



SCHOOL OF
COMPUTING

LAB RECORD

23CSE111- Object Oriented Programming

Submitted by

CH.SC.U4CSE24038 -S KUSUMA REDDY

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND
ENGINEERING

AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF COMPUTING

CHENNAI

March - 2025



**SCHOOL OF
COMPUTING**

**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.U4CSE24038 – S KUSUMA REDDY** 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

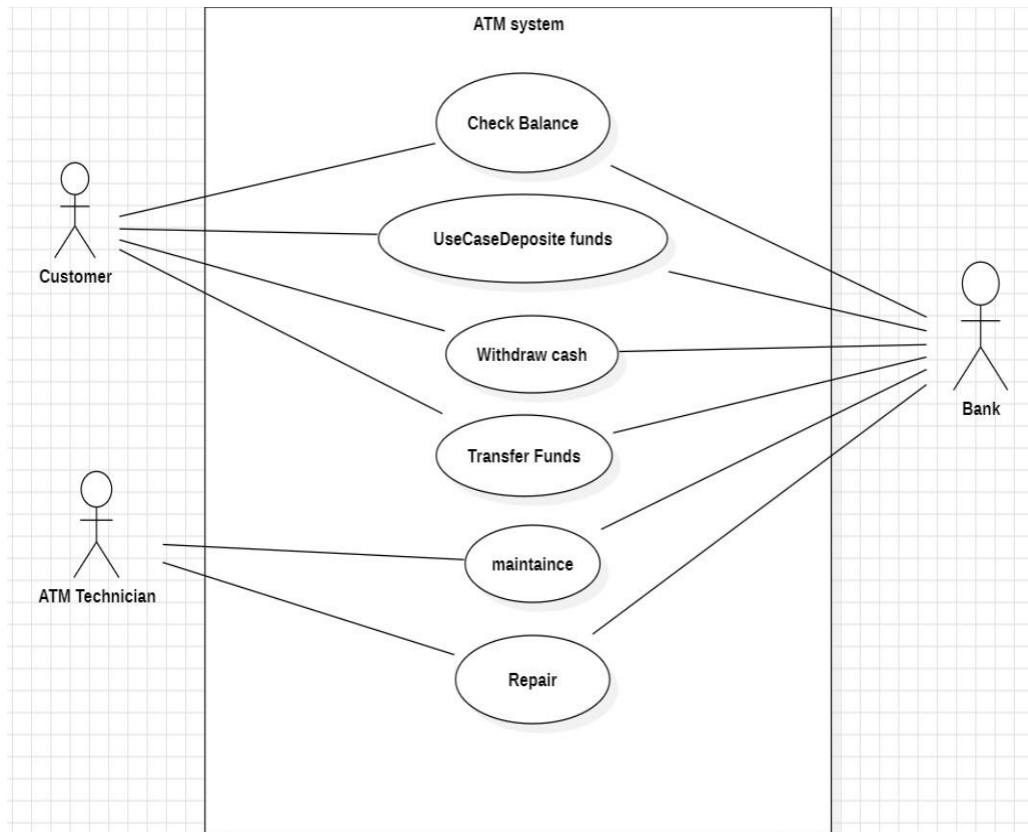
S.NO	TITLE	PAGE.NO
	UML DIAGRAM	
1.	ATM SYSTEM	
	1.a)Use Case Diagram	
	1.b)Class Diagram	
	1.c) Sequence Diagram	
	1.d) ACTIVITY DIAGRAM	
	1.e) STATE DIAGRAM	
2.	ONLINE ATTENDENCE	
	2.a) Use Case Diagram	
	2.b) Class Diagram	
	2.c) Sequence Diagram	
	2.d) STATE DIAGRAM	
	2.e) ACTIVITY DIAGRAM	
3.	BASIC JAVA PROGRAMS	
	3.a)calculator	
	3.b)diamond pattern	
	3.c)spiral matrix	
	3.d) Prime check	
	3.e) Magic Square	
	3.f) Prime Palindromes	
	3.g) Kaprekar's Constant	
	3.h) Generate a Hilbert Curve	
	3.i) Sudoku Puzzle	
	3.j) Number is Happy	
	INHERITANCE	
4.	SINGLE INHERITANCE PROGRAMS	
	4.a)area and volume	
	4.b) getAverageGradeForStudent	

5.	MULTILEVEL INHERITANCE PROGRAMS	
	5.a)employee management	
	5.b)grand prarent multi	
6.	HIERARCHICAL INHERITANCE PROGRAMS	
	6.a) employee	
	6.b) heirarcalth	
7.	HYBRID INHERITANCE PROGRAMS	
	7.a) hybrid bank	
	7.b) hybrid employee	
	POLYMORPHISM	
8.	CONSTRUCTOR PROGRAMS	
	8.a)copy constuctor	
9.	CONSTRUCTOR OVERLOADING PROGRAMS	
	9.a)constructor overloding	
10.	METHOD OVERLOADING PROGRAMS	
	10.a)hillstations	
	10.b) Test polymorphism	
11.	METHOD OVERRIDING PROGRAMS	
	11.a)banking	
	11.b)payslip	
	ABSTRACTION	
12.	INTERFACE PROGRAMS	
	12.a) Arithmetic Operations	
	12.b) Shape Calculation	
	12.c) Multiple Interface	
	12.d) Bank Transactions	
13.	ABSTRACT CLASS PROGRAMS	
	13.a) Animal Sounds	
	13.b)Employee Salary Calculation	
	13.c) Vehicle Features	
	13.d) Student Result Calculation	
	ENCAPSULATION	
14.	ENCAPSULATION PROGRAMS	
	14.a) Bank Account	
	14.b)Student data	
	14.c) Employee Details	
	14.d)Cra model	
15.	PACKAGES PROGRAMS	
	15.a)User Defined Packages	
	15.b)User Defined Packages	
	15.c)Built – in Package(3 Packages)	
	15.d)Built – in Package(3 Packages)	

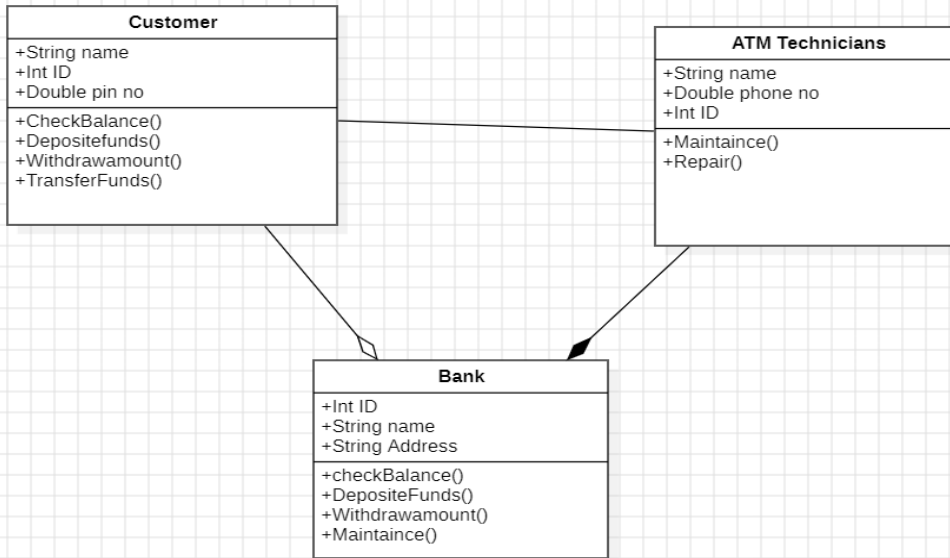
16.	EXCEPTION HANDLING PROGRAMS	
	16.a) Advanced Exception	
	16.b) Throwing Custom Exceptions	
	16.c) Exception Propagation	
	16.d) Rethrowing Exception	
17.	FILE HANDLING PROGRAMS	
	17.a) Copying Content	
	17.b) Counting Words	
	17.c) Appending Data	
	17.d) BufferedReader	

1.ATM SYSTEM

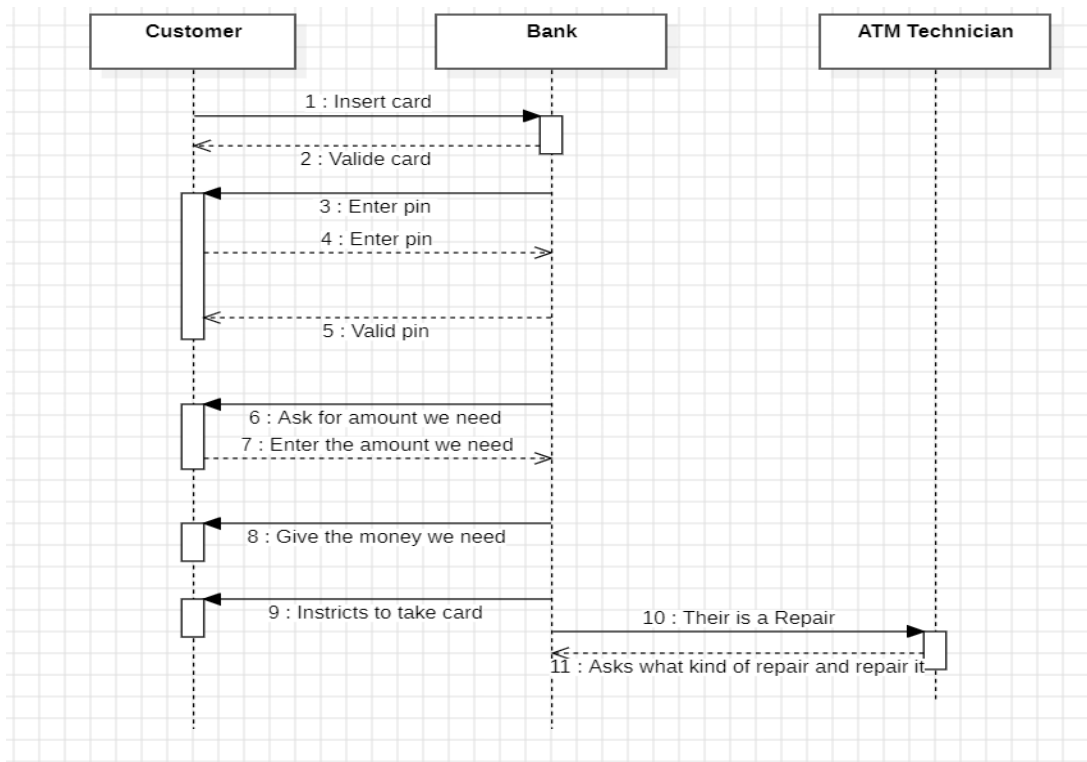
1(a): USE CASE DIAGRAM



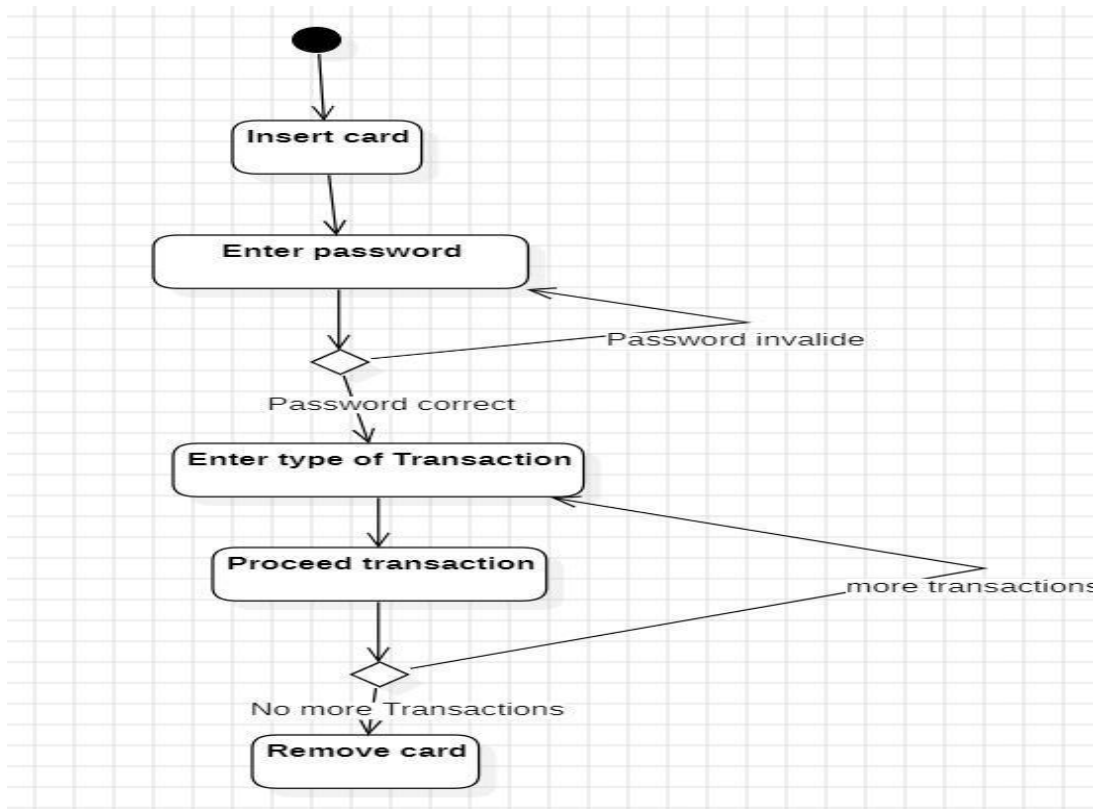
1(b): CLASS DIAGRAM



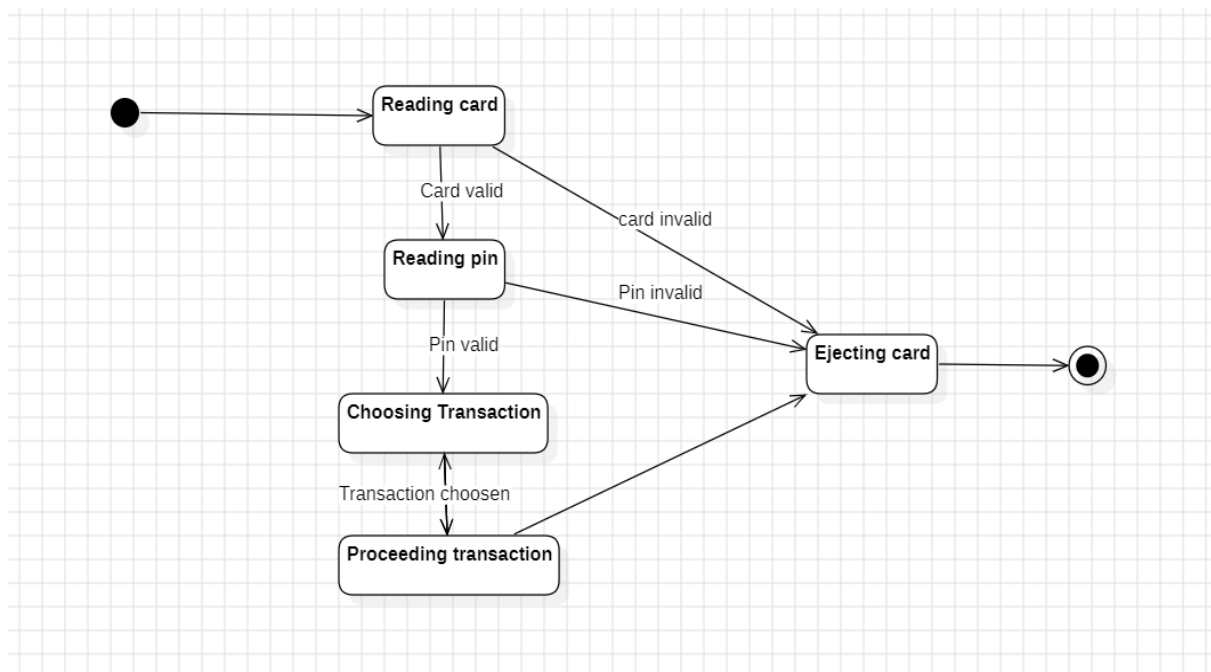
1(c): SEQUENCE DIAGRAM



1(e): ACTIVITY DIAGRAM

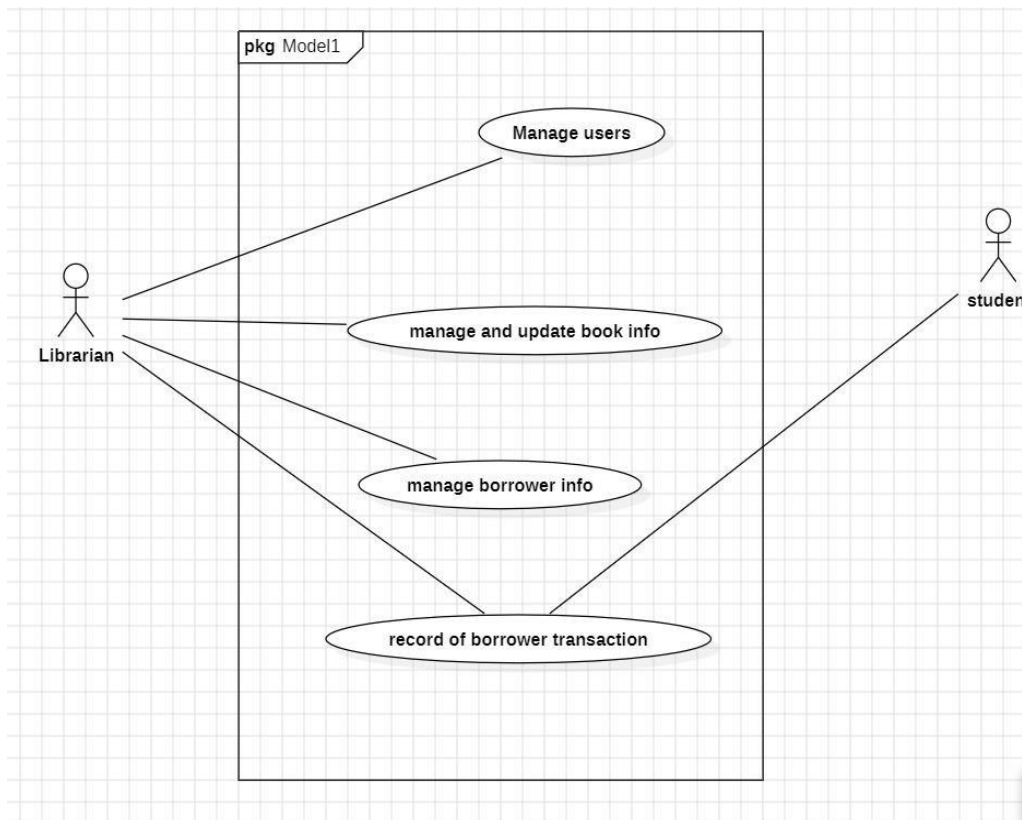


1(d): STATE DIAGRAM

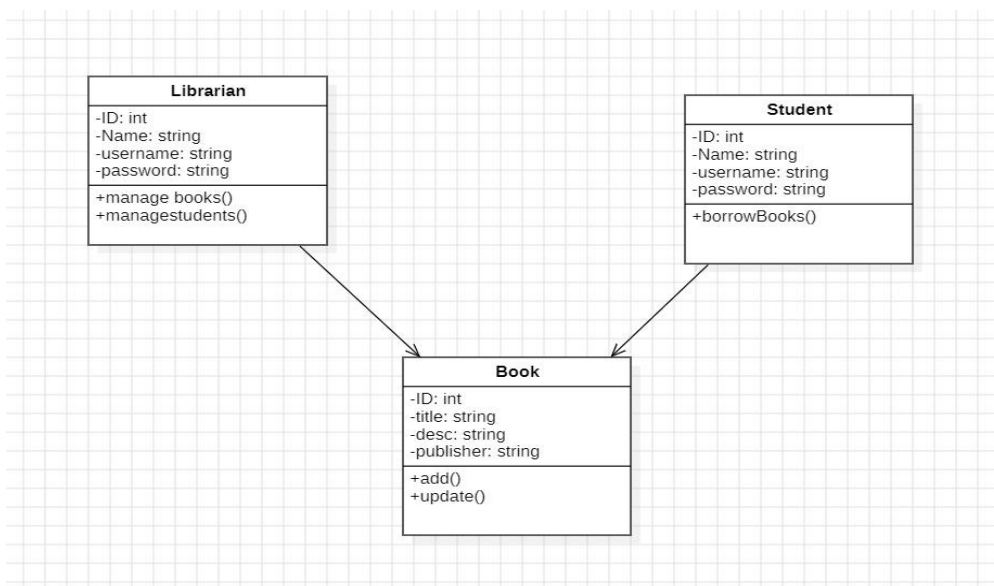


2. ONLINE ATTENDENCE

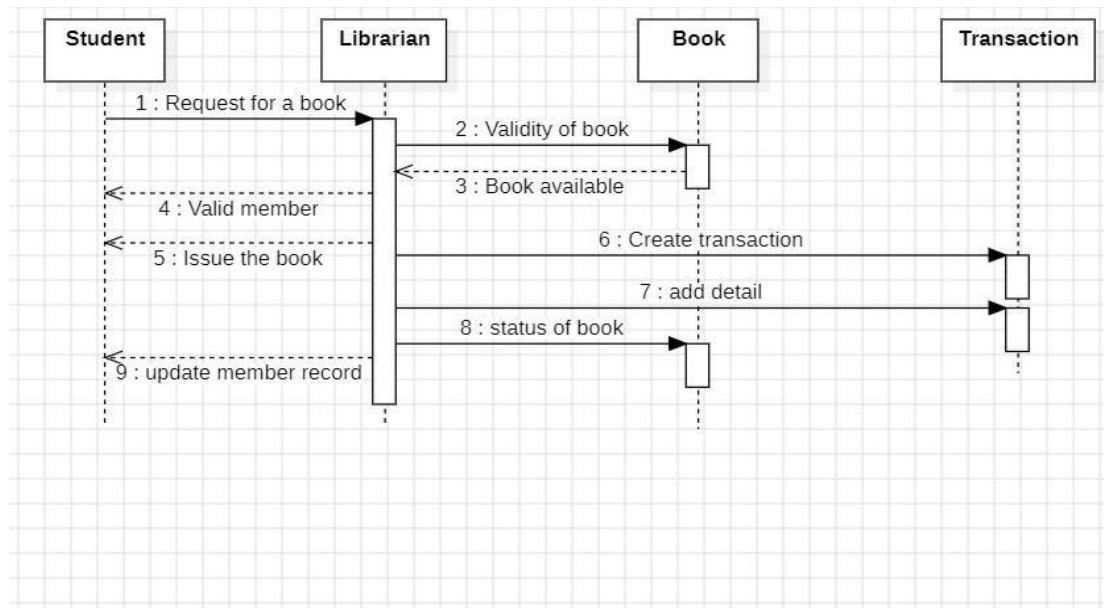
2.A): USE CASE DIAGRAM



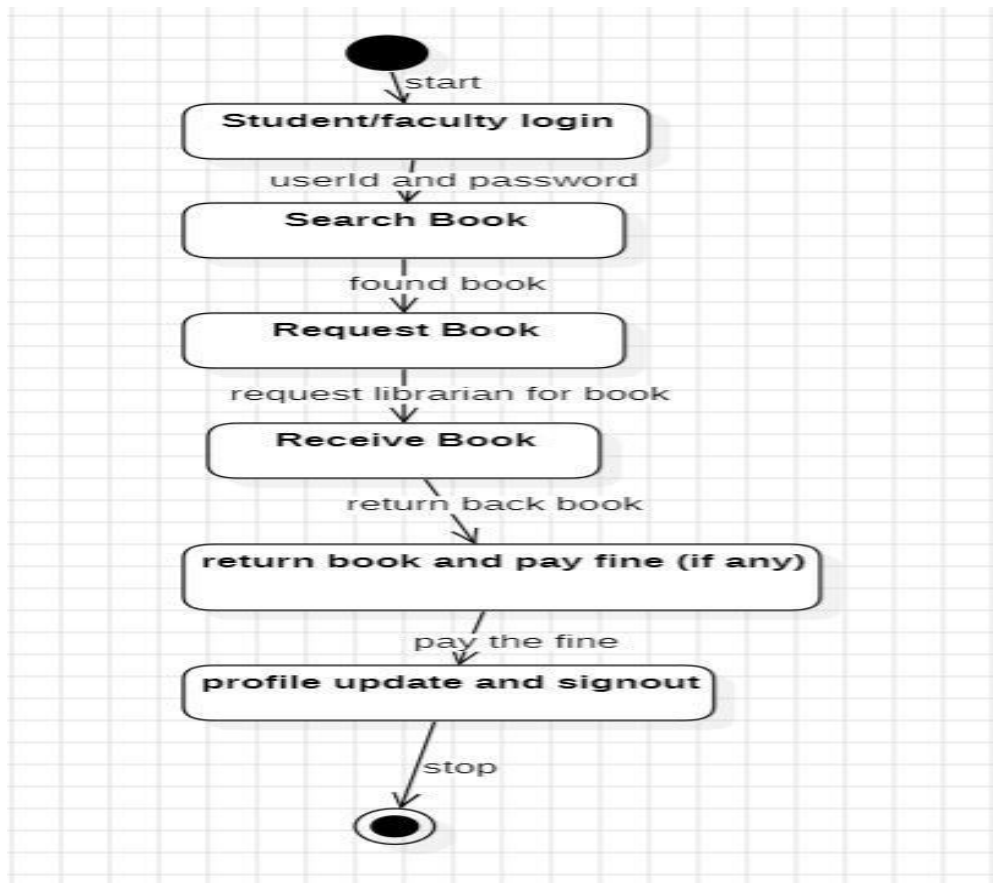
2(B): CLASS DIAGRAM



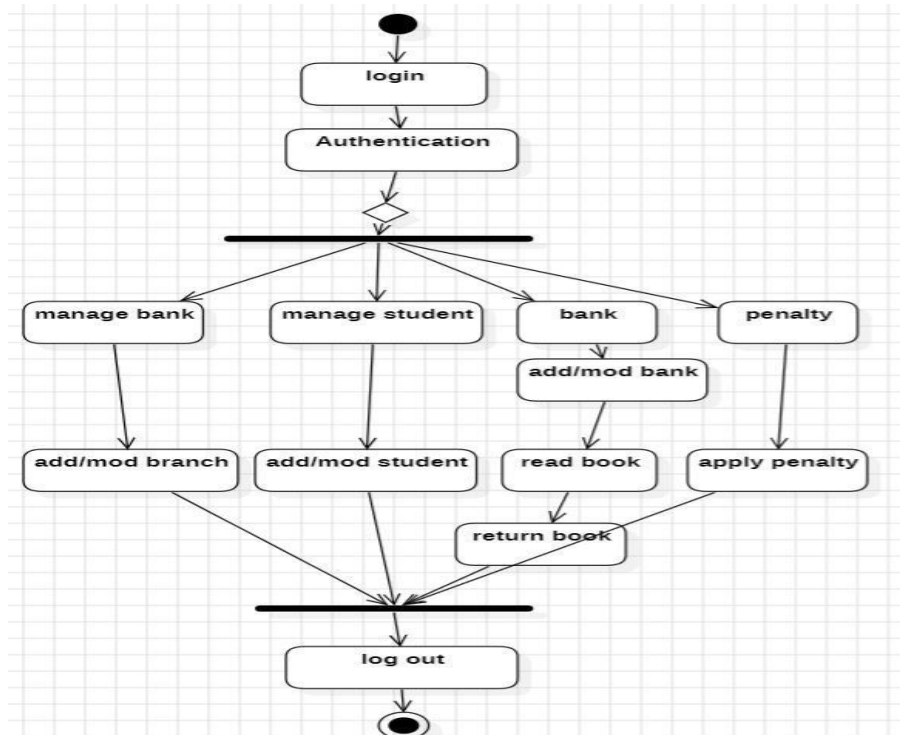
2(C): SEQUENCE DIAGRAM



2(D): STATE DIAGRAM



2(E): ACTIVITY DIAGRAM



3. BASIC JAVA PROGRAMS

3.a)calculator

Code:

```
import java.util.Scanner;
```

```
public class Calculator {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter first number: ");
        double num1 = sc.nextDouble();
        System.out.print("Enter second number: ");
        double num2 = sc.nextDouble();
        System.out.print("Enter an operator (+, -, *, /): ");
        char operator = sc.next().charAt(0);
        double result;

        switch (operator) {
            case '+':
                result = num1 + num2;
                break;
```

```
case '-':
    result = num1 - num2;
    break;
case '*':
    result = num1 * num2;
    break;
case '/':
    if (num2 != 0)
        result = num1 / num2;
    else {
        System.out.println("Cannot divide by zero!");
        return;
    }
    break;
default:
    System.out.println("Invalid operator!");
    return;
}

System.out.println("Result: " + result);
sc.close();
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac Calculator.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java Calculator
Enter first number: 5
Enter second number: 9
Enter an operator (+, -, *, /): /
Result: 0.5555555555555556

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

3.b)diamond pattern

Code:

```
import java.util.Scanner;

public class DiamondPattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows: ");
        int n = sc.nextInt();

        for (int i = 1; i <= n; i += 2) {
            for (int j = 0; j < (n - i) / 2; j++) System.out.print(" ");
            for (int j = 0; j < i; j++) System.out.print("*");
            System.out.println();
        }
    }
}
```

```
}

for (int i = n - 2; i >= 1; i -= 2) {
    for (int j = 0; j < (n - i) / 2; j++) System.out.print(" ");
    for (int j = 0; j < i; j++) System.out.print("*");
    System.out.println();
}

sc.close();
}
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac DiamondPattern.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java DiamondPattern
Enter the number of rows: 5
  *
 ***
*****
 ***
  *

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

3.c)spiral matrix

Code:

```
import java.util.Scanner;

public class SpiralMatrix {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the matrix (N): ");
        int n = sc.nextInt();
        int[][] matrix = new int[n][n];

        int value = 1, top = 0, bottom = n - 1, left = 0, right = n - 1;

        while (value <= n * n) {
            for (int i = left; i <= right; i++) matrix[top][i] = value++;
            top++;

            for (int i = top; i <= bottom; i++) matrix[i][right] = value++;
            right--;

            for (int i = right; i >= left; i--) matrix[bottom][i] = value++;
            bottom--;

            for (int i = bottom; i >= top; i--) matrix[i][left] = value++;
            left++;
        }
    }
}
```

```
        left++;  
    }  
  
    for (int[] row : matrix) {  
        for (int num : row) System.out.printf("%4d", num);  
        System.out.println();  
    }  
  
    sc.close();  
}  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac SpiralMatrix.java  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java SpiralMatrix  
Enter the size of the matrix (N): 4  
  1   2   3   4  
12 13 14  5  
11 16 15  6  
10  9  8  7  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

3.d)Prime check

Code:

```
import java.util.Scanner;  
  
public class PrimeCheck {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Enter number: ");  
        int num = sc.nextInt();  
        boolean isPrime = true;  
  
        if (num <= 1) isPrime = false;  
  
        for (int i = 2; i <= Math.sqrt(num); 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");  
    }  
}
```

```
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac PrimeCheck.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java PrimeCheck
Enter number: 5
5 is a Prime number

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

3.e) Magic Square

Code:

```
import java.util.Scanner;
```

```
public class MagicSquare {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter odd size of square: ");
        int n = sc.nextInt();

        if (n % 2 == 0) {
            System.out.println("Only odd order supported.");
            return;
        }

        int[][] magic = new int[n][n];
        int number = 1;
        int i = 0, j = n / 2;

        while (number <= n * n) {
            magic[i][j] = number++;
            i--;
            j++;
            if (number % n == 1) {
                i += 2;
                j--;
            } else if (j == n) {
                j -= n;
            } else if (i < 0) {
                i += n;
            }
        }

        System.out.println("Magic Square:");
        for (int[] row : magic) {
            for (int val : row)
                System.out.printf("%3d ", val);
            System.out.println();
        }
    }
}
```


Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac MagicSquare.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java MagicSquare
Enter odd size of square:
5
Magic Square:
 17  24   1   8  15
 23   5   7  14  16
  4   6  13  20  22
 10  12  19  21   3
 11  18  25   2   9

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

3.f) Prime Palindromes

Code:

```
public class PrimePalindrome {
    public static boolean isPrime(int n) {
        if (n < 2) return false;
        for (int i = 2; i <= Math.sqrt(n); i++)
            if (n % i == 0) return false;
        return true;
    }

    public static boolean isPalindrome(int n) {
        int rev = 0, original = n;
        while (n != 0) {
            rev = rev * 10 + n % 10;
            n /= 10;
        }
        return original == rev;
    }

    public static void main(String[] args) {
        System.out.println("Prime Palindromes between 1 and 1000:");
        for (int i = 1; i <= 1000; i++) {
            if (isPrime(i) && isPalindrome(i)) {
                System.out.print(i + " ");
            }
        }
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac PrimePalindrome.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java PrimePalindrome
Prime Palindromes between 1 and 1000:
2 3 5 7 11 101 131 151 181 191 313 353 373 383 727 757 787 797 919 929
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

3.g) Kaprekar's Constant

Code:

```
import java.util.Arrays;
import java.util.Scanner;

public class KaprekarConstant {
    static int kaprekarRoutine(int num) {
        int count = 0;
        while (num != 6174) {
            char[] digits = String.format("%04d", num).toCharArray();
            Arrays.sort(digits);
            int asc = Integer.parseInt(new String(digits));
            int desc = Integer.parseInt(new StringBuilder(new String(digits)).reverse().toString());
            num = desc - asc;
            count++;
            System.out.println(desc + " - " + asc + " = " + num);
            if (num == 0) return -1; // Stops if all digits are the same
        }
        return count;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a 4-digit number (non-repeating digits): ");
        int num = sc.nextInt();
        int steps = kaprekarRoutine(num);
        if (steps == -1) System.out.println("Invalid input! Try a number with different digits.");
        else System.out.println("Reached 6174 in " + steps + " steps.");
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java KaprekarConstant
Enter a 4-digit number (non-repeating digits): 9876
9876 - 6789 = 3087
8730 - 378 = 8352
8532 - 2358 = 6174
Reached 6174 in 3 steps.

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

3.h) Generate a Hilbert Curve

Code:

```
public class HilbertCurve {
    static void drawHilbert(int n, int angle) {
        if (n <= 0) return;
        System.out.print("L");
        drawHilbert(n - 1, -angle);
        System.out.print("F");
        System.out.print("R");
        drawHilbert(n - 1, angle);
        System.out.print("F");
        drawHilbert(n - 1, angle);
        System.out.print("R");
        System.out.print("F");
        drawHilbert(n - 1, -angle);
        System.out.print("L");
    }

    public static void main(String[] args) {
        int order = 3; // Order of Hilbert Curve
        System.out.println("Hilbert Curve of order " + order + ":" );
        drawHilbert(order, 90);
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac HilbertCurve.java  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java HilbertCurve  
Hilbert Curve of order 3:  
LLLFRRFRFLRFLFRFRFLFLFRFRFLRFLFRFRFLLRLLFRFRFLRFLFRFRFLFLFRFRFLRFLFRFRFLFLFRFRFLRFLFRFRLLRFLLRFRFRFL  
RLFRFRFLFLFRFRFLRFLFRFRFLLL  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

3.i) Sudoku Puzzle

Code:

```
import java.util.Random;

public class SudokuGenerator {
    static final int SIZE = 9;
    static final int[][] board = new int[SIZE][SIZE];

    public static void main(String[] args) {
        fillDiagonalBoxes();
        fillRemaining(0, 3);
        printBoard();
    }

    static void fillDiagonalBoxes() {
        for (int i = 0; i < SIZE; i += 3) fillBox(i, i);
    }

    static void fillBox(int row, int col) {
        Random rand = new Random();
        boolean[] used = new boolean[SIZE + 1];
        for (int i = 0; i < 3; i++)
```

```
        for (int j = 0; j < 3; j++) {
            int num;
            do num = rand.nextInt(SIZE) + 1;
            while (used[num]);
            used[num] = true;
            board[row + i][col + j] = num;
        }
    }

    static boolean fillRemaining(int i, int j) {
        if (j >= SIZE && i < SIZE - 1) {
            i += 1;
            j = 0;
        }
        if (i >= SIZE && j >= SIZE) return true;

        if (board[i][j] != 0) return fillRemaining(i, j + 1);

        for (int num = 1; num <= SIZE; num++) {
            if (isSafe(i, j, num)) {
                board[i][j] = num;
                if (fillRemaining(i, j + 1)) return true;
                board[i][j] = 0;
            }
        }
        return false;
    }

    static boolean isSafe(int row, int col, int num) {
        for (int x = 0; x < SIZE; x++)
            if (board[row][x] == num || board[x][col] == num) return false;
        return true;
    }

    static void printBoard() {
        for (int i = 0; i < SIZE; i++) {
            for (int j = 0; j < SIZE; j++) System.out.print(board[i][j] + " ");
            System.out.println();
        }
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac SudokuGenerator.java
```

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java SudokuGenerator
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 9 out of bounds for length 9
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:37)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:37)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:37)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:37)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:42)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:42)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:42)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:42)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:42)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:42)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:37)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:37)
    at SudokuGenerator.fillRemaining(SudokuGenerator.java:37)
```

3.j) Number is Happy

Code:

```
import java.util.Scanner;

public class HappyNumber {
    static int squareSum(int n) {
        int sum = 0;
        while (n != 0) {
            int digit = n % 10;
            sum += digit * digit;
            n /= 10;
        }
        return sum;
    }

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

        int slow = num, fast = num;

        do {
            slow = squareSum(slow);
            fast = squareSum(squareSum(fast));
        } while (slow != fast);

        if (slow == 1)
            System.out.println(num + " is a Happy number");
        else
            System.out.println(num + " is NOT a Happy number");
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac HappyNumber.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java HappyNumber
Enter number: 5
5 is NOT a Happy number

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

INHERITANCE

4.SINGLE INHERITANCE PROGRAMS

4.a)area and volume

Code:

```
class Area

{ public void a(int a, int b) {

System.out.println("Area: " + (a * b));}}

class Volume extends Area

{ public void v(int a, int b, int c)

{ System.out.println("Volume: " + (a * b * c)); }}

public class Main {

public static void main(String[] args) {

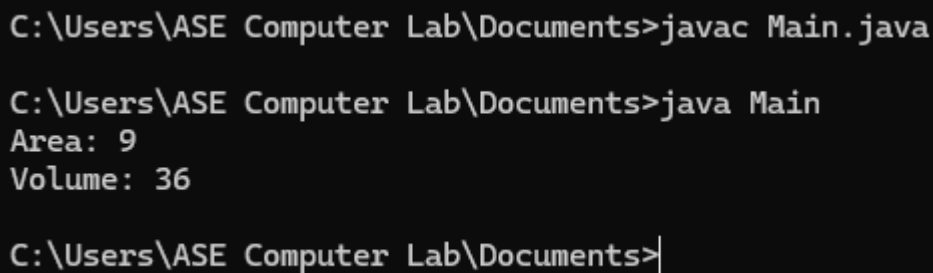
Volume vol = new Volume();

vol.a(3, 3);

vol.v(3, 3, 4);

} }
```

Output:



```
C:\Users\ASE Computer Lab\Documents>javac Main.java

C:\Users\ASE Computer Lab\Documents>java Main
Area: 9
Volume: 36

C:\Users\ASE Computer Lab\Documents>|
```

4.b) getAverageGradeForStudent

Code:

```
import java.util.ArrayList;

public class Main {
```

```
class Student {  
  
    private String studentName;  
    private ArrayList<Double> grades;  
  
    public Student(String studentName) {  
        this.studentName = studentName;  
        this.grades = new ArrayList<>();  
    }  
    public void addGrade(double grade) {  
        grades.add(grade);  
    }  
    public double calculateAverageGrade() {  
        if (grades.isEmpty()) {  
            return 0.0;  
        }  
        double sum = 0;  
        for (double grade : grades) {  
            sum += grade;  
        }  
        return sum / grades.size();  
    }  
    public String getStudentName() {  
        return studentName;  
    }  
}  
  
class Classroom {
```

```
private ArrayList<Student> students;

public Classroom() {
    students = new ArrayList<>();
}

public void addStudent(Student student) {
    students.add(student);
}

public double getAverageGradeForStudent(String studentName) {
    for (Student student : students) {
        if (student.getStudentName().equals(studentName)) {
            return student.calculateAverageGrade();
        }
    }
    return 0.0;
}

public static void main(String[] args) {
    Main mainInstance = new Main(); // Create an instance of the outer class
    Student student1 = mainInstance.new Student("Alice");
    Student student2 = mainInstance.new Student("Bob");
    student1.addGrade(85);
    student1.addGrade(90);
    student2.addGrade(75);
    student2.addGrade(80);
    Classroom classroom = mainInstance.new Classroom();
}
```



```
classroom.addStudent(student1);  
classroom.addStudent(student2);  
System.out.println("Average grade for Alice: " +  
classroom.getAverageGradeForStudent("Alice"));  
System.out.println("Average grade for Bob: " +  
classroom.getAverageGradeForStudent("Bob"));  
}  
}} Output:
```

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac Main.java  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java Main  
Average grade for Alice: 87.5  
Average grade for Bob: 77.5  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

MULTILEVEL INHERITANCE PROGRAMS

5.a)employee management

Code:

```
class Person {  
String name;  
int age;  
Person(String name, int age) {  
this.name = name;  
this.age = age;  
}  
void displayPO {  
System.out.println("Name of the employee is: " + name);
```

```
System.out.println("Age of the employee is: " + age);  
}  
}
```

```
class Employee extends Person {  
    int employeeID;  
    int salary;  
    Employee(int employeeID, int salary, String name, int age) {  
        super(name, age);  
        this.employeeID = employeeID;  
        this.salary = salary;  
    }  
    void displayP() {  
        super.displayP();  
        System.out.println("Employee ID of the employee is: " + employeeID);  
        System.out.println("Salary of the employee is: " + salary);  
    }  
}
```

```
class Manager extends Employee {  
    String department;  
    Manager(String department, int employeeID, int salary, String name, int age) {  
        super(employeeID, salary, name, age);  
        this.department = department;  
    }  
    void displayM() {  
        super.displayP();  
        System.out.println("Department of the employee is: " + department);  
    }  
}
```

```
}

public class Management {

public static void main(String[] args) {

Manager manager = new Manager("IT", 101, 85000, "Alice Johnson", 35);

manager.displayM();}}
```

output:

```
C:\Users\ASE Computer Lab\Desktop>javac Management.java

C:\Users\ASE Computer Lab\Desktop>java Management
Name of the employee is: Alice Johnson
Age of the employee is: 35
Employee ID of the employee is: 101
Salary of the employee is: 85000
Department of the employee is: IT

C:\Users\ASE Computer Lab\Desktop>|
```

5.b)grand prarent multi

Code:

```
// Base class (Grandparent)

class Vehicle {

    void start() {

        System.out.println("Vehicle is starting...");

    }

}

// Derived class (Parent)

class Car extends Vehicle {

    void drive() {
```

```
        System.out.println("Car is being driven...");
    }
}

// Derived class (Child)
class ElectricCar extends Car {
    void charge() {
        System.out.println("Electric Car is charging...");
    }
}

// Main class
public class MultilevelInheritanceExample {
    public static void main(String[] args) {
        ElectricCar tesla = new ElectricCar();

        // Calling methods from different levels of inheritance
        tesla.start(); // Inherited from Vehicle
        tesla.drive(); // Inherited from Car
        tesla.charge(); // Defined in ElectricCar
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac MultilevelInheritanceExample.java
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java MultilevelInheritanceExample
Vehicle is starting...
Car is being driven...
Electric Car is charging...
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

HIERARCHICAL INHERITANCE PROGRAMS

6.a) employee

Code:

```
class Employee {  
    String empName, empId, address, mailId;  
    long mobileNo;  
  
    public Employee(String empName, String empId, String address, String mailId, long mobileNo) {  
        this.empName = empName;  
        this.empId = empId;  
        this.address = address;  
        this.mailId = mailId;  
        this.mobileNo = mobileNo;  
    }  
  
    public void displayEmployeeDetails() {  
        System.out.println("Employee ID: " + empId);  
        System.out.println("Name: " + empName);  
        System.out.println("Address: " + address);  
        System.out.println("Mail ID: " + mailId);  
        System.out.println("Mobile No: " + mobileNo);  
    }  
}  
  
class Programmer extends Employee {  
    double basicPay;  
  
    public Programmer(String empName, String empId, String address, String mailId, long mobileNo, double  
basicPay) {
```

```
        super(empName, empId, address, mailId, mobileNo);

        this.basicPay = basicPay;
    }

    public void generatePaySlip() {
        SalaryCalculator.calculateSalary("Programmer", basicPay);
    }
}

class AssistantProfessor extends Employee {
    double basicPay;

    public AssistantProfessor(String empName, String empId, String address, String mailId, long mobileNo,
double basicPay) {
        super(empName, empId, address, mailId, mobileNo);
        this.basicPay = basicPay;
    }

    public void generatePaySlip() {
        SalaryCalculator.calculateSalary("Assistant Professor", basicPay);
    }
}

class AssociateProfessor extends Employee {
    double basicPay;

    public AssociateProfessor(String empName, String empId, String address, String mailId, long mobileNo,
double basicPay) {
        super(empName, empId, address, mailId, mobileNo);
        this.basicPay = basicPay;
    }
}
```

```
}

public void generatePaySlip() {
    SalaryCalculator.calculateSalary("Associate Professor", basicPay);
}

}

class Professor extends Employee {
    double basicPay;

    public Professor(String empName, String empId, String address, String mailId, long mobileNo, double
basicPay) {
        super(empName, empId, address, mailId, mobileNo);
        this.basicPay = basicPay;
    }

    public void generatePaySlip() {
        SalaryCalculator.calculateSalary("Professor", basicPay);
    }
}

class SalaryCalculator {
    public static void calculateSalary(String designation, double basicPay) {
        double da = 0.97 * basicPay;
        double hra = 0.10 * basicPay;
        double pf = 0.12 * basicPay;
        double staffClubFund = 0.001 * basicPay;
        double grossSalary = basicPay + da + hra;
        double netSalary = grossSalary - (pf + staffClubFund);
    }
}
```

```
System.out.println("\n--- Pay Slip for " + designation + " ---");

System.out.println("Basic Pay: " + basicPay);

System.out.println("DA (97%): " + da);

System.out.println("HRA (10%): " + hra);

System.out.println("PF (12%): " + pf);

System.out.println("Staff Club Fund (0.1%): " + staffClubFund);

System.out.println("Gross Salary: " + grossSalary);

System.out.println("Net Salary: " + netSalary);

System.out.println("-----\n");
}
}

public class HierarchicalInheritanceDemo {

    public static void main(String[] args) {

        Programmer p = new Programmer("Alice", "P1001", "123 Main St", "alice@mail.com", 9876543210L,
50000);

        AssistantProfessor ap = new AssistantProfessor("Bob", "AP1002", "456 Elm St", "bob@mail.com",
9876543211L, 60000);

        AssociateProfessor asp = new AssociateProfessor("Charlie", "ASP1003", "789 Oak St",
"charlie@mail.com", 9876543212L, 70000);

        Professor prof = new Professor("David", "PROF1004", "101 Maple St", "david@mail.com", 9876543213L,
90000);

        p.displayEmployeeDetails();

        p.generatePaySlip();

        ap.displayEmployeeDetails();

        ap.generatePaySlip();

        asp.displayEmployeeDetails();
```



```
asp.generatePaySlip();
```

```
prof.displayEmployeeDetails();
```

```
prof.generatePaySlip();
```

```
}
```

```
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac HierarchicalInheritanceDemo.java
```

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java HierarchicalInheritanceDemo
```

```
Employee ID: P1001  
Name: Alice  
Address: 123 Main St  
Mail ID: alice@mail.com  
Mobile No: 9876543210
```

```
--- Pay Slip for Programmer ---
```

```
Basic Pay: 50000.0  
DA (97%): 48500.0  
HRA (10%): 5000.0  
PF (12%): 6000.0  
Staff Club Fund (0.1%): 50.0  
Gross Salary: 103500.0  
Net Salary: 97450.0  
-----
```

```
Employee ID: AP1002  
Name: Bob  
Address: 456 Elm St  
Mail ID: bob@mail.com  
Mobile No: 9876543211
```

```
--- Pay Slip for Assistant Professor ---
```

```
Basic Pay: 60000.0  
DA (97%): 58200.0  
HRA (10%): 6000.0  
PF (12%): 7200.0  
Staff Club Fund (0.1%): 60.0  
Gross Salary: 124200.0  
Net Salary: 116940.0  
-----
```

```
Employee ID: ASP1003  
Name: Charlie  
Address: 789 Oak St  
Mail ID: charlie@mail.com  
Mobile No: 9876543212
```

```
--- Pay Slip for Associate Professor ---
```

```
Basic Pay: 70000.0  
DA (97%): 67900.0  
HRA (10%): 7000.0  
PF (12%): 8400.0  
Staff Club Fund (0.1%): 70.0  
Gross Salary: 144900.0  
Net Salary: 136430.0  
-----
```

```
Employee ID: PROF1004  
Name: David  
Address: 101 Maple St  
Mail ID: david@mail.com  
Mobile No: 9876543213
```

```
--- Pay Slip for Professor ---
```

```
Basic Pay: 90000.0  
DA (97%): 87300.0  
HRA (10%): 9000.0  
PF (12%): 10800.0  
Staff Club Fund (0.1%): 90.0  
Gross Salary: 186300.0  
Net Salary: 175410.0  
-----
```

6.b)heirarcalmath

Code:

```
import java.util.Scanner;

class MathOperations {

    int num;

    public MathOperations(int num) {

        this.num = num;

    }

    public void displayNumber() {

        System.out.println("The number is: " + num);

    }

}

class Factorial extends MathOperations {

    public Factorial(int num) {

        super(num);

    }

    public void computeFactorial() {

        long fact = 1;

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

            fact *= i;

        }

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

    }

}
```

```
}

class Fibonacci extends MathOperations {

    public Fibonacci(int num) {

        super(num);

    }

    public void generateFibonacci() {

        int a = 0, b = 1, next;

        System.out.print("Fibonacci series up to " + num + " terms: " + a + " " + b);

        for (int i = 2; i < num; i++) {

            next = a + b;

            System.out.print(" " + next);

            a = b;

            b = next;

        }

        System.out.println();

    }

}
```

```
class PrimeChecker extends MathOperations {

    public PrimeChecker(int num) {

        super(num);

    }

    public void isPrime() {

        boolean prime = num > 1;

        for (int i = 2; i <= Math.sqrt(num); i++) {

            if (num % i == 0) {
```

```
        prime = false;

        break;
    }

}

System.out.println(num + " is " + (prime ? "a Prime Number" : "not a Prime Number"));

}

}
```

```
class GCD_LCM extends MathOperations {

    int secondNum;

    public GCD_LCM(int num, int secondNum) {

        super(num);

        this.secondNum = secondNum;

    }

    public void computeGCD_LCM() {

        int a = num, b = secondNum, gcd = 1;

        for (int i = 1; i <= a && i <= b; i++) {

            if (a % i == 0 && b % i == 0)

                gcd = i;

        }

        int lcm = (a * b) / gcd;

        System.out.println("GCD of " + a + " and " + b + " is: " + gcd);

        System.out.println("LCM of " + a + " and " + b + " is: " + lcm);

    }

}
```

```
public class HierarchicalMath {
```

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
  
    System.out.print("Enter a number: ");  
  
    int num = scanner.nextInt();  
  
    Factorial factObj = new Factorial(num);  
  
    factObj.displayNumber();  
  
    factObj.computeFactorial();  
  
    Fibonacci fibObj = new Fibonacci(num);  
  
    fibObj.generateFibonacci();  
  
    PrimeChecker primeObj = new PrimeChecker(num);  
  
    primeObj.isPrime();  
  
    System.out.print("Enter another number for GCD & LCM calculation: ");  
  
    int num2 = scanner.nextInt();  
  
    GCD_LCM gcdLcmObj = new GCD_LCM(num, num2);  
  
    gcdLcmObj.computeGCD_LCM();  
}  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac HierarchicalMath.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java HierarchicalMath
Enter a number: 4
The number is: 4
Factorial of 4 is: 24
Fibonacci series up to 4 terms: 0 1 1 2
4 is not a Prime Number
Enter another number for GCD & LCM calculation: 6
GCD of 4 and 6 is: 2
LCM of 4 and 6 is: 12
```

HYBRID INHERITANCE PROGRAMS

7.a)hybrid bank

Code:

```
import java.util.Scanner;

class BankAccount {

    protected String accountHolder;

    protected double balance;


    public BankAccount(String accountHolder, double balance) {

        this.accountHolder = accountHolder;

        this.balance = balance;

    }


    public void displayBalance() {

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

    }

}

interface InvestmentServices {

    void investInMutualFunds(double amount);

}
```

```
interface LoanServices {  
    void applyForLoan(double amount);  
}  
  
class SavingsAccount extends BankAccount implements InvestmentServices {  
    public SavingsAccount(String accountHolder, double balance) {  
        super(accountHolder, balance);  
    }  
  
    public void deposit(double amount) {  
        balance += amount;  
        System.out.println("Deposited $" + amount + " into Savings Account.");  
    }  
  
    public void investInMutualFunds(double amount) {  
        if (amount > balance) {  
            System.out.println("Insufficient funds for investment!");  
        } else {  
            balance -= amount;  
            System.out.println(accountHolder + " invested $" + amount + " in Mutual Funds.");  
        }  
    }  
}  
  
class CurrentAccount extends BankAccount implements LoanServices {  
    public CurrentAccount(String accountHolder, double balance) {  
        super(accountHolder, balance);  
    }  
}
```



```
public void withdraw(double amount) {  
    if (amount > balance) {  
        System.out.println("Insufficient funds for withdrawal!");  
    } else {  
        balance -= amount;  
        System.out.println("Withdrawn $" + amount + " from Current Account.");  
    }  
}  
  
public void applyForLoan(double amount) {  
    System.out.println(accountHolder + " has applied for a loan of $" + amount);  
}  
}  
  
public class HybridBankingSystem {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.print("Enter Account Holder Name: ");  
        String name = scanner.nextLine();  
  
        System.out.print("Enter Initial Balance: ");  
        double balance = scanner.nextDouble();  
  
        SavingsAccount savings = new SavingsAccount(name, balance);  
        CurrentAccount current = new CurrentAccount(name, balance);  
  
        System.out.println("\n--- Savings Account Operations ---");
```

```
savings.deposit(500);

savings.displayBalance();

savings.investInMutualFunds(200);


System.out.println("\n--- Current Account Operations ---");

current.withdraw(300);

current.displayBalance();

current.applyForLoan(10000);

}

}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac HybridBankingSystem.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java HybridBankingSystem
Enter Account Holder Name: induuu
Enter Initial Balance: 234567865432

--- Savings Account Operations ---
Deposited $500.0 into Savings Account.
induuu's Account Balance: $2.34567865932E11
induuu invested $200.0 in Mutual Funds.

--- Current Account Operations ---
Withdrawn $300.0 from Current Account.
induuu's Account Balance: $2.34567865132E11
induuu has applied for a loan of $10000.0

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

7.b)hybrid employee

Code:

```
import java.util.*;

class Employee {

    protected String name;

    protected int id;
```

protected double salary;

public Employee(String name, int id, double salary) {

 this.name = name;

 this.id = id;

 this.salary = salary;

}

public void displayDetails() {

 System.out.println("Employee ID: " + id);

 System.out.println("Name: " + name);

 System.out.println("Salary: \$" + salary);

}

}

interface BonusCalculator {

 void calculateBonus();

}

interface PerformanceEvaluator {

 void evaluatePerformance();

}

class Manager extends Employee implements BonusCalculator {

 private int teamSize;

public Manager(String name, int id, double salary, int teamSize) {

 super(name, id, salary);

 this.teamSize = teamSize;

```
}
```

```
public void calculateBonus() {  
    double bonus = salary * 0.2;  
    System.out.println("Manager Bonus: $" + bonus);  
}
```

```
public void manageTeam() {  
    System.out.println(name + " manages a team of " + teamSize + " employees.");  
}  
}
```

```
class Developer extends Employee implements BonusCalculator, PerformanceEvaluator {  
    private int completedProjects;  
  
    public Developer(String name, int id, double salary, int completedProjects) {  
        super(name, id, salary);  
        this.completedProjects = completedProjects;  
    }
```

```
public void calculateBonus() {  
    double bonus = salary * 0.15 + (completedProjects * 500);  
    System.out.println("Developer Bonus: $" + bonus);  
}
```

```
public void evaluatePerformance() {  
    String performance = (completedProjects > 5) ? "Excellent" : "Good";  
    System.out.println(name + "'s Performance: " + performance);  
}
```

```
}
```

```
class Intern extends Employee implements PerformanceEvaluator {
```

```
    private int internshipDuration;
```

```
    public Intern(String name, int id, double salary, int internshipDuration) {
```

```
        super(name, id, salary);
```

```
        this.internshipDuration = internshipDuration;
```

```
    }
```

```
    public void evaluatePerformance() {
```

```
        String performance = (internshipDuration > 6) ? "Great Learning Progress" : "Needs Improvement";
```

```
        System.out.println(name + "'s Internship Performance: " + performance);
```

```
    }
```

```
}
```

```
public class HybridEmployeeSystem {
```

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

```
        Manager manager = new Manager("Alice", 101, 80000, 10);
```

```
        Developer developer = new Developer("Bob", 102, 60000, 7);
```

```
        Intern intern = new Intern("Charlie", 103, 20000, 5);
```

```
        System.out.println("--- Manager Details ---");
```

```
        manager.displayDetails();
```

```
        manager.calculateBonus();
```

```
        manager.manageTeam();
```

```
        System.out.println("\n--- Developer Details ---");
```

```
        developer.displayDetails();
```

```
        developer.calculateBonus();

        developer.evaluatePerformance();

        System.out.println("\n--- Intern Details ---");

        intern.displayDetails();

        intern.evaluatePerformance();

    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac HybridEmployeeSystem.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java HybridEmployeeSystem
--- Manager Details ---
Employee ID: 101
Name: Alice
Salary: $80000.0
Manager Bonus: $16000.0
Alice manages a team of 10 employees.

--- Developer Details ---
Employee ID: 102
Name: Bob
Salary: $60000.0
Developer Bonus: $12500.0
Bob's Performance: Excellent

--- Intern Details ---
Employee ID: 103
Name: Charlie
Salary: $20000.0
Charlie's Internship Performance: Needs Improvement
```

POLYMORPHISM

CONSTRUCTOR PROGRAMS

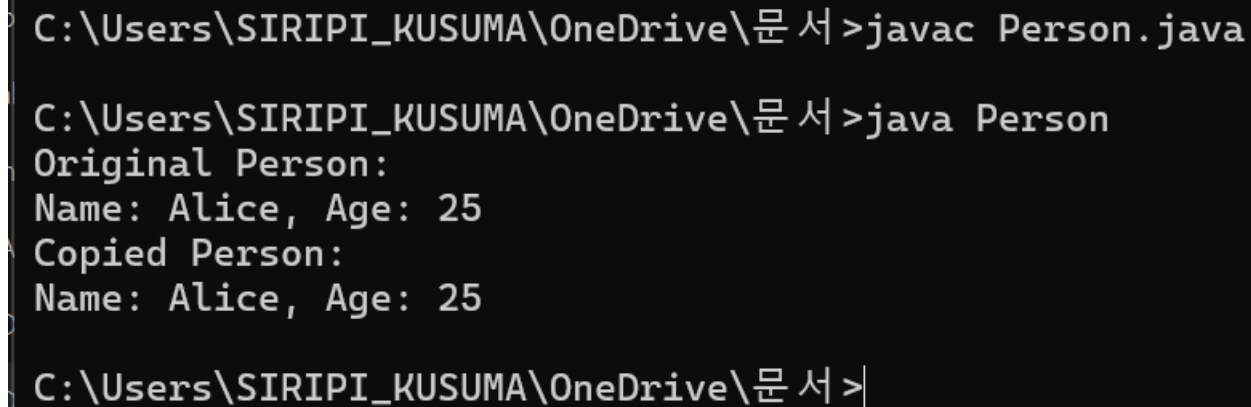
8.a)copy constructor

Code:

```
class Person {  
    String name;  
    int age;  
  
    public Person(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    public Person(Person other) {  
        this.name = other.name;  
        this.age = other.age;  
    }  
  
    public void display() {  
        System.out.println("Name: " + name + ", Age: " + age);  
    }  
  
    public static void main(String[] args) {  
        Person p1 = new Person("Alice", 25);  
        System.out.println("Original Person:");  
        p1.display();  
  
        Person p2 = new Person(p1);
```

```
        System.out.println("Copied Person:");  
        p2.display();  
    }  
}
```

Output:



```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac Person.java  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java Person  
Original Person:  
Name: Alice, Age: 25  
Copied Person:  
Name: Alice, Age: 25  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

CONSTRUCTOR OVERLOADING PROGRAMS

9.a)constructor overloading

Code:

```
class Employee {  
    String name;  
    int age;  
    double salary;  
  
    // Constructor Overloading  
    Employee() {  
        System.out.println("Default Constructor: No details provided.");  
    }  
}
```



```
Employee(String name, int age) {  
    this.name = name;  
    this.age = age;  
    System.out.println("Employee Name: " + name + ", Age: " + age);  
}
```

```
Employee(String name, int age, double salary) {  
    this.name = name;  
    this.age = age;  
    this.salary = salary;  
    System.out.println("Employee Name: " + name + ", Age: " + age + ", Salary: " + salary);  
}  
}
```

```
public class ConstructorOverloading {  
    public static void main(String[] args) {  
        Employee e1 = new Employee();  
        Employee e2 = new Employee("Alice", 25);  
        Employee e3 = new Employee("Bob", 30, 60000);  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac ConstructorOverloading.java  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java ConstructorOverloading  
Default Constructor: No details provided.  
Employee Name: Alice, Age: 25  
Employee Name: Bob, Age: 30, Salary: 60000.0  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

METHOD OVERLOADING PROGRAMS

10.a)hillstations

Code:

```
class Hillstations {  
    public void location() {  
        System.out.println("Location is unknown.");  
    }  
    public void famousfor() {  
        System.out.println("Famous for unknown attractions.");  
    }  
}  
  
class Manali extends Hillstations {  
    public void location() {  
        System.out.println("Location is: Manali is in Himachal Pradesh.");  
    }  
    public void famousfor() {  
        System.out.println("Famous for: Hadimba Temple and adventure sports.");  
    }  
}  
  
class Mussoorie extends Hillstations {  
    public void location() {  
        System.out.println("Location is: Mussoorie is in Uttarakhand.");  
    }  
    public void famousfor() {  
        System.out.println("Famous for: Kemtpy Falls and Camel's Back Road.");  
    }  
}
```

```
}
```

```
class Gulmarg extends Hillstations { // FIXED: Added opening brace
```

```
    public void location() {
```

```
        System.out.println("Location is: Gulmarg is in Jammu & Kashmir.");
```

```
    }
```

```
    public void famousfor() {
```

```
        System.out.println("Famous for: Skiing and Gondola ride.");
```

```
    }
```

```
}
```

```
public class HillstationDemo {
```

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

```
        Hillstations manali = new Manali();
```

```
        Hillstations mussoorie = new Mussoorie();
```

```
        Hillstations gulmarg = new Gulmarg();
```

```
        System.out.println("Manali Details:");
```

```
        manali.location();
```

```
        manali.famousfor(); // FIXED: Changed "Manali.famousfor()" to "manali.famousfor()"
```

```
        System.out.println("\nMussoorie Details:");
```

```
        mussoorie.location();
```

```
        mussoorie.famousfor(); // FIXED: Changed "Mussoorie.famousfor();" to "mussoorie.famousfor()"
```

```
        System.out.println("\nGulmarg Details:");
```

```
        gulmarg.location();
```

```
        gulmarg.famousfor();
```

```
    }
```

}

output:

de:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac HillstationDemo.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java HillstationDemo
Manali Details:
Location is: Manali is in Himachal Pradesh.
Famous for: Hadimba Temple and adventure sports.

Mussoorie Details:
Location is: Mussoorie is in Uttarakhand.
Famous for: Kempty Falls and Camel's Back Road.

Gulmarg Details:
Location is: Gulmarg is in Jammu & Kashmir.
Famous for: Skiing and Gondola ride.

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

10.b)Test polymor

Code:

```
class Shape {

    // Method to be overridden

    double calculateArea() {

        return 0;

    }

}

class Circle extends Shape {

    double radius;

    Circle(double radius) {

        this.radius = radius;

    }

}
```

```
}

// Overriding the method to calculate the area of a circle
@Override
double calculateArea() {
    return Math.PI * radius * radius;
}
}

class Rectangle extends Shape {
    double length, breadth;

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

    // Overriding the method to calculate the area of a rectangle
    @Override
    double calculateArea() {
        return length * breadth;
    }
}

class TestPolymorphism {
    public static void main(String[] args) {
        Shape shape1 = new Circle(5); // Circle object
        Shape shape2 = new Rectangle(10, 5); // Rectangle object
    }
}
```

```
        System.out.println("Area of Circle: " + shape1.calculateArea());  
  
        System.out.println("Area of Rectangle: " + shape2.calculateArea());  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac TestPolymorphism.java  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java TestPolymorphism  
Area of Circle: 78.53981633974483  
Area of Rectangle: 50.0  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

METHOD OVERRIDING PROGRAMS

11.a)banking

```
import java.util.Scanner;  
  
class BankAccount {  
  
    String accountNumber;  
  
    double balance;  
  
  
    BankAccount(String accountNumber, double balance) {  
  
        this.accountNumber = accountNumber;  
  
        this.balance = balance;  
  
    }  
  
  
    void deposit(double amount) {  
  
        balance += amount;  
  
        System.out.println("Deposited: $" + amount);  
  
    }  
}
```

```
void displayBalance() {  
    System.out.println("Account: " + accountNumber + ", Balance: $" + balance);  
}  
}  
  
class SavingsAccount extends BankAccount {  
    SavingsAccount(String accountNumber, double balance) {  
        super(accountNumber, balance);  
    }  
  
    void addInterest(double rate) {  
        balance += balance * (rate / 100);  
        System.out.println("Interest Added. New Balance: $" + balance);  
    }  
}  
  
class CurrentAccount extends BankAccount {  
    double overdraftLimit;  
  
    CurrentAccount(String accountNumber, double balance, double overdraftLimit) {  
        super(accountNumber, balance);  
        this.overdraftLimit = overdraftLimit;  
    }  
  
    void withdraw(double amount) {  
        if (balance - amount >= -overdraftLimit) {  
            balance -= amount;  
            System.out.println("Withdrawn: $" + amount);  
        }  
    }  
}
```

```
    } else {  
        System.out.println("Overdraft limit exceeded!");  
    }  
}  
}  
  
public class BankSystem {  
    public static void main(String[] args) {  
        SavingsAccount savings = new SavingsAccount("SA123", 5000);  
        savings.deposit(1000);  
        savings.addInterest(5);  
        savings.displayBalance();  
  
        CurrentAccount current = new CurrentAccount("CA456", 2000, 1000);  
        current.deposit(500);  
        current.withdraw(3000);  
        current.displayBalance();  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac BankSystem.java  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java BankSystem  
Deposited: $1000.0  
Interest Added. New Balance: $6300.0  
Account: SA123, Balance: $6300.0  
Deposited: $500.0  
Withdrawn: $3000.0  
Account: CA456, Balance: $-500.0  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```


11.b)payslip

Code:

```
import java.util.Scanner;

class Employee {

    String name, empId;

    double basicSalary;

    Employee(String name, String empId, double basicSalary) {

        this.name = name;

        this.empId = empId;

        this.basicSalary = basicSalary;

    }

    double calculateSalary() {

        return basicSalary;

    }

    void generatePaySlip() {

        System.out.println("\n----- PAY SLIP -----");

        System.out.println("Employee ID: " + empId);

        System.out.println("Employee Name: " + name);

        System.out.println("Basic Salary: $" + basicSalary);

        System.out.println("Total Salary: $" + calculateSalary());

        System.out.println("-----\n");

    }

}
```

```
class FullTimeEmployee extends Employee {  
    FullTimeEmployee(String name, String empId, double basicSalary) {  
        super(name, empId, basicSalary);  
    }  
  
    @Override  
    double calculateSalary() {  
        return basicSalary + 2000;  
    }  
}  
  
public class PaySlip {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        System.out.print("Enter Employee Name: ");  
        String name = sc.nextLine();  
  
        System.out.print("Enter Employee ID: ");  
        String empId = sc.nextLine();  
  
        System.out.print("Enter Basic Salary: ");  
        double basicSalary = sc.nextDouble();  
  
        FullTimeEmployee emp = new FullTimeEmployee(name, empId, basicSalary);  
        emp.generatePaySlip();  
  
        sc.close();  
    }  
}
```

```
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac PaySlip.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java PaySlip
Enter Employee Name: viswanadha reddy
Enter Employee ID: 128654321
Enter Basic Salary: 200000

----- PAY SLIP -----
Employee ID: 128654321
Employee Name: viswanadha reddy
Basic Salary: $200000.0
Total Salary: $202000.0
-----

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

ABSTRACTION

INTERFACE PROGRAMS

12.a)Arithmetic Operations

Code:

```
interface Arithmetic {

    int operation(int a, int b);

}

class Addition implements Arithmetic {

    public int operation(int a, int b) {

        return a + b;

    }

}
```

```
}
```

```
class Multiplication implements Arithmetic {
```

```
    public int operation(int a, int b) {
```

```
        return a * b;
```

```
    }
```

```
}
```

```
public class InterfaceArithmetic {
```

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

```
        Arithmetic add = new Addition();
```

```
        Arithmetic multiply = new Multiplication();
```

```
        System.out.println("Sum: " + add.operation(10, 5));
```

```
        System.out.println("Product: " + multiply.operation(10, 5));
```

```
    }
```

```
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac InterfaceArithmetic.java
```

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java InterfaceArithmetic
```

```
Sum: 15
```

```
Product: 50
```

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

12.b) Shape Calculation

Code:

```
interface Shape {
```

```
    double area();
```

```
    double perimeter();
```

}

class Circle implements Shape {

double radius;

Circle(double radius) {

this.radius = radius;

}

public double area() {

return Math.PI * radius * radius;

}

public double perimeter() {

return 2 * Math.PI * radius;

}

}

class Rectangle implements Shape {

double length, width;

Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public double area() {

return length * width;

}

```
public double perimeter() {  
    return 2 * (length + width);  
}  
}  
  
public class InterfaceShape {  
    public static void main(String[] args) {  
        Shape circle = new Circle(5);  
        Shape rectangle = new Rectangle(4, 6);  
  
        System.out.println("Circle Area: " + circle.area());  
        System.out.println("Rectangle Perimeter: " + rectangle.perimeter());  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java InterfaceShape  
Circle Area: 78.53981633974483  
Rectangle Perimeter: 20.0  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

12.c) Multiple Interface Inheritance

Code:

```
interface Engine {  
    void start();  
}
```

```
interface Vehicle {
```

```
void speedUp(int increment);  
}  
  
class Car implements Engine, Vehicle {  
    int speed;  
  
    public void start() {  
        System.out.println("Car Engine Started.");  
    }  
  
    public void speedUp(int increment) {  
        speed += increment;  
        System.out.println("Car Speed: " + speed + " km/h");  
    }  
}  
  
public class InterfaceCar {  
    public static void main(String[] args) {  
        Car myCar = new Car();  
        myCar.start();  
        myCar.speedUp(20);  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac InterfaceCar.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java InterfaceCar
Car Engine Started.
Car Speed: 20 km/h

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

12.d) Interface for Bank Transactions

Code:

```
interface BankAccount {

    void deposit(double amount);

    void withdraw(double amount);

    double getBalance();

}

class SavingsAccount implements BankAccount {

    private double balance;

    public void deposit(double amount) {

        balance += amount;

        System.out.println("Deposited: $" + amount);

    }

    public void withdraw(double amount) {

        if (amount > balance) {

            System.out.println("Insufficient Balance!");

        } else {

            balance -= amount;

            System.out.println("Withdrawn: $" + amount);

        }

    }

}
```



```
    }  
}  
  
public double getBalance() {  
    return balance;  
}  
}  
  
public class InterfaceBank {  
    public static void main(String[] args) {  
        SavingsAccount acc = new SavingsAccount();  
        acc.deposit(500);  
        acc.withdraw(200);  
        System.out.println("Current Balance: $" + acc.getBalance());  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac InterfaceBank.java  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java InterfaceBank  
Deposited: $500.0  
Withdrawn: $200.0  
Current Balance: $300.0  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

Abstract Class Programs

13.a) Abstract Class for Animal Sounds

Code:

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

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

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

public class AbstractAnimal {
    public static void main(String[] args) {
        Animal dog = new Dog();
        Animal cat = new Cat();

        dog.makeSound();
        cat.makeSound();
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac AbstractAnimal.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java AbstractAnimal
Dog barks: Woof Woof!
Cat meows: Meow Meow!

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

13.b) Abstract Class for Employee Salary Calculation

Code:

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

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

    abstract double calculateSalary();
}

class FullTimeEmployee extends Employee {
    double monthlySalary;

    FullTimeEmployee(String name, int id, double salary) {
        super(name, id);
    }
}
```

```
        this.monthlySalary = salary;
    }

    double calculateSalary() {
        return monthlySalary;
    }
}

class PartTimeEmployee extends Employee {
    double hourlyRate;
    int hoursWorked;

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

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

public class AbstractEmployee {
    public static void main(String[] args) {
        Employee emp1 = new FullTimeEmployee("Alice", 101, 5000);
        Employee emp2 = new PartTimeEmployee("Bob", 102, 20, 120);

        System.out.println("Alice's Salary: $" + emp1.calculateSalary());
        System.out.println("Bob's Salary: $" + emp2.calculateSalary());
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac AbstractEmployee.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java AbstractEmployee
Alice's Salary: $5000.0
Bob's Salary: $2400.0

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

13.c) Abstract Class for Vehicle Features

Code:

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

class Bike extends Vehicle {
    void start() {
```

```
        System.out.println("Bike Started.");
    }

    void stop() {
        System.out.println("Bike Stopped.");
    }
}

class Truck extends Vehicle {
    void start() {
        System.out.println("Truck Engine Started.");
    }

    void stop() {
        System.out.println("Truck Stopped.");
    }
}

public class AbstractVehicle {
    public static void main(String[] args) {
        Vehicle bike = new Bike();
        Vehicle truck = new Truck();

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

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac AbstractVehicle.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java AbstractVehicle
Bike Started.
Truck Engine Started.
Bike Stopped.

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

13.d) Abstract Class for Student Result Calculation

Code:

```
abstract class Student {
    String name;
    int rollNumber;

    Student(String name, int rollNumber) {
        this.name = name;
        this.rollNumber = rollNumber;
    }

    abstract void calculateGrade();
}
```

```
}

class EngineeringStudent extends Student {
    double marks;

    EngineeringStudent(String name, int rollNumber, double marks) {
        super(name, rollNumber);
        this.marks = marks;
    }

    void calculateGrade() {
        if (marks >= 90) {
            System.out.println(name + "'s Grade: A");
        } else if (marks >= 75) {
            System.out.println(name + "'s Grade: B");
        } else {
            System.out.println(name + "'s Grade: C");
        }
    }
}

class MedicalStudent extends Student {
    double marks;

    MedicalStudent(String name, int rollNumber, double marks) {
        super(name, rollNumber);
        this.marks = marks;
    }

    void calculateGrade() {
        if (marks >= 85) {
            System.out.println(name + "'s Grade: A");
        } else if (marks >= 70) {
            System.out.println(name + "'s Grade: B");
        } else {
            System.out.println(name + "'s Grade: C");
        }
    }
}

public class AbstractStudent {
    public static void main(String[] args) {
        Student enggStudent = new EngineeringStudent("John", 201, 88);
        Student medStudent = new MedicalStudent("Alice", 301, 92);

        enggStudent.calculateGrade();
        medStudent.calculateGrade();
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac AbstractStudent.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java AbstractStudent
John's Grade: B
Alice's Grade: A

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

ENCAPSULATION

14.a) Bank Account

Code:

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

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

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

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

    public double getBalance() {
        return balance;
    }
}

public class EncapsulationBank {
    public static void main(String[] args) {
        BankAccount acc = new BankAccount("123456", 1000);
        acc.deposit(500);
        acc.withdraw(200);
    }
}
```

```
        System.out.println("Final Balance: $" + acc.getBalance());  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac EncapsulationBank.java  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java EncapsulationBank  
Deposited: $500.0  
Withdrawn: $200.0  
Final Balance: $1300.0  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

14.b) Student Data

Code:

```
class Student {  
    private String name;  
    private int rollNumber;  
    private double marks;  
  
    public Student(String name, int rollNumber, double marks) {  
        this.name = name;  
        this.rollNumber = rollNumber;  
        this.marks = marks;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public int getRollNumber() {  
        return rollNumber;  
    }  
  
    public double getMarks() {  
        return marks;  
    }  
  
    public void setMarks(double marks) {  
        if (marks >= 0 && marks <= 100) {  
            this.marks = marks;  
        } else {  
            System.out.println("Invalid marks!");  
        }  
    }  
}  
  
public class EncapsulationStudent {  
    public static void main(String[] args) {
```

```
Student student = new Student("Alice", 101, 88.5);
System.out.println("Student Name: " + student.getName());
System.out.println("Roll Number: " + student.getRollNumber());
student.setMarks(92);
System.out.println("Updated Marks: " + student.getMarks());
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac EncapsulationStudent.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java EncapsulationStudent
Student Name: Alice
Roll Number: 101
Updated Marks: 92.0
```

14.c) Employee Details

Code:

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

    public Employee(String empName, int empID, double salary) {
        this.empName = empName;
        this.empID = empID;
        this.salary = salary;
    }

    public String getEmpName() {
        return empName;
    }

    public int getEmpID() {
        return empID;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        if (salary > 0) {
            this.salary = salary;
        } else {
            System.out.println("Invalid Salary Amount!");
        }
    }
}

public class EncapsulationEmployee {
    public static void main(String[] args) {
        Employee emp = new Employee("John", 5001, 50000);
    }
}
```



```
System.out.println("Employee Name: " + emp.getEmpName());  
System.out.println("Employee ID: " + emp.getEmpID());  
emp.setSalary(55000);  
System.out.println("Updated Salary: " + emp.getSalary());  
}  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac EncapsulationEmployee.java  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java EncapsulationEmployee  
Employee Name: John  
Employee ID: 5001  
Updated Salary: 55000.0  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>|
```

14.d) Car Model

Code:

```
class Car {  
    private String model;  
    private double price;  
  
    public Car(String model, double price) {  
        this.model = model;  
        this.price = price;  
    }  
  
    public String getModel() {  
        return model;  
    }  
  
    public double getPrice() {  
        return price;  
    }  
  
    public void setPrice(double price) {  
        if (price > 0) {  
            this.price = price;  
        } else {  
            System.out.println("Invalid Price!");  
        }  
    }  
}  
  
public class EncapsulationCar {  
    public static void main(String[] args) {  
        Car car = new Car("Tesla Model 3", 50000);  
        System.out.println("Car Model: " + car.getModel());  
        car.setPrice(52000);  
        System.out.println("Updated Price: $" + car.getPrice());  
    }  
}
```

```
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac EncapsulationCar.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java EncapsulationCar
Car Model: Tesla Model 3
Updated Price: $52000.0
```

Packages Programs

15.a) User-Defined Package - CalculatorCode 1:

```
package mypackage;
```

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

    public int multiply(int a, int b) {
        return a * b;
    }
}
```

Code 2:

```
import mypackage.Calculator;
```

```
public class PackageDemo {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Sum: " + calc.add(10, 5));
        System.out.println("Product: " + calc.multiply(10, 5));
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac -d . Calculator.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac PackageDemo.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java PackageDemo
Sum: 15
Product: 50

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

15.b) User-Defined Package - Student Info

Code1:

```
package schoolpackage;
```

```
public class StudentInfo {  
    public void displayInfo(String name, int rollNumber) {  
        System.out.println("Student Name: " + name);  
        System.out.println("Roll Number: " + rollNumber);  
    }  
}
```

Code2:

```
import schoolpackage.StudentInfo;
```

```
public class PackageDemo2 {  
    public static void main(String[] args) {  
        StudentInfo student = new StudentInfo();  
        student.displayInfo("Alice", 101);  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac PackageDemo2.java  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java PackageDemo2  
Student Name: Alice  
Roll Number: 101  
  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

15.c) Built-in Package - Using java.util & java.io

Code:

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

```
public class BuiltInPackageDemo {  
    public static void main(String[] args) throws IOException {  
        Scanner sc = new Scanner(System.in);  
        System.out.print("Enter your name: ");  
        String name = sc.nextLine();  
  
        File file = new File("test.txt");  
        FileWriter writer = new FileWriter(file);  
        writer.write("Hello " + name);  
        writer.close();  
  
        System.out.println("Data written to file.");  
        sc.close();  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac BuiltInPackageDemo.java
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java BuiltInPackageDemo
Enter your name: kusuma
Data written to file.
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

15.d) Built-in Package - Using java.time, java.lang & java.util

Code:

```
import java.time.LocalDate;
import java.util.Random;

public class BuiltInPackageExample {
    public static void main(String[] args) {
        LocalDate today = LocalDate.now();
        System.out.println("Today's Date: " + today);

        Random rand = new Random();
        System.out.println("Random Number: " + rand.nextInt(100));

        System.out.println("Math Square Root of 25: " + Math.sqrt(25));
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac BuiltInPackageExample.java
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java BuiltInPackageExample
Today's Date: 2025-04-05
Random Number: 66
Math Square Root of 25: 5.0
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```

Exception Handling

16.a) Nested Try-Catch Blocks

```
public class NestedTryCatchExample {  
    public static void main(String[] args) {  
        try {  
            int[] arr = {10, 20, 30};  
            try {  
                System.out.println(arr[5]);  
            } catch (ArrayIndexOutOfBoundsException e) {  
                System.out.println("Inner Catch: Array index is out of bounds.");  
            }  
            int result = 10 / 0;  
        } catch (ArithmeticException e) {  
            System.out.println("Outer Catch: Cannot divide by zero.");  
        }  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac NestedTryCatchExample.java  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java NestedTryCatchExample  
Inner Catch: Array index is out of bounds.  
Outer Catch: Cannot divide by zero.
```

16.b) Throwing Custom

Code:

```
class AgeException extends Exception {  
    public AgeException(String message) {  
        super(message);  
    }  
}  
  
class Voter {  
    void checkEligibility(int age) throws AgeException {  
        if (age < 18) {  
            throw new AgeException("Age is less than 18. You are not eligible to vote.");  
        } else if (age > 120) {  
            throw new AgeException("Age cannot be more than 120. Invalid input.");  
        } else {  
            System.out.println("You are eligible to vote.");  
        }  
    }  
}  
  
public class CustomExceptionDemo {  
    public static void main(String[] args) {  
        Voter voter = new Voter();  
        try {  
            voter.checkEligibility(150);  
        } catch (AgeException e) {
```

```
        System.out.println("Exception: " + e.getMessage());
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac CustomExceptionDemo.java
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java CustomExceptionDemo
Exception: Age cannot be more than 120. Invalid input.
```

16.c) Exception Propagation

Code:

```
public class ExceptionPropagationExample {
    void method1() throws ArithmeticException {
        method2();
    }

    void method2() throws ArithmeticException {
        method3();
    }

    void method3() throws ArithmeticException {
        throw new ArithmeticException("Exception thrown from method3.");
    }

    public static void main(String[] args) {
        ExceptionPropagationExample obj = new ExceptionPropagationExample();
        try {
            obj.method1();
        } catch (ArithmeticException e) {
            System.out.println("Exception caught in main: " + e.getMessage());
        }
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac ExceptionPropagationExample.java
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java ExceptionPropagationExample
Exception caught in main: Exception thrown from method3.
```

16.d) Rethrowing Exception

Code:

```
public class RethrowExceptionExample {

    static void level1() throws Exception {
```

```
try {
    level2();
} catch (Exception e) {
    System.out.println("Exception caught in level1, rethrowing...");
    throw e; // Rethrowing the same exception
}

static void level2() throws Exception {
    try {
        level3();
    } catch (Exception e) {
        System.out.println("Exception caught in level2, rethrowing...");
        throw e; // Rethrowing the same exception
    }
}

static void level3() throws Exception {
    throw new Exception("Exception thrown at level3");
}

public static void main(String[] args) {
    try {
        level1();
    } catch (Exception e) {
        System.out.println("Exception finally handled in main: " + e.getMessage());
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac RethrowExceptionExample.java

C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java RethrowExceptionExample
Exception caught in level2, rethrowing...
Exception caught in level1, rethrowing...
Exception finally handled in main: Exception thrown at level3
```

File Handling

17.a) Copying Content from One File to Another

Code:

```
import java.io.*;

public class CopyFileExample {
    public static void main(String[] args) {
        try (BufferedReader reader = new BufferedReader(new FileReader("input.txt"));
            BufferedWriter writer = new BufferedWriter(new FileWriter("output.txt"))) {

            String line;
            while ((line = reader.readLine()) != null) {
                writer.write(line);
                writer.newLine();
            }
            System.out.println("File copied successfully.");
        } catch (IOException e) {
            System.out.println("Error occurred: " + e.getMessage());
        }
    }
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac CopyFileExample.java
```

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java CopyFileExample
Error occurred: input.txt (The system cannot find the file specified)
```

17.b) Counting Words and Characters in a File

Code;

```
import java.io.*;
import java.util.StringTokenizer;

public class FileWordCharacterCount {
    public static void main(String[] args) {
        try (BufferedReader reader = new BufferedReader(new FileReader("sample.txt"))) {
            int charCount = 0, wordCount = 0, lineCount = 0;
            String line;

            while ((line = reader.readLine()) != null) {
                lineCount++;
                charCount += line.length();
                StringTokenizer tokens = new StringTokenizer(line);
                wordCount += tokens.countTokens();
            }

            System.out.println("Lines: " + lineCount);
            System.out.println("Words: " + wordCount);
            System.out.println("Characters: " + charCount);
        } catch (IOException e) {
            System.out.println("File not found: " + e.getMessage());
        }
    }
}
```

Output:


```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac FileWordCharacterCount.java  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java FileWordCharacterCount  
File not found: sample.txt (The system cannot find the file specified)
```

17.c) Appending Data

Code:

```
import java.io.FileWriter;  
import java.io.IOException;  
  
public class AppendToFileExample {  
    public static void main(String[] args) {  
        try (FileWriter writer = new FileWriter("data.txt", true)) {  
            writer.write("\nThis is appended text.");  
            System.out.println("Data appended successfully.");  
        } catch (IOException e) {  
            System.out.println("An error occurred: " + e.getMessage());  
        }  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac AppendToFileExample.java  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java AppendToFileExample  
Data appended successfully.
```

17.d) Reading a File Using a BufferedReader

Code:

```
import java.io.*;  
  
public class BufferedReaderExample {  
    public static void main(String[] args) {  
        try (BufferedReader br = new BufferedReader(new FileReader("example.txt"))) {  
            String line;  
            while ((line = br.readLine()) != null) {  
                System.out.println(line);  
            }  
        } catch (IOException e) {  
            System.out.println("Error reading file: " + e.getMessage());  
        }  
    }  
}
```

Output:

```
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>javac BufferedReaderExample.java  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>java BufferedReaderExample  
Error reading file: example.txt (The system cannot find the file specified)  
C:\Users\SIRIPI_KUSUMA\OneDrive\문서>
```