Experiment 5

Student Name: Riya
UID: 22BCS12529
Branch: CSE
Section/Group: 619-A
Semester: 6th
Date of Performance: 25/02/25
Subject Name: PBLJ-LAB
Subject Code: 22CSH-359

1. Aim:

A: Autoboxing, Unboxing, and Sum Calculation
Objective: To understand the concepts of autoboxing and unboxing in Java and how they can be utilized to simplify the manipulation of primitive data types with wrapper classes.
B: Serialization and Deserialization of a Student Object
Objective: To understand the concepts of serialization and deserialization in Java and how they are used to store and retrieve objects.
C: Menu-Based Application for Employee Management
Objective: To create a menu-based Java application that allows the addition and display of employee details, and stores this information in a file for persistence.

2. Implementation/Code:

(A)

```
import java.util.*;

public class AutoboxingUnboxing {
    public static void main(String[] args) {
        String[] numberStrings = {"10", "20", "30", "40"};
        List<Integer> integerList = new ArrayList<>();

        for (String number : numberStrings) {
            integerList.add(Integer.parseInt(number));
        }
```

```
        int sum = 0;
        for (Integer num : integerList) {
            sum += num;
        }

        System.out.println("Sum of integers: " + sum);
    }
}
```

(B)

```
import java.io.*;

class Student implements Serializable {
    private String name;
    private int age;
    private String course;

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

    @Override
    public String toString() {
        return "Student{name='" + name + "', age=" + age + ", course='" + course
+ "'}";
    }
}

public class StudentSerialization {
    public static void main(String[] args) {
        Student student = new Student("John Doe", 22, "Computer Science");

        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream("student.ser"))) {
            out.writeObject(student);
            System.out.println("Student serialized successfully!");
        } catch (IOException e) {
            e.printStackTrace();
```

```
        }

        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream("student.ser"))) {
            Student deserializedStudent = (Student) in.readObject();
            System.out.println("Deserialized Student: " + deserializedStudent);
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

(C)

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

class Employee implements Serializable {
    private int id;
    private String name;
    private String designation;
    private double salary;

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

    @Override
    public String toString() {
        return "Employee ID: " + id + "\nName: " + name + "\nDesignation: " +
designation + "\nSalary: " + salary;
    }
}

public class EmployeeManagement {
    private static final String FILE_NAME = "employees.ser";
```

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

        while (true) {
            System.out.println("\nMenu:");
            System.out.println("1. Add Employee");
            System.out.println("2. Display All Employees");
            System.out.println("3. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    addEmployee(scanner);
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    System.out.println("Exiting program.");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid option. Try again.");
            }
        }
    }

    private static void addEmployee(Scanner scanner) {
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Enter Name: ");
        String name = scanner.nextLine();

        System.out.print("Enter Designation: ");
        String designation = scanner.nextLine();

        System.out.print("Enter Salary: ");
```

```
        double salary = scanner.nextDouble();

        Employee employee = new Employee(id, name, designation, salary);
        saveEmployee(employee);
        System.out.println("Employee added successfully!");
    }

    private static void saveEmployee(Employee employee) {
        List<Employee> employees = loadEmployees();
        employees.add(employee);

        try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            out.writeObject(employees);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static List<Employee> loadEmployees() {
        File file = new File(FILE_NAME);
        if (!file.exists()) {
            return new ArrayList<>();
        }

        try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            return (List<Employee>) in.readObject();
        } catch (IOException | ClassNotFoundException e) {
            return new ArrayList<>();
        }
    }

    private static void displayEmployees() {
        List<Employee> employees = loadEmployees();
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
            return;
        }
        for (Employee emp : employees) {
            System.out.println(emp);
            System.out.println("----------------------");
```

```
        }
    }
}
```

2. Output

A

```
Sum of integers: 100
```

B

```
Student serialized successfully!
Deserialized Student: Student{name='John Doe', age=22, course='Computer Science'}
```

C

```
Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 1
Enter Employee ID: 101
Enter Name: Jhon Doe
Enter Designation: SE
Enter Salary: 100
Employee added successfully!

Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 2
Employee ID: 123
Name: Jhon Doe
Designation: SE
Salary: 100.0
----------------------
Employee ID: 101
Name: Jhon Doe
Designation: SE
Salary: 100.0
----------------------
```

3. Learning Outcome:

1. Understanding Autoboxing and Unboxing in Java
2. Gain hands-on experience in converting Java objects into a **byte stream** (serialization) and storing them in a file.
3. Learn how to create a **menu-driven Java application** using Scanner for user input.