

C:\2nd sem\java prc\ENCAPSULATION>java EncapsulationStudent
Student Name: RAAGHAV VAL
Marks Scored: 88
```

14e) Hospital

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("P121", "vasu", 18, "Fever");
        p.displayPatientInfo();
        p.setAge(-5);
        p.setDisease("Flu");
        p.displayPatientInfo();
    }
}
```

Output:

```
C:\2nd sem\java prc\ENCAPSULATION>javac Hospital.java

C:\2nd sem\java prc\ENCAPSULATION>java Hospital
Patient ID: P123
Name: Kailash
Age: 18
Disease: Fever
Age must be positive!
Patient ID: P123
Name: Kailash
Age: 18
Disease: Flu
```

14f) student info

Code:

```
public class Student {
   private final String id;
   private final String name;
   private final int age;

   public Student(String id, String name, int age) {
      this.id = id;
      this.name = name;
      this.age = age;
   }

   public String getId() {
      return id;
   }

   public String getName() {
      return name;
   }

   public int getAge() {
      return age;
   }
}

public class Main {
   public static void main(String[] args) {
      Student student = new Student("S1001", "Alice", 20);
```

```
        System.out.println("Student ID: " + student.getId());
        System.out.println("Name: " + student.getName());
        System.out.println("Age: " + student.getAge());

    }
}
```

Output:

```
C:\2nd sem\java prc\ENCAPSULATION>java Main
Student ID: S1001
Name: Alice
Age: 20
```

14g) Employee salary

Code:

```
class Employee {
    private String name;
    private int age;
    private double salary;
    private String department;

    public Employee(String name, int age, double salary, String department) {
        this.name = name;
        this.age = age;
        this.salary = salary;
        this.department = department;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public double getSalary() {
        return salary;
    }

    public String getDepartment() {
        return department;
    }


    public void setName(String name) {
```

```
    if (name != null && !name.isEmpty()) {
        this.name = name;
    } else {
        System.out.println("Invalid name provided");
    }
  }

  public void setAge(int age) {
    if (age >= 18 && age <= 65) {
        this.age = age;
    } else {
        System.out.println("Age must be between 18-65");
    }
  }

  public void setSalary(double salary) {
    if (salary >= 0) {
        this.salary = salary;
    } else {
        System.out.println("Salary cannot be negative");
    }
  }

  public void setDepartment(String department) {
    if (department != null && !department.isEmpty()) {
        this.department = department;
    } else {
        System.out.println("Department cannot be empty");
    }
  }


  public void giveRaise(double percentage) {
    if (percentage > 0 && percentage <= 20) {
        this.salary += this.salary * (percentage / 100);
        System.out.println("Raise applied. New salary: " + this.salary);
    } else {
        System.out.println("Raise percentage must be between 0-20%");
    }
  }
}

public class EmpSal {
  public static void main(String[] args) {
    // Create employee object
    Employee emp = new Employee("John Doe", 30, 50000.0, "Engineering");

    // Access data through getters
    System.out.println("Employee Name: " + emp.getName());
    System.out.println("Department: " + emp.getDepartment());
    System.out.println("Current Salary: $" + emp.getSalary());

    // Modify data through setters
    emp.setName("garr");
```

```
        emp.setAge(67);
        emp.setSalary(-1000);

        // Give raise through business method
        emp.giveRaise(10);
        emp.giveRaise(25);

        // Print updated information
        System.out.println("\nUpdated Information:");
        System.out.println("Name: " + emp.getName());
        System.out.println("Age: " + emp.getAge());
        System.out.println("Salary: $" + emp.getSalary());
    }
}
```

Output:

```
C:\2nd sem\java prc\ENCAPSULATION>javac EmpSal.java

C:\2nd sem\java prc\ENCAPSULATION>java EmpSal
Employee Name: John Doe
Department: Engineering
Current Salary: $50000.0
Age must be between 18-65
Salary cannot be negative
Raise applied. New salary: 55000.0
Raise percentage must be between 0-20%

Updated Information:
Name: garr
Age: 30
Salary: $55000.0
```

PACKAGES PROGRAMS

15.a) basic math package

Code:

```
public class BuiltInExample2 {
   public static void main(String[] args) {
      double num = 4.0;

      System.out.println("Square root: " + Math.sqrt(num));
      System.out.println("Power: " + Math.pow(num, 3));
      System.out.println("Random number: " + Math.random());
   }
}
```

Output:

```
C:\2nd sem\java prc\PACKAGES PROGRAMS>javac BuiltInExample.java

C:\2nd sem\java prc\PACKAGES PROGRAMS>java BuiltInExample
Square root: 2.0
Power: 64.0
Random number: 0.04868881362057287
```

15.b) Date Operations

Code:

```
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;

public class DateOperations {
   public static void main(String[] args) {
      LocalDate today = LocalDate.now();
      LocalDate birthday = LocalDate.of(1990, 5, 15);

      DateTimeFormatter formatter = DateTimeFormatter.ofPattern("MMMM dd, yyyy");

      System.out.println("Today: " + today.format(formatter));
      System.out.println("Birthday: " + birthday.format(formatter));

      long daysAlive = ChronoUnit.DAYS.between(birthday, today);
      System.out.println("Days alive: " + daysAlive);
   }
```

}

Output:

```
C:\2nd sem\java prc\PACKAGES PROGRAMS>javac DateOperations.java

C:\2nd sem\java prc\PACKAGES PROGRAMS>java DateOperations
Today: April 06, 2025
Birthday: May 15, 1990
Days alive: 12745
```

15.c) basic math package

Code:

```
package com.myutils;

public class MathOperations {
   public static int add(int a, int b) {
      return a + b;
   }

   public static double multiply(double x, double y) {
      return x * y;
   }
}


package com.myutils;

public class CalculatorApp {
   public static void main(String[] args) {
      System.out.println("5 + 3 = " + MathOperations.add(5, 3));
      System.out.println("2.5 * 4.0 = " + MathOperations.multiply(2.5, 4.0));
   }
}
```

Output:

```
javac -d . MathOperations.java CalculatorApp.java
java com.myutils.CalculatorApp
```

15.d) basic math package

Code:

```
package com.bank.model;

public class Account {
   private String accNumber;
   private double balance;

   public Account(String accNumber) {
      this.accNumber = accNumber;
      this.balance = 0.0;
   }

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

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


package com.bank;

import com.bank.model.Account;

public class Main {
   public static void main(String[] args) {
      Account acc1 = new Account("ACC1001");
      acc1.deposit(1000);
      acc1.displayBalance();
   }
}
```

Output:

```
Account ACC1001 balance: $1000.0
```

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:

```
C:\2nd sem\java prc\EXCEPTION HANDLING PROGRAMS>javac BankTransaction.java

C:\2nd sem\java prc\EXCEPTION HANDLING PROGRAMS>java BankTransaction
Enter amount to withdraw: 1000
Withdrawal successful! Remaining balance: 4000.0
Thank you for using our banking service.

C:\2nd sem\java prc\EXCEPTION HANDLING PROGRAMS>java BankTransaction
Enter amount to withdraw: 50000
Transaction Failed: Insufficient balance! Available: 5000.0
Thank you for using our banking service.
```

16.b) Vowel Check

Code:

```
public class Vowel_Check {
  public static void main(String[] args) {
    try {
      String text = "Java handling and managing exceptions ";
      System.out.println("Original string: " + text);
      checkVowels(text);
      System.out.println("String contains vowels.");
    } catch (NoVowelsException e) {
      System.out.println("Error: " + e.getMessage());
    }
  }

  public static void checkVowels(String text) throws NoVowelsException {
    boolean containsVowels = false;
    String vowels = "aeiouAEIOU";

    for (int i = 0; i < text.length(); i++) {
      char ch = text.charAt(i);
      if (vowels.contains(String.valueOf(ch))) {
        containsVowels = true;
        break;
      }
    }
    if (!containsVowels) {
      throw new NoVowelsException("String does not contain any vowels.");
    }
  }
}
class NoVowelsException extends Exception {
  public NoVowelsException(String message) {
    super(message);
  }
}
```

Output:

```
C:\2nd sem\java prc\EXCEPTION HANDLING PROGRAMS>javac Vowel_Check.java

C:\2nd sem\java prc\EXCEPTION HANDLING PROGRAMS>java Vowel_Check
Original string: Java handling and managing exceptions
String contains vowels.
```

16.c) Exception OddNumber

Code:

```
public class Exception_OddNumber {
  public static void main(String[] args) {
    int n = 18;
    trynumber(n);
    n = 7;
    trynumber(n);
  }

  public static void trynumber(int n) {
    try {
      checkEvenNumber(n);
      System.out.println(n + " is even.");
    } catch (IllegalArgumentException e) {
      System.out.println("Error: " + e.getMessage());
    }
  }

  public static void checkEvenNumber(int number) {
    if (number % 2 != 0) {
      throw new IllegalArgumentException(number + " is odd.");
    }
  }
}
```

Output:

```
C:\2nd sem\java prc\EXCEPTION HANDLING PROGRAMS>javac Exception_OddNumber.java

C:\2nd sem\java prc\EXCEPTION HANDLING PROGRAMS>java Exception_OddNumber
18 is even.
Error: 7 is odd.
```

16.d) Onlynumber

Code:

```
public class Onlynumber {
   public static void main(String[] args) {
      try {
         String s = "abc";
         int num = Integer.parseInt(s);
         System.out.println("Number: " + num);
      } catch (NumberFormatException e) {
         System.out.println("Invalid number format!");
      }
   }
}
```

Output:

```
C:\2nd sem\java prc\EXCEPTION HANDLING PROGRAMS>javac Onlynumber.java

C:\2nd sem\java prc\EXCEPTION HANDLING PROGRAMS>java Onlynumber
Invalid number format!
```

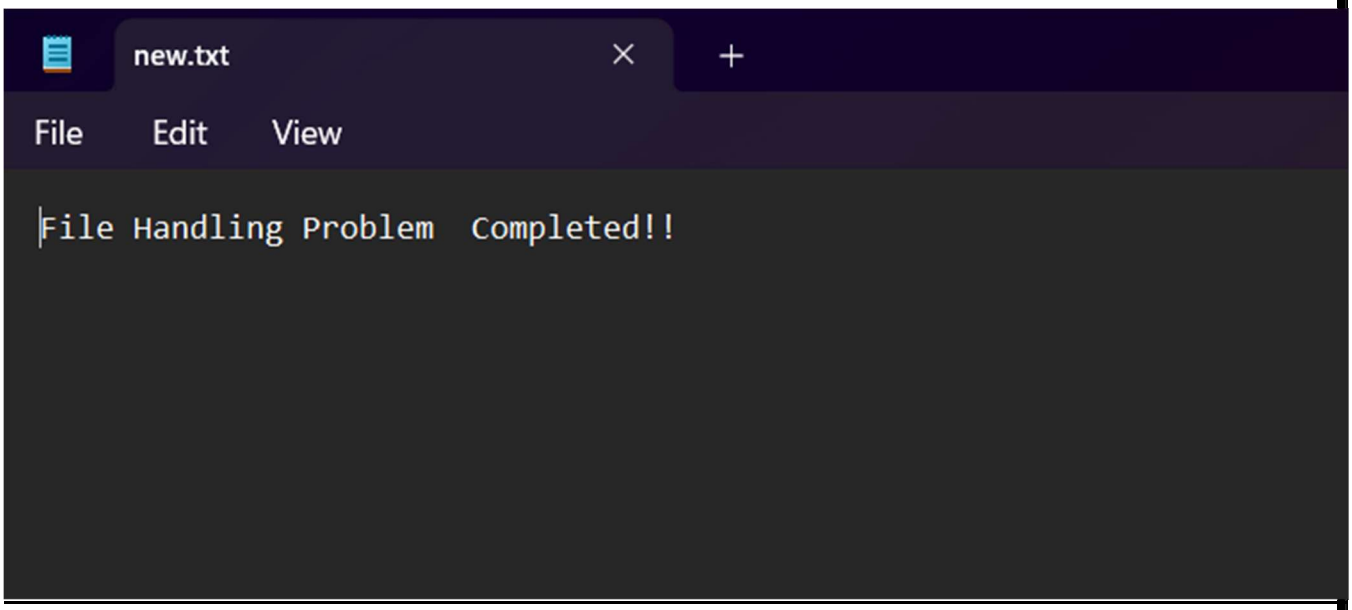
FILE HANDLING PROGRAMS

17.a)file handing ex1

Code:

```
import java.io.FileWriter;
public class f1{
        public static void main(String[]args){
          try{
            FileWriter f1=new FileWriter("new.txt");
            f1.write("File Handling Problem");
            f1.append("  Completed!!");
            f1.close();}
        catch(Exception error){
     System.out.println("Error is: "+error);
    }
}}
```

Output:



17.b) file handing ex2

Code:

```
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
public class f2{
```

```
public static void main(String[]args){
try{
FileReader f1=new FileReader("Mera.txt");
BufferedReader b1=new BufferedReader(f1);
String c=b1.readLine();
while (c!=null){
System.out.println(c);
c=b1.readLine();
}
b1.close();
}
catch(Exception e){
System.out.println("Error found");
}
}
}
```

Output:

```
C:\2nd sem\java prc\FILE HANDLING PROGRAMS>javac f2.java

C:\2nd sem\java prc\FILE HANDLING PROGRAMS>java f2
Error found
```

17.c) file handing ex3

Code:

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

class f3 {
  public static void main(String[] args) {
    try {
      File myFile = new File("example.txt");
      Scanner reader = new Scanner(myFile);

      while (reader.hasNextLine()) {
        String data = reader.nextLine();
        System.out.println(data);
      }

      reader.close();
    }
    catch (Exception e) {
      System.out.println("An error occurred.");
    }
  }
}
```

Output:

```
C:\2nd sem\java prc\FILE HANDLING PROGRAMS>java f3
An error occurred.

C:\2nd sem\java prc\FILE HANDLING PROGRAMS>java f3
An error occurred.
```

17.d) file handing ex 4

Code:

```
import java.io.*;

public class f4 {
   public static void main(String[] args) {

      String fileName = "example.txt";

      try (FileWriter writer = new FileWriter(fileName)) {
         writer.write("Hello, this is a test file!\n");
         writer.write("This is line 2.");
         System.out.println("Successfully wrote to the file.");
      } catch (IOException e) {
         System.out.println("Error writing to file: " + e.getMessage());
      }


      try (BufferedReader reader = new BufferedReader(new FileReader(fileName))) {
         System.out.println("\nFile contents:");
         String line;
         while ((line = reader.readLine()) != null) {
            System.out.println(line);
         }
      } catch (IOException e) {
         System.out.println("Error reading file: " + e.getMessage());
      }


      try (FileWriter writer = new FileWriter(fileName, true)) {
         writer.write("\nThis is an appended line.");
         System.out.println("\nSuccessfully appended to the file.");
      } catch (IOException e) {
         System.out.println("Error appending to file: " + e.getMessage());
      }


      try (FileInputStream fis = new FileInputStream(fileName)) {
         System.out.println("\nReading using FileInputStream:");
```

```
int content;
          while ((content = fis.read()) != -1) {
             System.out.print((char) content);
          }
       } catch (IOException e) {
          System.out.println("Error reading file: " + e.getMessage());
       }
    }
}
```

Output: