Assignment 1

1. **Write a Java program to perform matrix multiplication on a two-dimensional array. Read matrices A and B from the user and store the result in matrix C.**

```
import java.util.Scanner;

public class A1Q1 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);



        System.out.println("Enter the rows and columns of Matrix A:");

        int rowsA = scanner.nextInt();

        int colsA = scanner.nextInt();



        System.out.println("Enter the rows and columns of Matrix B:");

        int rowsB = scanner.nextInt();

        int colsB = scanner.nextInt();



        if (colsA != rowsB) {

            System.out.println("Matrix multiplication not possible");

            scanner.close();

            return;

        }



        int[][] A = new int[rowsA][colsA];

        int[][] B = new int[rowsB][colsB];

        int[][] C = new int[rowsA][colsB];
```

```
System.out.println("Enter elements of Matrix A:");
for (int i = 0; i < rowsA; i++)
    for (int j = 0; j < colsA; j++)
        A[i][j] = scanner.nextInt();

System.out.println("Enter elements of Matrix B:");
for (int i = 0; i < rowsB; i++)
    for (int j = 0; j < colsB; j++)
        B[i][j] = scanner.nextInt();

// Matrix Multiplication Logic
for (int i = 0; i < rowsA; i++) {
    for (int j = 0; j < colsB; j++) {
        C[i][j] = 0;  // Initialize to 0
        for (int k = 0; k < colsA; k++) {
            C[i][j] += A[i][k] * B[k][j];
        }
    }
}
System.out.println("Resulting Matrix:");
for (int i = 0; i < rowsA; i++) {
    for (int j = 0; j < colsB; j++) {
        System.out.print(C[i][j] + " ");
    }
    System.out.println();
```

```
        }


        scanner.close();
    }
}
```

2. **Write a Java program to store and display the transpose of a matrix with 4 rows and 6 columns.**

```
import java.util.Scanner;

public class A1Q2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        int rows = 4, cols = 6;
        int[][] matrix = new int[rows][cols];


        System.out.println("Enter the elements of the 4x6 matrix:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }


        // Transpose the matrix
        int[][] transpose = new int[cols][rows];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
```

```
            transpose[j][i] = matrix[i][j]; // Swap row and column
        }
    }
    // Display the transposed matrix
    System.out.println("Transpose of the matrix:");
    for (int i = 0; i < cols; i++) {
        for (int j = 0; j < rows; j++) {
            System.out.print(transpose[i][j] + " ");
        }
        System.out.println();
    }
    scanner.close();
  }
}
```

3. **Write a function in Java to check if a number is an Armstrong number. (Read the number from the command line argument).**

```
public class a1q3 {
  public static void main(String[] args) {
    if (args.length != 1) {
        System.out.println("Enter a number as an argument.");
        return;
    }

    int number = Integer.parseInt(args[0]);
    int originalNumber = number;
    int sum = 0;
```

```
        while (number > 0) {
            int digit = number % 10;
            sum += digit * digit * digit;
            number /= 10;
        }

        if (sum == originalNumber) {
            System.out.println(originalNumber + " is an Armstrong number.");
        } else {
            System.out.println(originalNumber + " is not an Armstrong number.");
        }
    }
}
```

4. **Create two packages: Trigonometry and Statistics. Implement the following functions:**
    - **Trigonometry package:** `sine(angle), cosine(angle), tan(angle)`
    - **Statistics package:** `mean(A[]), median(A[]), mode(A[])`
    - **Access and display the calculated values of all functions.**

Step 1: ***`TrigFunctions.java`***

```
package Trigonometry;

public class TrigFunctions {
    public static double sine(double angle) {
        return Math.sin(Math.toRadians(angle));
    }
```

```
    public static double cosine(double angle) {
        return Math.cos(Math.toRadians(angle));
    }

    public static double tan(double angle) {
        return Math.tan(Math.toRadians(angle));
    }
}
```

Step 2: StatsFunctions.java

```
package Statistics;
import java.util.Arrays;
import java.util.HashMap;

public class StatsFunctions {
    public static double mean(int[] A) {
        double sum = 0;
        for (int num : A) {
            sum += num;
        }
        return sum / A.length;
    }

    public static double median(int[] A) {
        Arrays.sort(A);
        int n = A.length;
```

```
        if (n % 2 == 0) {
            return (A[n / 2 - 1] + A[n / 2]) / 2.0;
        } else {
            return A[n / 2];
        }
    }

    public static int mode(int[] A) {
        HashMap<Integer, Integer> freqMap = new HashMap<>();
        for (int num : A) {
            freqMap.put(num, freqMap.getOrDefault(num, 0) + 1);
        }
        int mode = A[0], maxCount = 0;
        for (int num : freqMap.keySet()) {
            if (freqMap.get(num) > maxCount) {
                maxCount = freqMap.get(num);
                mode = num;
            }
        }
        return mode;
    }
}
```

File 3: `a1q4.java`

```
import Trigonometry.TrigFunctions;
import Statistics.StatsFunctions;
```

```
import java.util.Scanner;


public class a1q4 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);


        // Trigonometry calculations
        System.out.print("Enter angle in degrees: ");
        double angle = sc.nextDouble();
        System.out.println("Sine: " + TrigFunctions.sine(angle));
        System.out.println("Cosine: " + TrigFunctions.cosine(angle));
        System.out.println("Tangent: " + TrigFunctions.tan(angle));


        // Statistics calculations
        System.out.print("Enter number of elements in array: ");
        int n = sc.nextInt();
        int[] A = new int[n];
        System.out.println("Enter elements:");
        for (int i = 0; i < n; i++) {
            A[i] = sc.nextInt();
        }
        System.out.println("Mean: " + StatsFunctions.mean(A));
        System.out.println("Median: " + StatsFunctions.median(A));
        System.out.println("Mode: " + StatsFunctions.mode(A));
        sc.close();    }}
```

3) Compile the packages first

```
javac -d . Trigonometry/TrigFunctions.java
javac -d . Statistics/StatsFunctions.java
javac a1q4.java
```

Assignment 2

SET A

1. **Create a superclass Employee (id, name). Inherit a class Manager (dept, salary) from Employee. Inherit a class SalesManager (bonus) from Manager. Override the salary() method in SalesManager. Display all details along with the salary. *(Use default and parameterized constructors and toString() method).***

```
// Superclass: Employee

class Employee {

    int id;

    String name;



    // Default Constructor

    public Employee() {

        this.id = 0;

        this.name = "Unknown";

    }



    // Parameterized Constructor

    public Employee(int id, String name) {

        this.id = id;

        this.name = name;
```

```
    }

    // toString() Method
    @Override
    public String toString() {
        return "Employee ID: " + id + "\nName: " + name;
    }
}

// Subclass: Manager (inherits Employee)
class Manager extends Employee {
    String dept;
    double salary;

    // Default Constructor
    public Manager() {
        super();
        this.dept = "Not Assigned";
        this.salary = 0.0;
    }

    // Parameterized Constructor
    public Manager(int id, String name, String dept, double salary) {
        super(id, name);
        this.dept = dept;
```

```
        this.salary = salary;
    }

    // Method to Calculate Salary
    public double calculateSalary() {
        return salary;
    }

    // toString() Method
    @Override
    public String toString() {
        return super.toString() + "\nDepartment: " + dept + "\nSalary: " + salary;
    }
}

// Subclass: SalesManager (inherits Manager)
class SalesManager extends Manager {
    double bonus;

    // Default Constructor
    public SalesManager() {
        super();
        this.bonus = 0.0;
    }
```

```
    // Parameterized Constructor
    public SalesManager(int id, String name, String dept, double salary, double
bonus) {
        super(id, name, dept, salary);
        this.bonus = bonus;
    }

    // Override calculateSalary() to include bonus
    @Override
    public double calculateSalary() {
        return salary + bonus;
    }

    // toString() Method
    @Override
    public String toString() {
        return super.toString() + "\nBonus: " + bonus + "\nTotal Salary: " +
calculateSalary();
    }
}

// Main Class
public class a2a1q1 {
    public static void main(String[] args) {
        // Creating a Manager object
        Manager m = new Manager(101, "Sham", "HR", 50000);
```

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

        // Creating a SalesManager object
        SalesManager sm = new SalesManager(102, "Raj", "Sales", 60000, 10000);
        System.out.println("SalesManager Details:");
        System.out.println(sm);
    }
}
```

2. **Create a superclass Account (acno, name, bal). Inherit Savings (min_bal, tran_limit, int_rate) and Current (min_bal). Implement withdraw() and deposit() functions. Check transaction limits for the Savings account. *(Use default and parameterized constructors and toString() method).***

```
// Superclass: Account
class Account {
    int acno;
    String name;
    double bal;

    // Default Constructor
    public Account() {
        this.acno = 0;
        this.name = "Unknown";
        this.bal = 0.0;
    }
```

```
    // Parameterized Constructor
    public Account(int acno, String name, double bal) {
        this.acno = acno;
        this.name = name;
        this.bal = bal;
    }

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

    // toString() Method
    @Override
    public String toString() {
        return "Account No: " + acno + "\nName: " + name + "\nBalance: " + bal;
    }
}

// Subclass: Savings (inherits Account)
```

```
class Savings extends Account {
    double min_bal;
    int tran_limit;
    double int_rate;
    int transactionCount;

    // Default Constructor
    public Savings() {
        super();
        this.min_bal = 1000;
        this.tran_limit = 3;
        this.int_rate = 4.0;
        this.transactionCount = 0;
    }

    // Parameterized Constructor
    public Savings(int acno, String name, double bal, double min_bal, int tran_limit,
double int_rate) {
        super(acno, name, bal);
        this.min_bal = min_bal;
        this.tran_limit = tran_limit;
        this.int_rate = int_rate;
        this.transactionCount = 0;
    }

    // Withdraw method (with transaction limit check)
```

```
    public void withdraw(double amount) {
        if (transactionCount >= tran_limit) {
            System.out.println("Transaction limit reached! Cannot withdraw.");
            return;
        }
        if (bal - amount < min_bal) {
            System.out.println("Insufficient balance! Cannot withdraw below minimum balance.");
        } else {
            bal -= amount;
            transactionCount++;
            System.out.println("Withdrawn: " + amount);
        }
    }

    // toString() Method
    @Override
    public String toString() {
        return super.toString() + "\nMinimum Balance: " + min_bal +
                "\nTransaction Limit: " + tran_limit +
                "\nInterest Rate: " + int_rate + "%\nTransactions Made: " + transactionCount;
    }
}

// Subclass: Current (inherits Account)
```

```
class Current extends Account {
    double min_bal;

    // Default Constructor
    public Current() {
        super();
        this.min_bal = 5000;
    }

    // Parameterized Constructor
    public Current(int acno, String name, double bal, double min_bal) {
        super(acno, name, bal);
        this.min_bal = min_bal;
    }

    // Withdraw method (without transaction limit)
    public void withdraw(double amount) {
        if (bal - amount < min_bal) {
            System.out.println("Insufficient balance! Cannot withdraw below minimum balance.");
        } else {
            bal -= amount;
            System.out.println("Withdrawn: " + amount);
        }
    }
```

```
    // toString() Method
    @Override
    public String toString() {
        return super.toString() + "\nMinimum Balance: " + min_bal;
    }
}

// Main Class
public class a2a1q2 {
    public static void main(String[] args) {
        // Create Savings Account
        Savings sAcc = new Savings(101, "Ram", 5000, 1000, 3, 4.0);
        System.out.println("Savings Account Details:");
        System.out.println(sAcc);
        sAcc.deposit(2000);
        sAcc.withdraw(500);
        sAcc.withdraw(1000);
        sAcc.withdraw(1500); // Exceeds limit
        sAcc.withdraw(100);  // Should not allow
        System.out.println("-------------------------------");

        // Create Current Account
        Current cAcc = new Current(202, "Raj", 10000, 5000);
        System.out.println("Current Account Details:");
        System.out.println(cAcc);
```

```
        cAcc.deposit(3000);
        cAcc.withdraw(7000);
        cAcc.withdraw(2000); // Should not allow
    }
}
```

3. **Create an abstract class Shape with abstract functions area() and volume(). Inherit Cone and Cylinder from Shape. Create objects and calculate the area and volume for both. *(Use default and parameterized constructors and toString() method).***

```
// Abstract class Shape
abstract class Shape {
    abstract double area();
    abstract double volume();
}

// Cone class (inherits Shape)
class Cone extends Shape {
    double radius, height;

    // Default Constructor
    public Cone() {
        this.radius = 1;
        this.height = 1;
    }

    // Parameterized Constructor
    public Cone(double radius, double height) {
```

```
        this.radius = radius;
        this.height = height;
    }

    // Override area() method
    @Override
    public double area() {
        double slantHeight = Math.sqrt((radius * radius) + (height * height));
        return Math.PI * radius * (radius + slantHeight);
    }

    // Override volume() method
    @Override
    public double volume() {
        return (Math.PI * radius * radius * height) / 3;
    }

    // toString() Method
    @Override
    public String toString() {
        return "Cone:\nRadius: " + radius + "\nHeight: " + height +
                "\nSurface Area: " + String.format("%.2f", area()) +
                "\nVolume: " + String.format("%.2f", volume());
    }
}
```

```
// Cylinder class (inherits Shape)
class Cylinder extends Shape {
    double radius, height;

    // Default Constructor
    public Cylinder() {
        this.radius = 1;
        this.height = 1;
    }

    // Parameterized Constructor
    public Cylinder(double radius, double height) {
        this.radius = radius;
        this.height = height;
    }

    // Override area() method
    @Override
    public double area() {
        return 2 * Math.PI * radius * (radius + height);
    }

    // Override volume() method
    @Override
```

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

    // toString() Method
    @Override
    public String toString() {
        return "Cylinder:\nRadius: " + radius + "\nHeight: " + height +
                "\nSurface Area: " + String.format("%.2f", area()) +
                "\nVolume: " + String.format("%.2f", volume());
    }
}

// Main class
public class a2a1q3 {
    public static void main(String[] args) {
        // Creating Cone object
        Cone cone = new Cone(3, 5);
        System.out.println(cone);
        System.out.println("------------------------");

        // Creating Cylinder object
        Cylinder cylinder = new Cylinder(3, 5);
        System.out.println(cylinder);
    }
```

```
}
```

4. **Create an interface Polygon with an abstract function area(). Implement Square (side) and Rectangle (length, width) from Polygon. Calculate the area of both objects.**

```
// Interface Polygon
interface Polygon {
    double area();  // Abstract method
}


// Square class implementing Polygon
class Square implements Polygon {
    double side;

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

    // Override area() method
    @Override
    public double area() {
        return side * side;
    }

    // toString() Method
    public String toString() {
```

```
        return "Square:\nSide: " + side + "\nArea: " + area();
    }
}

// Rectangle class implementing Polygon
class Rectangle implements Polygon {
    double length, width;

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

    // Override area() method
    @Override
    public double area() {
        return length * width;
    }

    // toString() Method
    public String toString() {
        return "Rectangle:\nLength: " + length + "\nWidth: " + width + "\nArea: " +
area();
    }
}
```

```
// Main class
public class a2a1q4 {
    public static void main(String[] args) {
        // Creating Square object
        Square square = new Square(4);
        System.out.println(square);
        System.out.println("------------------------");

        // Creating Rectangle object
        Rectangle rectangle = new Rectangle(5, 3);
        System.out.println(rectangle);
    }
}
```

SET B

5. **Create an ArrayList of Days. Add 7 days to it. Display the size and all days using an iterator. Check if “Wed” is present. Delete the 3rd and 5th day.**

```
import java.util.ArrayList;
import java.util.Iterator;

public class a2a1q5 {
    public static void main(String[] args) {
        // Create an ArrayList to store days of the week
        ArrayList<String> days = new ArrayList<>();
```

```
        // Adding 7 days to the ArrayList
        days.add("Sun");
        days.add("Mon");
        days.add("Tue");
        days.add("Wed");
        days.add("Thu");
        days.add("Fri");
        days.add("Sat");

        // Display the size of the list
        System.out.println("Size of the list: " + days.size());

        // Display all days using Iterator
        System.out.println("Days in the list:");
        Iterator<String> it = days.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }

        // Check if "Wed" is present
        System.out.println(days.contains("Wed") ? "\"Wed\" is present in the list." :
"\"Wed\" is not present in the list.");

        // Remove the 3rd (index 2) and 5th (index 4) day
        if (days.size() > 2) days.remove(2); // Removing "Tue"
        if (days.size() > 4) days.remove(4); // Removing "Fri" (after shift)
```

```
        // Display the updated list
        System.out.println("Updated list after deletions:");
        for (String day : days) {
            System.out.println(day);
        }
    }
}
```

6. **Create a LinkedList of Colors. Add 4 colors. Display all colors using an iterator. Add two colors at the beginning. Remove the last color.**

```
import java.util.LinkedList;
import java.util.Iterator;

public class a2a1q6 {
    public static void main(String[] args) {
        // Create a LinkedList to store colors
        LinkedList<String> colors = new LinkedList<>();

        // Add 4 colors to the LinkedList
        colors.add("Red");
        colors.add("Blue");
        colors.add("Green");
        colors.add("Yellow");

        // Display all colors using Iterator
        System.out.println("Original Colors List:");
```

```
        Iterator<String> it = colors.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }

        // Add two new colors at the beginning
        colors.addFirst("Purple");
        colors.addFirst("Orange");

        // Remove the last color safely
        if (!colors.isEmpty()) {
            colors.removeLast();
        }

        // Display updated list
        System.out.println("\nUpdated Colors List:");
        for (String color : colors) {
            System.out.println(color);
        }
    }
}
```

7. **Create a Vector class. Print its capacity and check if it's empty. Add 4 objects (Integer, Float, String, StringBuffer). Display all elements and delete all.**

```
import java.util.Vector;
```

```
public class a2a1q7 {
    public static void main(String[] args) {
        // Create a Vector
        Vector<Object> vector = new Vector<>();

        // Print the initial capacity of the Vector
        System.out.println("Initial Capacity: " + vector.capacity());

        // Check if the Vector is empty
        System.out.println("Is the vector empty? " + vector.isEmpty());

        // Add 4 different types of objects
        vector.add(10);                    // Integer
        vector.add(20.5f);                 // Float
        vector.add("Hello World");         // String
        vector.add(new StringBuffer("Java"));  // StringBuffer

        // Display all elements in the Vector
        System.out.println("\nElements in the Vector:");
        for (Object obj : vector) {
            System.out.println(obj);
        }

        // Delete all elements from the Vector
        vector.clear();
```

```
        // Check if the Vector is empty after deletion
        System.out.println("\nVector after deletion. Is it empty? " + vector.isEmpty());
    }
}
```

8. **Create a Hashtable named Student with <rollno, name>. Add 5 records. Display the Hashtable. Remove the student with the name “John”.**

```
import java.util.Hashtable;
import java.util.Enumeration;
import java.util.Map;

public class a2a1q8 {
    public static void main(String[] args) {
        // Create a Hashtable with <Integer, String> for roll number and name
        Hashtable<Integer, String> students = new Hashtable<>();

        // Add 5 student records
        students.put(101, "Alice");
        students.put(102, "Bob");
        students.put(103, "John");
        students.put(104, "David");
        students.put(105, "Emma");

        // Display all elements of the Hashtable
        System.out.println("Student Records:");
        Enumeration<Integer> keys = students.keys();
```

```
        while (keys.hasMoreElements()) {
            Integer key = keys.nextElement();
            System.out.println("Roll No: " + key + ", Name: " + students.get(key));
        }

        // Remove the student with name "John"
        Integer keyToRemove = null;
        for (Map.Entry<Integer, String> entry : students.entrySet()) {
            if (entry.getValue().equals("John")) {
                keyToRemove = entry.getKey();
                break;
            }
        }

        if (keyToRemove != null) {
            students.remove(keyToRemove);
            System.out.println("\nStudent 'John' has been removed.");
        }

        // Display the updated Hashtable
        System.out.println("\nUpdated Student Records:");
        for (Map.Entry<Integer, String> entry : students.entrySet()) {
            System.out.println("Roll No: " + entry.getKey() + ", Name: " +
entry.getValue());
        }
    }
```

}

Assignment 3

SET A

1. **Create a user-defined exception class InvalidNameException. Name should not contain numbers or special symbols.**

```
import java.util.Scanner;

// User-defined exception

class InvalidNameException extends Exception {

    public InvalidNameException(String message) {

        super(message);

    }

}

// Main class

public class a3q1 {

    public static void validateName(String name) throws InvalidNameException {

        if (!name.matches("[a-zA-Z ]+")) { // Only letters and spaces allowed

            throw new InvalidNameException("Invalid Name! Name should not contain numbers or special symbols.");

        }

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

    }



    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter your name: ");
```

```
        String name = sc.nextLine();
        try {
            validateName(name);
        } catch (InvalidNameException e) {
            System.out.println("Error: " + e.getMessage());

        sc.close();
    }
}
```

2. **Create a user-defined exception class InvalidEmailException. A valid email should have domain names .com, .co.in, .org.**

```
import java.util.Scanner;
// User-defined exception
class InvalidEmailException extends Exception {
    public InvalidEmailException(String message) {
        super(message);
    }
}
// Main class
public class a3q2 {
    public static void validateEmail(String email) throws InvalidEmailException {
        if (!email.matches("^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.(com|co\\.in|org)$")) {
            throw new InvalidEmailException("Invalid Email! Email must end with .com, .co.in,
or .org");
        }
        System.out.println("Valid Email: " + email);
```

```
    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter your email: ");

        String email = sc.nextLine();

        try {

            validateEmail(email);

        } catch (InvalidEmailException e) {

            System.out.println("Error: " + e.getMessage());

        }

        sc.close();

    }

}
```

SET B

3. **Create a text file "InFile.txt" containing student names and phone numbers. Read the file using DataInputStream and write its content to "OutFile.txt" using DataOutputStream.**

```
import java.io.*;

public class a3q3 {

    public static void main(String[] args) {

        try {

            // Writing data to "InFile.txt"

            FileWriter fw = new FileWriter("InFile.txt");

            fw.write("om 9876543210\nrohan 9123456789\nmahu 9988776655\n");

            fw.close();

            // Reading from "InFile.txt" using DataInputStream
```

```
            FileInputStream fis = new FileInputStream("InFile.txt");
            DataInputStream dis = new DataInputStream(fis);
            BufferedReader br = new BufferedReader(new InputStreamReader(dis));
            // Writing to "OutFile.txt" using DataOutputStream
            FileOutputStream fos = new FileOutputStream("OutFile.txt");
            DataOutputStream dos = new DataOutputStream(fos);
            String line;
            while ((line = br.readLine()) != null) {
               dos.writeBytes(line + "\n"); // Writing data to "OutFile.txt"
            }
            // Closing streams
            br.close();
            dis.close();
            dos.close();
            fis.close();
            fos.close();
            System.out.println("Data successfully written to OutFile.txt!");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

4. **Create a class Game (`game_name`, `no_of_players`). Create 5 objects and use serialization to store them in a file. Read the file and display all objects.**

```
import java.io.*;
public class a3q2 {
```

```
public static void main(String[] args) {
    try {
        // Writing data to "InFile.txt"
        FileWriter fw = new FileWriter("InFile.txt");
        fw.write("Alice 9876543210\nBob 8765432109\nCharlie 7654321098\n");
        fw.close();
        System.out.println("Data written to InFile.txt");
        // Reading from "InFile.txt" and writing to "OutFile.txt"
        BufferedReader br = new BufferedReader(new FileReader("InFile.txt"));
        BufferedWriter bw = new BufferedWriter(new FileWriter("OutFile.txt"));
        String line;
        while ((line = br.readLine()) != null) {
            bw.write(line + "\n");  // Writing to output file
        }
        br.close();
        bw.close();
        System.out.println("Data successfully copied to OutFile.txt")
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());        }    } }
```

Assignment 5

SET A – Swing Applet

1. **Login form with password validation.**

```
import javax.swing.*;
import java.awt.*;
```

```
import java.awt.event.*;

public class LoginForm {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Login Form");
        frame.setSize(300, 200);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());

        JLabel userLabel = new JLabel("Username:");
        JTextField userField = new JTextField(15);
        JLabel passLabel = new JLabel("Password:");
        JPasswordField passField = new JPasswordField(15);
        JButton loginButton = new JButton("Login");
        JLabel result = new JLabel("");

        loginButton.addActionListener(e -> {
            String username = userField.getText();
            String password = new String(passField.getPassword());

            if (username.equals("admin") && password.equals("12345")) {
                result.setText("Login Successful!");
            } else {
                result.setText("Invalid Login!");
            }
```

```
        });
        frame.add(userLabel);
        frame.add(userField);
        frame.add(passLabel);
        frame.add(passField);
        frame.add(loginButton);
        frame.add(result);
        frame.setVisible(true);
    }
}
```

2. Menu-driven program.

```
import javax.swing.*;
import java.awt.event.*;

public class MenuProgram {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Menu Example");
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JMenuBar menuBar = new JMenuBar();
        JMenu fileMenu = new JMenu("File");
        JMenuItem openItem = new JMenuItem("Open");
        JMenuItem exitItem = new JMenuItem("Exit");
```

```
        exitItem.addActionListener(e -> System.exit(0));

        openItem.addActionListener(e -> JOptionPane.showMessageDialog(frame, "Open
Clicked!"));

        fileMenu.add(openItem);

        fileMenu.add(exitItem);

        menuBar.add(fileMenu);

        frame.setJMenuBar(menuBar);

        frame.setVisible(true);

    }

}
```

3. Unit converter.

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;


public class UnitConverter {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Unit Converter");

        frame.setSize(300, 200);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        frame.setLayout(new FlowLayout());


        JLabel label = new JLabel("Enter Km:");

        JTextField inputField = new JTextField(10);

        JButton convertButton = new JButton("Convert");

        JLabel result = new JLabel("Miles: ");
```

```
        convertButton.addActionListener(e -> {
            double km = Double.parseDouble(inputField.getText());
            double miles = km * 0.621371;
            result.setText("Miles: " + miles);
        });
        frame.add(label);
        frame.add(inputField);
        frame.add(convertButton);
        frame.add(result);
        frame.setVisible(true);
    }
}
```

4. Registration form.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class RegistrationForm {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Registration Form");
        frame.setSize(300, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new FlowLayout());
```

```
JLabel nameLabel = new JLabel("Name:");
JTextField nameField = new JTextField(15);
JLabel emailLabel = new JLabel("Email:");
JTextField emailField = new JTextField(15);
JLabel passLabel = new JLabel("Password:");
JPasswordField passField = new JPasswordField(15);
JButton registerButton = new JButton("Register");
JLabel result = new JLabel("");

registerButton.addActionListener(e -> {
    String name = nameField.getText();
    String email = emailField.getText();
    String password = new String(passField.getPassword());

    if (!name.isEmpty() && email.contains("@")) {
        result.setText("Registration Successful!");
    } else {
        result.setText("Invalid Details!");
    }
});
frame.add(nameLabel);
frame.add(nameField);
frame.add(emailLabel);
frame.add(emailField);
frame.add(passLabel);
```

```
        frame.add(passField);
        frame.add(registerButton);
        frame.add(result);
        frame.setVisible(true);
    }
}
```

SET B – Database Programming

5. **Write a program to display database metadata and list all tables in the database. *(Use `DatabaseMetaData`)**

```
import java.sql.*;
public class DBMetadata {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/your_database"; // Change 'your_database' to your DB name
        String user = "root"; // Change if needed
        String password = "mahesh@123"; // Your MySQL password
        try (Connection conn = DriverManager.getConnection(url, user, password)) {
            DatabaseMetaData metaData = conn.getMetaData();
            // Display database metadata
            System.out.println("Database Name: " + metaData.getDatabaseProductName());
            System.out.println("Database Version: " + metaData.getDatabaseProductVersion());
            System.out.println("Driver Name: " + metaData.getDriverName());
            System.out.println("Driver Version: " + metaData.getDriverVersion());
            // List all tables in the database
            System.out.println("\nTables in the database:");
            ResultSet tables = metaData.getTables(null, null, "%", new String[]{"TABLE"});
```

```
            while (tables.next()) {
                System.out.println(tables.getString("TABLE_NAME"));
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

6. **Write a program to display column details of the DONAR table. *(Use `ResultSetMetaData`)***

```
import java.sql.*;
public class DonarMetadata {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/your_database"; // Change to your database name
        String user = "root"; // Change if needed
        String password = "mahesh@123"; // Your MySQL password
        String query = "SELECT * FROM DONAR"; // Query to fetch data from DONAR table

        try (Connection conn = DriverManager.getConnection(url, user, password);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {

            ResultSetMetaData metaData = rs.getMetaData();
            int columnCount = metaData.getColumnCount();
```

```
            System.out.println("Column Details of DONAR Table:");
            System.out.println("---------------------------------");
            for (int i = 1; i <= columnCount; i++) {
                System.out.println("Column " + i + ": " + metaData.getColumnName(i));
                System.out.println("  - Type: " + metaData.getColumnTypeName(i));
                System.out.println("  - Size: " + metaData.getColumnDisplaySize(i));
                System.out.println("---------------------------------");
            }
        } catch (SQLException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

7. **Create a BOOK table (`book_id`, `book_name`, `book_publication`, `price`). Insert values and display details in tabular format using Swing.**

```
import javax.swing.*;
import java.awt.*;
import java.sql.*;

public class BookTableSimple {
    public static void main(String[] args) {
        new BookGUI();
    }
}

class BookGUI extends JFrame {
```

```
Connection conn;
String url = "jdbc:mysql://localhost:3306/your_database"; // Change to your database
String user = "root"; // Change if needed
String password = "mahesh@123"; // Your MySQL password

BookGUI() {
    setTitle("Book Table");
    setSize(400, 300);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new FlowLayout());

    JButton btnInsert = new JButton("Insert Book");
    JButton btnShow = new JButton("Show Books");
    add(btnInsert);
    add(btnShow);

    btnInsert.addActionListener(e -> insertBook());
    btnShow.addActionListener(e -> showBooks());

    createTable();
    setVisible(true);
}

void createTable() {
    try {
```

```
            conn = DriverManager.getConnection(url, user, password);

            Statement stmt = conn.createStatement();

            stmt.executeUpdate("CREATE TABLE IF NOT EXISTS BOOK (id INT
AUTO_INCREMENT PRIMARY KEY, name VARCHAR(100), price DOUBLE)");

        } catch (SQLException e) {

            JOptionPane.showMessageDialog(this, "DB Error: " + e.getMessage());

        }

    }



    void insertBook() {

        String name = JOptionPane.showInputDialog("Enter Book Name:");

        String price = JOptionPane.showInputDialog("Enter Price:");



        try (PreparedStatement pst = conn.prepareStatement("INSERT INTO BOOK (name,
price) VALUES (?, ?)")) {

            pst.setString(1, name);

            pst.setDouble(2, Double.parseDouble(price));

            pst.executeUpdate();

            JOptionPane.showMessageDialog(this, "Book Inserted!");

        } catch (SQLException e) {

            JOptionPane.showMessageDialog(this, "Insert Error: " + e.getMessage());

        }

    }

    void showBooks() {

        try (Statement stmt = conn.createStatement();

             ResultSet rs = stmt.executeQuery("SELECT * FROM BOOK")) {
```

```
            StringBuilder data = new StringBuilder("Book Details:\n");

            while (rs.next()) {

                data.append("ID: ").append(rs.getInt("id")).append(", Name:
").append(rs.getString("name"))

                    .append(", Price: ").append(rs.getDouble("price")).append("\n");

            }

            JOptionPane.showMessageDialog(this, data.toString());

        } catch (SQLException e) {

            JOptionPane.showMessageDialog(this, "Fetch Error: " + e.getMessage());

        }

    }

}
```

Assignment 6

SET A – Servlets

1. **Design a servlet that provides client request details (IP address, browser type) and server details (OS type, loaded servlets).**

```
import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.PrintWriter;

import java.util.Enumeration;
```

```
@WebServlet("/requestDetails")

public class RequestDetailsServlet extends HttpServlet {

   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

       response.setContentType("text/html");

       PrintWriter out = response.getWriter();


       // Client details

       String clientIP = request.getRemoteAddr();

       String browser = request.getHeader("User-Agent");


       // Server details

       String serverOS = System.getProperty("os.name");


       out.println("<html><body>");

       out.println("<h2>Client Request Details</h2>");

       out.println("<p><b>Client IP:</b> " + clientIP + "</p>");

       out.println("<p><b>Browser:</b> " + browser + "</p>");


       out.println("<h2>Server Details</h2>");

       out.println("<p><b>OS:</b> " + serverOS + "</p>");

       out.println("</body></html>");

   }}
```

Web.xml

```
<servlet>

   <servlet-name>RequestDetailsServlet</servlet-name>
```

```
    <servlet-class>com.example.RequestDetailsServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>RequestDetailsServlet</servlet-name>
    <url-pattern>/requestDetails</url-pattern>
</servlet-mapping>
```

2. **Create a servlet-based login page. Display a welcome message if the username and password match.**

Index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Login Page</title>
</head>
<body>
    <h2>Login</h2>
    <form action="LoginServlet" method="post">
        Username: <input type="text" name="username"><br><br>
        Password: <input type="password" name="password"><br><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```

LoginServlet.java (• Inside `src/main/java`, create a **package** (e.g., `com.example`).

- Inside this package, create a new **Java class** (`LoginServlet.java`).**)**

```
import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.PrintWriter;


@WebServlet("/LoginServlet")

public class LoginServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();


        // Get user input

        String username = request.getParameter("username");

        String password = request.getParameter("password");


        // Hardcoded username & password

        if ("admin".equals(username) && "1234".equals(password)) {

            out.println("<h2>Welcome, " + username + "!</h2>");

        } else {

            out.println("<h2>Invalid username or password!</h2>");

            out.println("<a href='index.html'>Try Again</a>");

        }
```

```
    }
}
```

3. **Write a servlet to display Good Morning/Afternoon/Evening based on the current time.**

```
import java.io.IOException;
import java.io.PrintWriter;
import java.time.LocalTime;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/GreetingServlet")
public class GreetingServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        // Get current time
        LocalTime currentTime = LocalTime.now();
        int hour = currentTime.getHour();

        // Determine greeting based on time
        String greeting;
```

```
        if (hour >= 5 && hour < 12) {
            greeting = "Good Morning!";
        } else if (hour >= 12 && hour < 17) {
            greeting = "Good Afternoon!";
        } else {
            greeting = "Good Evening!";
        }

        // Display greeting
        out.println("<h2>" + greeting + "</h2>");
        out.println("<p>Current Time: " + currentTime + "</p>");
    }
}
```

SET B – JSP

4. **Write a JSP program to display Company details (`Name`, `Location`, `Address`, `Email-id`, `Phone_no`) in tabular format.**

create a Dynamic Web Project

1. **File → New → Dynamic Web Project**
2. Name it **"CompanyDetailsJSP"**
3. Select **Apache Tomcat** as the runtime
4. Click **Finish**

Create `company.jsp` Inside `WebContent` Folder

1. Inside `WebContent`, create a new file **`company.jsp`**
2. Paste the following **JSP code**:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>
```

```
    <title>Company Details</title>
    <style>
        table {
            width: 60%;
            border-collapse: collapse;
            margin: 20px auto;
            font-family: Arial, sans-serif;
        }
        th, td {
            border: 1px solid black;
            padding: 10px;
            text-align: center;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <h2 align="center">Company Details</h2>
    <table>
        <tr>
            <th>Name</th>
            <th>Location</th>
            <th>Address</th>
```

```
    <th>Email</th>
    <th>Phone No</th>
  </tr>
  <tr>
    <td>Tata Consultancy Services</td>
    <td>Mumbai</td>
    <td>123, IT Park, Mumbai</td>
    <td>contact@tcs.com</td>
    <td>9876543210</td>
  </tr>
  <tr>
    <td>Infosys</td>
    <td>Bangalore</td>
    <td>456, Tech Hub, Bangalore</td>
    <td>support@infosys.com</td>
    <td>8765432109</td>
  </tr>
  <tr>
    <td>Wipro</td>
    <td>Pune</td>
    <td>789, Innovation Tower, Pune</td>
    <td>help@wipro.com</td>
    <td>7654321098</td>
  </tr>
</table>
```

```
</body>

</html>
```

5. **Write a JSP program to display Player details (`Name`, `Game`, `No_Matches_played`, `Email-id`) in tabular format.**

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

   <title>Player Details</title>

   <style>

      table {

         width: 60%;

         border-collapse: collapse;

         margin: 20px auto;

         font-family: Arial, sans-serif;

      }

      th, td {

         border: 1px solid black;

         padding: 10px;

         text-align: center;

      }

      th {

         background-color: #f2f2f2;

      }
```

```
    </style>
</head>
<body>
    <h2 align="center">Player Details</h2>
    <table>
        <tr>
            <th>Name</th>
            <th>Game</th>
            <th>No. of Matches Played</th>
            <th>Email ID</th>
        </tr>
        <tr>
            <td>Virat Kohli</td>
            <td>Cricket</td>
            <td>254</td>
            <td>virat@cricket.com</td>
        </tr>
        <tr>
            <td>Messi</td>
            <td>Football</td>
            <td>980</td>
            <td>messi@football.com</td>
        </tr>
        <tr>
            <td>PV Sindhu</td>
```

```
            <td>Badminton</td>
            <td>120</td>
            <td>sindhu@badminton.com</td>
        </tr>
    </table>
</body>
</html>
```