Experiment - 5

Student Name: Anish Anand   UID: 22BCS10055
Branch: BE-CSE   Section/Group: 22BCS 639 - A
Semester: 6th   Date of Performance:21/02/2025
Subject Name: Project Based Learning   Subject Code: 22CSH-359
                in Java with Lab

1. **Aim:** Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.

2. **Objective:** To demonstrate autoboxing, unboxing, and collection handling in Java, along with object serialization and deserialization while implementing proper exception handling.
Additionally, to implement a menu-based employee management system using collections.

3. **Implementation/Code:**

   3.1. **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()).**

```java
import java.util.ArrayList;
import java.util.List;

public class SumUsingAutoboxing {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        numbers.add(parseInteger("10"));
        numbers.add(parseInteger("20"));
        numbers.add(parseInteger("30"));
        numbers.add(parseInteger("40"));
        numbers.add(parseInteger("50"));
        int sum = calculateSum(numbers);
        System.out.println("Sum of numbers: " + sum);
    }
    private static Integer parseInteger(String str) {
        return Integer.parseInt(str);
    }
    private static int calculateSum(List<Integer> numbers) {
        int sum = 0;
        for (Integer num : numbers) {
            sum += num;
        }
        return sum;
    }
}
```

   3.2. **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.

```java
import java.io.*;

class Student implements Serializable {
private static final long serialVersionUID = 1L;
    private int id;
  private String name;
  private double gpa;
public Student(int id, String name, double gpa) {
this.id = id; this.name = name; this.gpa = gpa;
} public void display() {
    System.out.println("Student ID: " + id);
    System.out.println("Name: " + name);
    System.out.println("GPA: " + gpa);
    }
    }
public class StudentSerialization {
private static final String FILE_NAME = "student.ser";
public static void main(String[] args) {
    Student student = new Student(101, "ABCD", 8.3);
    serializeStudent(student);
    deserializeStudent();
} private static void serializeStudent(Student student) {
try (ObjectOutputStream oos = new ObjectOutputStream(newFileOutputStream(FILE_NAME))) {
    oos.writeObject(student);
    System.out.println("Student object serialized successfully.");
        }
    catch(IOException e)
    {
    System.err.println("Error during serialization: " + e.getMessage());
    }
} private static void deserializeStudent() {
try (ObjectInputStream ois = new ObjectInputStream(newFileInputStream(FILE_NAME))) {
    Student student = (Student) ois.readObject();
    System.out.println("Deserialized Student Object:");
    student.display();
        }
    catch(FileNotFoundException e)
    {
    System.err.println("File not found: " + e.getMessage());
        }
    catch(IOException e)
    {
```

```
        System.err.println("Error during deserialization: " + e.getMessage());
            } catch
        (ClassNotFoundException e) {
            System.err.println("Class not found: " + e.getMessage()); }
        }
        }
```

3.3 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.

```java
import java.util.ArrayList;
import java.util.Scanner;
class Employee { int
    id;
    String name; String
    designation; 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 EmployeeManagement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); ArrayList<Employee>
        employees = new ArrayList<>(); while (true) {
            System.out.println("\n1. Add an 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(); switch (choice) {

                case 1:
                    System.out.print("Enter Employee ID: "); int id =
                    scanner.nextInt();
                    scanner.nextLine(); // Consume newline System.out.print("Enter Name:
                    ");
                    String name = scanner.nextLine();
```
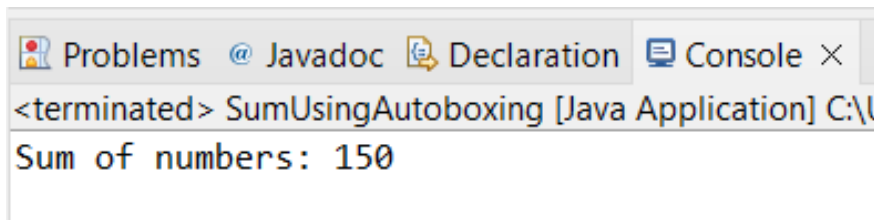
```java
        System.out.print("Enter Designation: ");
        String designation = scanner.nextLine(); System.out.print("Enter
        Salary: "); double salary = scanner.nextDouble(); employees.add(new
        Employee(id, name, designation, salary));
        System.out.println("Employee added successfully."); break;

    case 2: if (employees.isEmpty())
        {
            System.out.println("No employees found.");
        } else {
            System.out.println("\nEmployee List:");
            for (Employee emp : employees) { System.out.println(emp);
            }
        } break;
        case 3:
        System.out.println("Exiting application.");
        scanner.close(); System.exit(0); break;
    default:
        System.out.println("Invalid choice. Please try again."); }
        }
    }
}
```
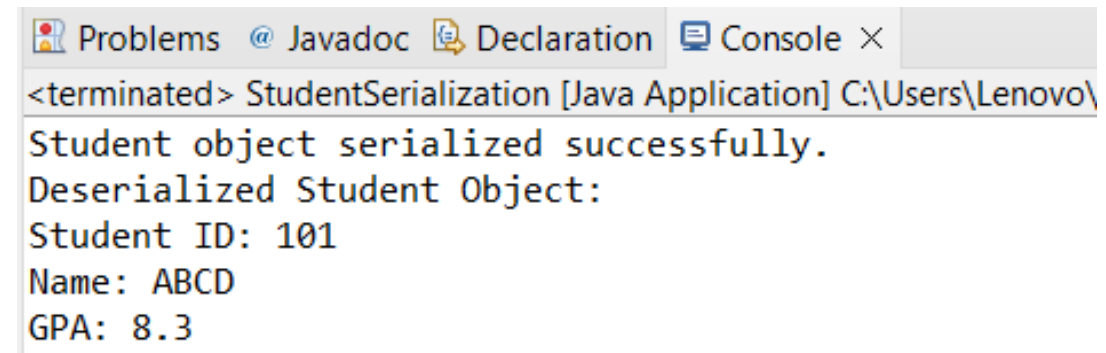
4. Output:

    4.1.

```
Problems  @ Javadoc  Declaration  Console  ×
<terminated> SumUsingAutoboxing [Java Application] C:\L
Sum of numbers: 150
```

    4.2.

```
Problems  @ Javadoc  Declaration  Console  ×
<terminated> StudentSerialization [Java Application] C:\Users\Lenovo\
Student object serialized successfully.
Deserialized Student Object:
Student ID: 101
Name: ABCD
GPA: 8.3
```

4.3.

```
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee ID: 101
Enter Name: ABCD
Enter Designation: Manager
Enter Salary: 110000
Employee added successfully.

1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 2

Employee List:
ID: 101, Name: ABCD, Designation: Manager, Salary: 110000.0
```

5. Learning Outcomes:

- Understand autoboxing and unboxing in Java.
- Learn object serialization and deserialization using streams.
- Handle exceptions like IOException and ClassNotFoundException.
- Work with collections and perform arithmetic operations.
- Use try-with-resources for efficient file handling.
- Implement a menu-driven employee management system using collections.