Hrishikesh Sarma

RA2311003010302

WEEK 9 ASSIGNMENT:

Q1.

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

public class SimpleLibrarySystem extends JFrame {
    private ArrayList<Book> books = new ArrayList<>();
    private DefaultTableModel tableModel;

    public SimpleLibrarySystem() {
        setTitle("Simple Library System");
        setSize(400, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Table for displaying books
        tableModel = new DefaultTableModel(new String[]{"Title", "Author"}, 0);
        JTable bookTable = new JTable(tableModel);
        add(new JScrollPane(bookTable), BorderLayout.CENTER);
```

```
        // Panel for adding books
        JPanel panel = new JPanel();
        JTextField titleField = new JTextField(10);
        JTextField authorField = new JTextField(10);
        JButton addButton = new JButton("Add Book");

        panel.add(new JLabel("Title:"));
        panel.add(titleField);
        panel.add(new JLabel("Author:"));
        panel.add(authorField);
        panel.add(addButton);

        add(panel, BorderLayout.SOUTH);

        // Action listener for adding books
        addButton.addActionListener(e -> {
            String title = titleField.getText();
            String author = authorField.getText();
            if (!title.isEmpty() && !author.isEmpty()) {
                books.add(new Book(title, author));
                tableModel.addRow(new Object[]{title, author});
                titleField.setText("");
                authorField.setText("");
            } else {
                JOptionPane.showMessageDialog(this, "Please enter both title and author.", "Error",
JOptionPane.ERROR_MESSAGE);
            }
        });
```

```
        setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(SimpleLibrarySystem::new);
    }
}

class Book {
    private String title;
    private String author;

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

    public String getTitle() { return title; }
    public String getAuthor() { return author; }
}
```

Q2:

```
import java.util.Scanner;

public class CircleArea {

    // Pure function to calculate the area of a circle
```

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

    // Impure function to prompt user for radius and print the area
    public static void promptUserAndPrintArea() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the radius of the circle: ");
        double radius = scanner.nextDouble();  // User input
        double area = calculateArea(radius);    // Calling the pure function
        System.out.printf("The area of the circle with radius %.2f is: %.2f%n", radius, area);
    }

    public static void main(String[] args) {
        // Call the impure function to execute the program
        promptUserAndPrintArea();
    }
}
```

OUTPUT:

```
Enter the radius of the circle: 5
The area of the circle with radius 5.00 is: 78.54
```

Q3.

```
import java.util.Arrays;

public class NumberList {
```

```
    // Pure function to find the maximum value in a list of numbers
    public static int findMax(int[] numbers) {
        int max = numbers[0];
        for (int num : numbers) {
            if (num > max) {
                max = num;
            }
        }
        return max;
    }

    // Impure function to sort a list of numbers in ascending order
    public static void sortNumbers(int[] numbers) {
        Arrays.sort(numbers); // This modifies the original array
    }

    public static void main(String[] args) {
        int[] numbers = {34, 12, 5, 67, 23};

        // Using the pure function
        int max = findMax(numbers);
        System.out.println("Maximum value: " + max);

        // Using the impure function
        sortNumbers(numbers);
        System.out.println("Sorted numbers: " + Arrays.toString(numbers));
    }
}
```

OUTPUT:

```
Maximum value: 67
Sorted numbers: [5, 12, 23, 34, 67]
```

Q4.

```
import java.util.ArrayList;
import java.util.List;

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

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

    public int getAge() {
        return age;
    }

    // Additional getters can be added if needed
}

public class StudentRecords {
```

```
private List<Student> students;

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

// Mutable function to add a new student to the list
public void addStudent(String name, int age, double grade) {
    Student newStudent = new Student(name, age, grade);
    students.add(newStudent);
}

// Immutable function to calculate the average age of students
public double calculateAverageAge() {
    if (students.isEmpty()) {
        return 0; // Avoid division by zero
    }
    int totalAge = 0;
    for (Student student : students) {
        totalAge += student.getAge();
    }
    return (double) totalAge / students.size();
}

public static void main(String[] args) {
    StudentRecords records = new StudentRecords();

    // Adding students
    records.addStudent("Alice", 20, 85.5);
```

```
        records.addStudent("Bob", 22, 90.0);
        records.addStudent("Charlie", 19, 78.5);

        // Calculating average age
        double averageAge = records.calculateAverageAge();
        System.out.printf("Average age of students: %.2f%n", averageAge);
    }
}
```

OUTPUT:

```
Average age of students: 20.33
```

Q5.

```
class BankAccount {
    private String accountHolder;
    private double balance;
    private double interestRate; // Annual interest rate as a percentage

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

    // Mutable function to update the balance of the bank account
    public void updateBalance(double amount) {
        this.balance += amount; // Add or subtract amount to the balance
    }
```

```
// Immutable function to calculate the interest earned on the account
public double calculateInterest(int years) {
    return (this.balance * this.interestRate / 100) * years; // Simple interest calculation
}

// Getter for balance
public double getBalance() {
    return balance;
}

public static void main(String[] args) {
    // Create a new bank account
    BankAccount account = new BankAccount("John Doe", 1000.0, 5.0);

    // Display initial balance
    System.out.printf("Initial balance: $%.2f%n", account.getBalance());

    // Update balance by depositing $500
    account.updateBalance(500);
    System.out.printf("Balance after deposit: $%.2f%n", account.getBalance());

    // Update balance by withdrawing $200
    account.updateBalance(-200);
    System.out.printf("Balance after withdrawal: $%.2f%n", account.getBalance());

    // Calculate interest earned over 3 years
    double interestEarned = account.calculateInterest(3);
    System.out.printf("Interest earned over 3 years: $%.2f%n", interestEarned);
```

```
    }
}
```

OUTPUT:

```
Initial balance: $1000.00
Balance after deposit: $1500.00
Balance after withdrawal: $1300.00
Interest earned over 3 years: $195.00
```

Q6.

```
import java.util.function.BiFunction;


public class AnonymousFunctionExample {
    public static void main(String[] args) {
        // Anonymous function to add two numbers and immediately call it
        int sum = ((a, b) -> a + b).apply(5, 3);
        System.out.println("Sum of 5 and 3: " + sum);


        // Anonymous function to multiply two numbers and assign it to a variable
        BiFunction<Integer, Integer, Integer> multiply = (a, b) -> a * b;
        int product = multiply.apply(4, 6);
        System.out.println("Product of 4 and 6: " + product);
    }
}
```

OUTPUT:

```
Sum of 5 and 3: 8
Product of 4 and 6: 24
```

Q7.

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

public class AnonymousFunctionExample {
    public static void main(String[] args) {
        // List of numbers
        List<Integer> numbersToSquare = Arrays.asList(1, 2, 3, 4);

        // Anonymous function to square each element in the list
        List<Integer> squaredNumbers = numbersToSquare.stream()
            .map(num -> num * num) // Squaring each number
            .collect(Collectors.toList());

        System.out.println("Squared Numbers: " + squaredNumbers);

        // List of numbers to filter
        List<Integer> numbersToFilter = Arrays.asList(1, 2, 3, 4, 5, 6);

        // Anonymous function to filter even numbers from the list
        List<Integer> evenNumbers = numbersToFilter.stream()
            .filter(num -> num % 2 == 0) // Filtering even numbers
            .collect(Collectors.toList());

        System.out.println("Even Numbers: " + evenNumbers);
    }
```

}

OUTPUT:

```
Squared Numbers: [1, 4, 9, 16]
Even Numbers: [2, 4, 6]
```

Q8.

```
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

class Employee {
    private String name;
    private double salary;
    private String department;

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

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
```

```
    }

    public String getDepartment() {
        return department;
    }
}

public class EmployeeExample {
    public static void main(String[] args) {
        // Sample list of employees
        List<Employee> employees = Arrays.asList(
            new Employee("Alice", 70000, "Engineering"),
            new Employee("Bob", 60000, "Engineering"),
            new Employee("Charlie", 80000, "Marketing"),
            new Employee("David", 50000, "Marketing"),
            new Employee("Eve", 90000, "HR")
        );

        // Map: Create a new list containing the employee names and their corresponding salaries
        List<String> employeeNamesWithSalaries = employees.stream()
            .map(e -> e.getName() + " - $" + e.getSalary())
            .collect(Collectors.toList());

        System.out.println("Employee Names and Salaries:");
        employeeNamesWithSalaries.forEach(System.out::println);

        // Filter: Filter employees who belong to a specific department
        String targetDepartment = "Engineering";
        List<Employee> filteredEmployees = employees.stream()
```

```
            .filter(e -> e.getDepartment().equals(targetDepartment))
            .collect(Collectors.toList());

        System.out.println("\nEmployees in " + targetDepartment + " Department:");
        filteredEmployees.forEach(e -> System.out.println(e.getName()));

        // Reduce: Calculate the average salary for employees in each department
        Map<String, Double> averageSalaryByDepartment = employees.stream()
            .collect(Collectors.groupingBy(
                Employee::getDepartment,
                Collectors.averagingDouble(Employee::getSalary)
            ));

        System.out.println("\nAverage Salary by Department:");
        averageSalaryByDepartment.forEach((department, avgSalary) ->
            System.out.println(department + ": $" + avgSalary));
    }
}
```

OUTPUT:

```
Employee Names and Salaries:
Alice - $70000.0
Bob - $60000.0
Charlie - $80000.0
David - $50000.0
Eve - $90000.0

Employees in Engineering Department:
Alice
Bob

Average Salary by Department:
Engineering: $65000.0
Marketing: $65000.0
HR: $90000.0
```

Q9.

```
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

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

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

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

    public int getAge() {
        return age;
    }

    public double getGrade() {
        return grade;
    }
}

public class StudentExample {
    public static void main(String[] args) {
        // Sample list of students
        List<Student> students = Arrays.asList(
            new Student("Alice", 20, 85.5),
            new Student("Bob", 22, 90.0),
            new Student("Charlie", 19, 78.0),
            new Student("David", 21, 88.5),
            new Student("Eve", 23, 92.0)
        );

        // Map: Create a new list containing only the names of the students
        List<String> studentNames = students.stream()
            .map(Student::getName)
            .collect(Collectors.toList());
```

```
        System.out.println("Student Names:");
        studentNames.forEach(System.out::println);

        // Filter: Filter the students who are above a certain age threshold
        int ageThreshold = 21;
        List<Student> filteredStudents = students.stream()
            .filter(s -> s.getAge() > ageThreshold)
            .collect(Collectors.toList());

        System.out.println("\nStudents above age " + ageThreshold + ":");
        filteredStudents.forEach(s -> System.out.println(s.getName()));

        // Reduce: Calculate the average grade of all students
        double averageGrade = students.stream()
            .collect(Collectors.averagingDouble(Student::getGrade));

        System.out.printf("\nAverage Grade of all students: %.2f%n", averageGrade);
    }
}
```

OUTPUT:

```
Student Names:
Alice
Bob
Charlie
David
Eve

Students above age 21:
Bob
Eve

Average Grade of all students: 86.80
```