Experiment 5

Student Name: Prince **UID:22BCS14033**
Branch: B.E CSE **Section: IOT-643-A**
Semester: 6th **DOP:24/02/25**
Subject: PBLJ **Subject Code: 22CSH-359**

Aim:

Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.

Problem Statement :

1) 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) 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.

3) 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.

Algorithm:

1. **Sum of a List of Integers Using Autoboxing & Unboxing:**

   - **Initialize an empty list** to store integers.
   - **Prompt the user** to enter integers.
   - **Read input as a string**, and if it's a valid number, parse it using Integer.parseInt().
     - **Autoboxing** occurs when adding int values to the List<Integer>.
   - Repeat until the user enters "stop".
   - Call a method calculateSum():
     - **Iterate through the list** and perform **unboxing** (Integer → int) while calculating the sum.

2. Student Serialization & Deserialization:

- Create a Student class with fields (id, name, GPA) and implement Serializable.
- In the main method:
  - Prompt the user to enter student details.
  - Create a Student object with user input.
- Serialize (Save) the Student object:
  - Open a file using FileOutputStream.
  - Write the Student object using ObjectOutputStream.
  - Handle IOException.
- Deserialize (Load) the Student object:
  - Open the same file using FileInputStream.
  - Read the object using ObjectInputStream.
  - Cast it back to a Student object.
  - Handle FileNotFoundException, IOException, and ClassNotFoundException.
- Print the student details after deserialization.
- End program.

3. Employee Management System (Menu-Based) :

- Create Employee class (fields: id, name, designation, salary), implement Serializable.
- Load employees from file (if available).
- Menu:
  - Add Employee → Get details, create object, append to list, serialize & save.
  - Display All Employees → Deserialize & print details.
  - Exit → Terminate.
- Handle exceptions (FileNotFoundException, IOException, ClassNotFoundException).

Program :

1. Sum of a List of Integers Using Autoboxing & Unboxing:

```
import java.util.*;

public class AutoboxingUnboxingExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter integers (type 'stop' to finish):");

        int sum = 0;
        while (scanner.hasNextInt()) {
            sum += scanner.nextInt();
        }

        System.out.println("Sum: " + sum);
        scanner.close();
    }
}
```

2. Student Serialization & Deserialization:

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

class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    int id;
    String name;
    double gpa;

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

    @Override
    public String toString() {
        return "Student ID: " + id + ", Name: " + name + ", GPA: " + gpa;
    }
}

public class StudentManagement {
    private static final String FILE_NAME = "students.ser";
```

```
    private static final Scanner scanner = new Scanner(System.in);
    private static List<Student> students = new ArrayList<>();

    private static void saveStudents() {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            oos.writeObject(students);
        } catch (IOException e) {
            System.out.println("Error saving students: " + e.getMessage());
        }
    }

    private static void loadStudents() {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            students = (List<Student>) ois.readObject();
        } catch (FileNotFoundException e) {
            System.out.println("No saved students found.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error loading students: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        loadStudents();
        while (true) {
            System.out.println("\n1. Add Student  2. Display Students  3. Exit");
            System.out.print("Choice: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1 -> {
                    System.out.print("ID: ");
                    int id = scanner.nextInt();
                    scanner.nextLine();
                    System.out.print("Name: ");
                    String name = scanner.nextLine();
                    System.out.print("GPA: ");
                    double gpa = scanner.nextDouble();
                    students.add(new Student(id, name, gpa));
                    saveStudents();
                }
                case 2 -> students.forEach(System.out::println);
```

```
            case 3 -> {
                System.out.println("Exiting...");
                return;
            }
            default -> System.out.println("Invalid choice.");
        }
    }
}
```

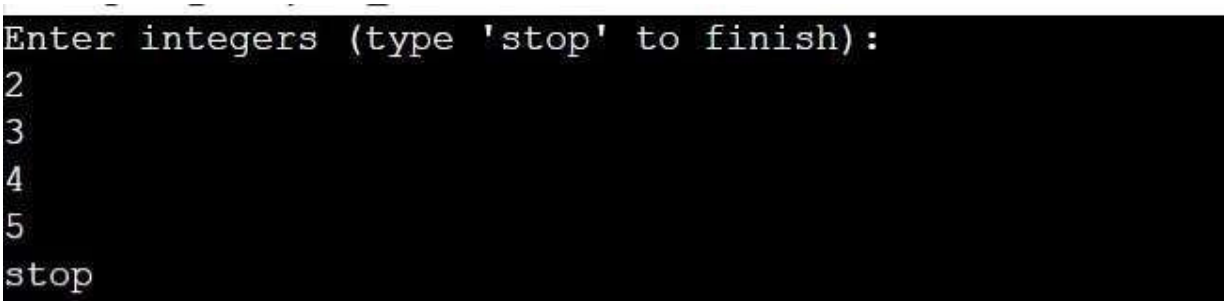
3. Employee Management System (Menu-Based) :

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

class Employee {
    String id, name, designation;
    double salary;

    public Employee(String 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 + ", " + name + ", " + designation + ", " + salary;
    }
}

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

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\n1. Add Employee 2. Display All 3. Exit");
            System.out.print("Choice: ");
            switch (scanner.nextInt()) {
                case 1 -> addEmployee(scanner);
                case 2 -> displayEmployees();
                case 3 -> {
                    System.out.println("Goodbye!");
                    scanner.close();
                    return;
```

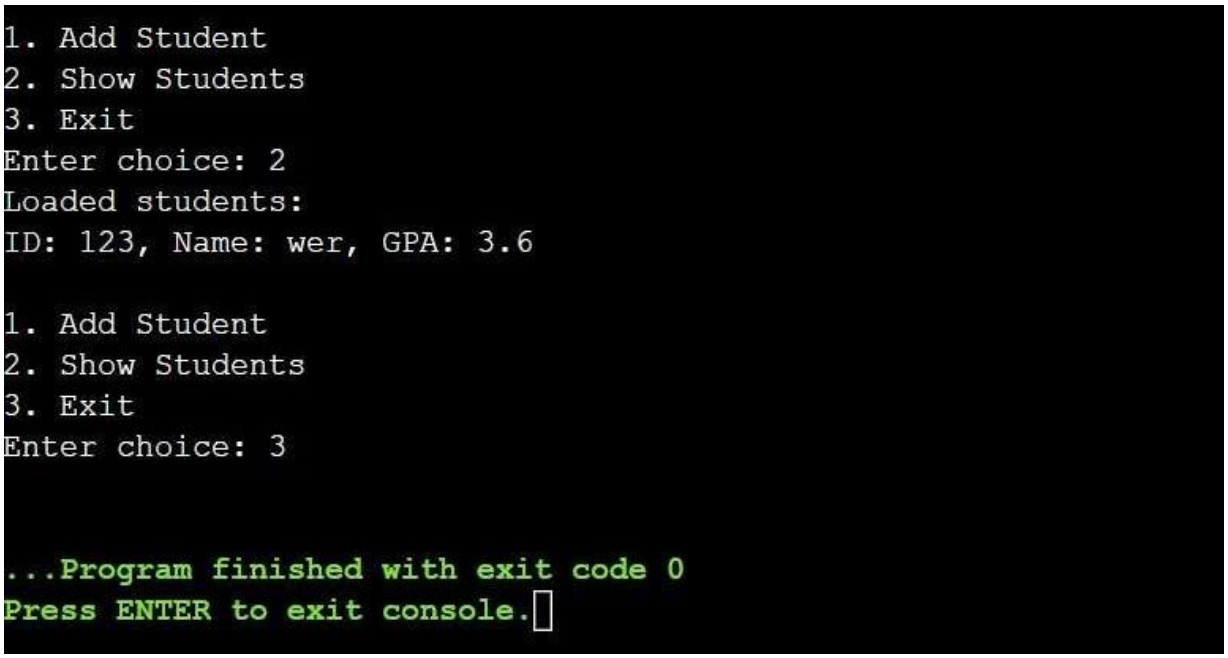
```
                }
                default -> System.out.println("Invalid choice!");
            }
        }
    }

    private static void addEmployee(Scanner scanner) {
        try (PrintWriter out = new PrintWriter(new FileWriter(FILE_NAME, true))) {
            scanner.nextLine();
            System.out.print("ID: ");
            String id = scanner.nextLine();
            System.out.print("Name: ");
            String name = scanner.nextLine();
            System.out.print("Designation: ");
            String designation = scanner.nextLine();
            System.out.print("Salary: ");
            double salary = scanner.nextDouble();
            out.println(new Employee(id, name, designation, salary));
            System.out.println("Employee added!");
        } catch (IOException e) {
            System.out.println("Error saving employee.");
        }
    }

    private static void displayEmployees() {
        try (BufferedReader br = new BufferedReader(new FileReader(FILE_NAME))) {
            System.out.println("\nEmployees:");
            br.lines().forEach(System.out::println);
        } catch (IOException e) {
            System.out.println("No employees found.");
        }
    }
}
```

OUTPUT :

1. **Sum of a List of Integers Using Autoboxing & Unboxing:**



2. Student Serialization & Deserialization:



3. Employee Management System (Menu-Based) :

Learning Outcomes:

- **Implement object-oriented programming with classes, encapsulation, and serialization.**
- **Utilize core Java concepts like loops, conditionals, autoboxing, and unboxing.**
- **Apply file handling with serialization, deserialization, and exception management.**