EXPERINMENT-5

1.AIM(a): Java Programming Concepts Part 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.

2.CODE:

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


public class Main {
    private static final String EMPLOYEE_FILE = "employees.dat";


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


        // Part A: Autoboxing & Unboxing (Sum Calculation)
        String[] numberStrings = {"10", "20", "30", "40"};
        int sum = calculateSum(numberStrings);
        System.out.println("Sum of integers: " + sum);


        // Part B: Serialization & Deserialization of Student Object
        Student student = new Student("Tamanna Singh", 20, "Computer Science");
        serializeStudent(student, "student.ser");
        Student deserializedStudent = deserializeStudent("student.ser");
        System.out.println("\nDeserialized Student:");
        deserializedStudent.display();
```

```
// Part C: Employee Management System
while (true) {
    System.out.println("\nEmployee Management System");
    System.out.println("1. Add Employee");
    System.out.println("2. Display All Employees");
    System.out.println("3. Exit");
    System.out.print("Enter your choice: ");

    int choice = scanner.nextInt();
    scanner.nextLine(); // Consume newline

    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 choice. Please try again.");
    }
}
```

```
}

// 🚀 Part A: Autoboxing & Unboxing (Sum Calculation)
public static int calculateSum(String[] numberStrings) {
    List<Integer> integerList = new ArrayList<>();
    for (String number : numberStrings) {
        integerList.add(Integer.parseInt(number)); // Autoboxing
    }
    int sum = 0;
    for (Integer num : integerList) {
        sum += num; // Unboxing
    }
    return sum;
}

// 🚀 Part B: Serialization & Deserialization (Student)
static class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    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;
    }
```

```
    public void display() {
        System.out.println("Student{name='" + name + "', age=" + age + ", course='" + course + "'}");
    }
}

public static void serializeStudent(Student student, String filename) {
    try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename))) {
        out.writeObject(student);
        System.out.println("Student object serialized successfully!");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static Student deserializeStudent(String filename) {
    try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {
        return (Student) in.readObject();
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
        return null;
    }
}

// 🚀 Part C: Employee Management System
```

```
    static class Employee implements Serializable {
        private static final long serialVersionUID = 1L;
        private String name;
        private int employeeId;
        private String designation;
        private double salary;

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

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

    public static void addEmployee(Scanner scanner) {
        System.out.print("Enter Name: ");
        String name = scanner.nextLine();
```

```
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Enter Designation: ");
        String designation = scanner.nextLine();
        System.out.print("Enter Salary: ");
        double salary = scanner.nextDouble();
        scanner.nextLine(); // Consume newline

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

    public static void displayEmployees() {
        List<Employee> employees = readEmployeesFromFile();
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            System.out.println("\n--- Employee List ---");
            for (Employee emp : employees) {
                System.out.println(emp);
            }
        }
    }

    private static void saveEmployeeToFile(Employee employee) {
```

```
        List<Employee> employees = readEmployeesFromFile();
        employees.add(employee);

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

    private static List<Employee> readEmployeesFromFile() {
        List<Employee> employees = new ArrayList<>();
        File file = new File(EMPLOYEE_FILE);

        if (file.exists()) {
            try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(EMPLOYEE_FILE))) {
                employees = (List<Employee>) in.readObject();
            } catch (IOException | ClassNotFoundException e) {
                e.printStackTrace();
            }
        }
        return employees;
    }
}
```

3.output:

```
Main [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.10.v20240120-1143\jre\bin\javaw.exe  (27-Feb-2025, 4:16:36 pm) [pid: 18244]
Sum of integers: 100
Student object serialized successfully!

Deserialized Student:
Student{name='Tamanna Singh', age=20, course='Computer Science'}

Employee Management System
1. Add Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Name: Tamanna Singh
Enter Employee ID: 12
Enter Designation: cse
Enter Salary: 100000
Employee added successfully!
```