DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Experiment 5 .1

Student Name: Reshma Saluja **UID: 22BCS50001**
Branch: CSE **Section/Group: 643/B**
Semester: 6th **Date of Performance: 24/02/25**
Subject Name: PBLJ **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. **Code:**

```
import java.util.ArrayList;

public class IntegerSum {
    public static void main(String[] args) {
        String[] numberStrings = {"10", "20", "30", "40", "50"};

        ArrayList<Integer> numbers = new ArrayList<>();
        for (String s : numberStrings) {
            // Parsing string to Integer (autoboxing happens when added to list)
            numbers.add(Integer.parseInt(s));
        }

        int sum = 0;
        for (Integer number : numbers) {
            // Unboxing (Integer to int) happens automatically
            sum += number;
        }

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

3. **Output:**

```
Sum of integers: 150


...Program finished with exit code 0
Press ENTER to exit console.
```

DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Experiment 5 .2

Student Name: Reshma Saluja **UID:22BCS50001**
Branch: CSE **Section/Group: 643/B**
Semester: 6th **Date of Performance: 24/02/25**
Subject Name: PBLJ **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()). Medium Level: Create a Java program to serialize and deserialize a Student object. The program should: Serialize a Student object (containing id, name, and GPA) and save it to a file. Deserialize the object from the file and display the student details. Handle FileNotFoundException, IOException, and ClassNotFoundException using exception handling.

2. **Code:-**

```
import java.io.*;

// Serializable Student class
class Student implements Serializable {
    private int id;
    private String name;
    private double gpa;

    // Constructor
    public Student(int id, String name, double gpa) {
        this.id = id;
        this.name = name;
        this.gpa = gpa;
    }

    // Display method
    public void display() {
        System.out.println("Student Details:");
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("GPA: " + gpa);
    }
}

public class StudentSerializationApp {

    private static final String FILE_NAME = "student_data.ser";

    public static void main(String[] args) {
        // Create a Student object
        Student student = new Student(101, "Alice Johnson", 3.9);
```

```
    // Serialize the Student object
    serializeStudent(student);

    // Deserialize the Student object
    Student deserializedStudent = deserializeStudent();

    // Display deserialized student details if successfully read
    if (deserializedStudent != null) {
        deserializedStudent.display();
    } else {
        System.out.println("Failed to read student data.");
    }
}

// Method to serialize a student object
private static void serializeStudent(Student student) {
    try (ObjectOutputStream oos = new ObjectOutputStream(new
  FileOutputStream(FILE_NAME))) {
        oos.writeObject(student);
        System.out.println("Student object serialized successfully.");
    } catch (FileNotFoundException e) {
        System.err.println("File not found during serialization: " + e.getMessage());
    } catch (IOException e) {
        System.err.println("IOException during serialization: " + e.getMessage());
    }
}

// Method to deserialize a student object
private static Student deserializeStudent() {
    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME)))
  {
        return (Student) ois.readObject();
    } catch (FileNotFoundException e) {
        System.err.println("File not found during deserialization: " + e.getMessage());
    } catch (IOException e) {
        System.err.println("IOException during deserialization: " + e.getMessage());
    } catch (ClassNotFoundException e) {
        System.err.println("Class not found during deserialization: " + e.getMessage());
    }
    return null;
}
  }
```

3. **Output:-**

```
Student object serialized successfully.
Student Details:
ID: 101
Name: Alice Johnson
GPA: 3.9


...Program finished with exit code 0
Press ENTER to exit console.
```

Experiment 5.3

Student Name: Reshma Saluja **UID:22BCS50001**
Branch: CSE **Section/Group: 643/B**
Semester: 6th **Date of Performance:24/02/25**
Subject Name: PBLJ **Subject Code: 22CSH-359**

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. Code:

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

// Employee class implementing Serializable
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 "ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " + salary;
    }
}

public class EmployeeManagementApp {
    private static final String FILE_NAME = "employees.dat";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Employee> employees = loadEmployees();

        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("Select an option: ");

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

        switch (choice) {
            case 1:
                addEmployee(employees, scanner);
                saveEmployees(employees);
                break;
            case 2:
                displayAllEmployees(employees);
                break;
            case 3:
                saveEmployees(employees);
                System.out.println("Exiting...");
                return;
            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
}

private static void addEmployee(List<Employee> employees, Scanner scanner) {
    System.out.print("Enter Employee ID: ");
    int id = scanner.nextInt();
    scanner.nextLine();  // Consume newline

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

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

    System.out.print("Enter Salary: ");
    double salary = scanner.nextDouble();

    employees.add(new Employee(id, name, designation, salary));
    System.out.println("Employee added successfully.");
}

private static void displayAllEmployees(List<Employee> employees) {
    if (employees.isEmpty()) {
        System.out.println("No employees found.");
    } else {
        employees.forEach(System.out::println);
```

```
        }
    }

    private static void saveEmployees(List<Employee> employees) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
            oos.writeObject(employees);
        } catch (IOException e) {
            System.err.println("Failed to save employees: " + e.getMessage());
        }
    }

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

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
            return (List<Employee>) ois.readObject();
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Failed to load employees: " + e.getMessage());
        }
        return new ArrayList<>();
    }
}
```

3. Output:

```
Menu:
1. Add Employee
2. Display All Employees
3. Exit
Select an option: 1
Enter Employee ID: 50001
Enter Employee Name: Reshma Saluja
Enter Designation: Software Engg
Enter Salary: 50000
Employee added successfully.
```