Experiment: 5

Student Name: AKASH DEEP **UID:22BCS10195**
Branch: CSE **Section/Group: DL-902**
Semester: 6th **Date of Performance:10/02/25**
Subject Name: Project Based Learning In Java with Lab **Subject Code: 22CSH-359**

1. Aim: Write a Java program to calculate the sum of a list of integers using autoboxing and unboxing. Include methods to parse strings into their respective wrapper classes (e.g., Integer.parseInt()).

2. Objective: Demonstrate autoboxing and unboxing in Java by computing the sum of a list of integers while parsing string inputs into their respective wrapper classes.

3. Code:

```
import java.util.*;
public class SumCal{
   public static void main(String[] args) {
      String[] numStr = {"11","24","54","43","69"};
      List<Integer> no = new ArrayList<>();
      int s=0;
      for (String str : numStr) {
         no.add(Integer.parseInt(str));    // Autoboxing
      }
      for (int num : no) {
         s+= num;      // Unboxing
      }
      System.out.println("Total Sum:"+s);
   }
}
```

4. Output:

1. **Aim: Create a Java program to serialize and deserialize a Student object.**

2. **Objective:** To implement serialization and deserialization of a Student object, ensuring data persistence and retrieval using Java's ObjectOutputStream and ObjectInputStream.

3. Code:

```
import java.io.*;


class Student implements Serializable {
    private int id;
    private String name;
    private double gpa;
    Student(int id, String name, double gpa) {
        this.id = id;
        this.name = name;
        this.gpa = gpa;
    }
    void get() {
        System.out.println("Student ID:" + id);
        System.out.println("Name:" + name);
        System.out.println("GPA:" + gpa);
    }
}
```

```
public class StudentData {
    public static void main(String[] args) {
        String f = "student_data.ser";
        Student stud = new Student(10195, "Akash", 8.4);
        // Serialization
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(f))) {
            out.writeObject(stud);
            System.out.println("student data saved...");
        } catch (IOException e) {
            System.err.println("Error:" + e.getMessage());
        }
        // Deserialization
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(f))) {
            Student load_Stud = (Student) in.readObject();
            System.out.println("Deserialized student:");
            load_Stud.get();
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Deserialization Error: " + e.getMessage());
        }
    }
}
```

4. Output:

```
akashdeep@Akashs-MacBook-
student data saved...
Deserialized student:
Student ID:10195
Name:Akash
GPA:8.4
```

1. **Aim:** Create a menu-based Java application with the following options. 1.Add an Employee 2. Display All 3. Exit If option 1 is selected, the application should gather details of the employee like employee name, employee id, designation and salary and store it in a file. If option 2 is selected, the application should display all the employee details. If option 3 is selected the application should exit.

2. **Objective:** To develop a menu-driven Java application that allows adding, storing, and displaying employee details using file handling for data persistence.

3. Code:

```
import java.io.*;

import java.util.*;


class Employee implements Serializable {

    private int id;

    private String name;

    private String role;

    private double salary;

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

        this.id = id;

        this.name = name;

        this.role = role;

        this.salary = salary;

    }

    void show() {

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

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

        System.out.println("Role:"+ role);
```

```
        System.out.println("Salary:"+ salary);
        System.out.println("-----------------------");
    }
}

public class Emp{
    private static final String path_file = "employees_data.ser";
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Employee> empList = loadEmployees();
        while (true) {
            System.out.println("\n1. Add Employee");
            System.out.println("2. Show Employees");
            System.out.println("3. Exit");
            System.out.print("Select option: ");
            int ch = scanner.nextInt();
            scanner.nextLine();
            switch (ch) {
                case 1:
                    System.out.print("Enter ID: ");
                    int id = scanner.nextInt();
                    scanner.nextLine();
                    System.out.print("Enter Name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter Role: ");
```

```
                String role = scanner.nextLine();
                System.out.print("Enter Salary: ");
                double salary = scanner.nextDouble();
                empList.add(new Employee(id, name, role, salary));
                saveEmployees(empList);
                System.out.println("Employee added successfully.");
                break;
            case 2:
                System.out.println("\nEmployee Details:");
                for (Employee emp : empList) {
                    emp.show();
                }
                break;
            case 3:
                System.out.println("Exiting...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid ,Try again....byee");
        }
    }
}
private static void saveEmployees(List<Employee> empList) {
    try (ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(path_file))) {
```

```
            out.writeObject(empList);
        } catch (IOException e) {
            System.err.println("Error saving data: " + e.getMessage());
        }
    }
    private static List<Employee> loadEmployees() {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(path_file))) {
            return (List<Employee>) in.readObject();
        } catch (IOException | ClassNotFoundException e) {
            return new ArrayList<>();
        }
    }
}
```

4. Output:

```
1. Add Employee
2. Show Employees
3. Exit
Select option: 1
Enter ID: 10195
Enter Name: Akash
Enter Role: manager
Enter Salary: 100000
Employee added successfully.

1. Add Employee
2. Show Employees
3. Exit
Select option: 2

Employee Details:
Employee ID:10195
Name:Akash
Role:manager
Salary:100000.0
----------------------

1. Add Employee
2. Show Employees
3. Exit
Select option: 3
Exiting...
```

6. Learning Outcome :

- Understand and implement autoboxing and unboxing for efficient handling of primitive and wrapper class conversions.
- Use Java wrapper classes and parsing methods (Integer.parseInt(), etc.) for data conversion.
- Apply serialization and deserialization techniques to store and retrieve objects using file streams.
- Implement file handling to store and manage employee records persistently.
- Develop a menu-driven application to perform CRUD operations using user input and file storage.