Experiment 1.3

Student Name: Nikhil Sharma **UID:**22BCS15209
Branch:CSE **Section/Group:**640/B
Semester: 6 **Date of Performance:**05/03/25
Subject Name:project based learning using java **Subject Code:** 22CSH-354

Aim 1: Implement and manage data using Java Collections and Exception Handling

objective : Easy Level: 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()).

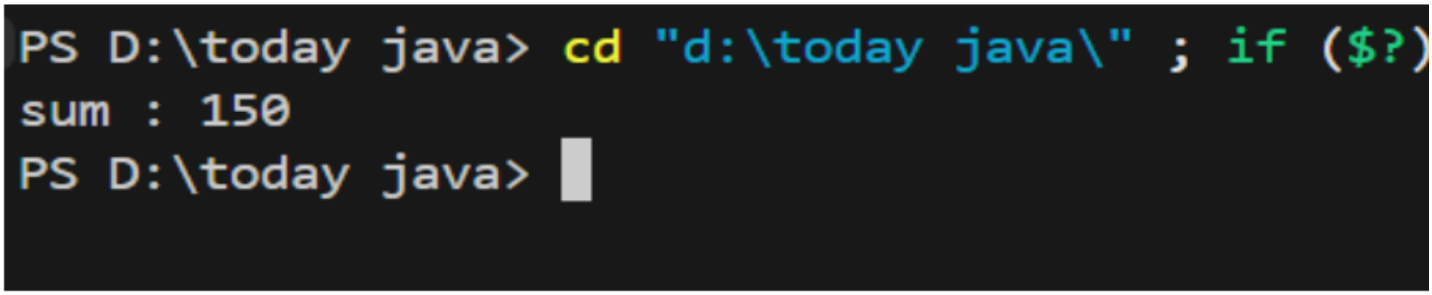
• Medium Level: Create a Java program to serialize and deserialize a Student object. The program should:
o Serialize a Student object (containing ID, Name, and GPA) and save it to a file.
o Deserialize the object from the file and display the student details.
o Handle FileNotFoundException, IOException, and ClassNotFoundException using exception handling.

• Hard Level: 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.

Problem 1.1: Sum of Integers Using Autoboxing and Unboxing

```
import java.util.*;
public class SumCalculator {
 public static int calculateSum(List<Integer> numbers) {
 int sum = 0;
 for (Integer num : numbers) {
 sum += num; // Autounboxing happens here
 }
 return sum;
 }
 public static void main(String[] args) {
 List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);
 System.out.println("Sum: " + calculateSum(numbers));
 }
}
```

output 1.1



Problem 1.2: Serialization and Deserialization of a Student Object

```
import java.io.*;

class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    private String name;
    private int age;
    private String department;

    public Student(String name, int age, String department) {
        this.name = name;
        this.age = age;
        this.department = department;
    }

    @Override
    public String toString() {
        return "Student{name='" + name + "', age=" + age + ", department='" + department + "'}";
    }
}

public class SerializationDemo {
    public static void serializeStudent(Student student, String filename) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))) {
            oos.writeObject(student);
            System.out.println("Serialization successful!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

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

```
    }

    public static void main(String[] args) {
        String filename = "student.ser";
        Student student = new Student("Nikhil Sharma", 21, "CSE");

        // Serialize
        serializeStudent(student, filename);

        // Deserialize
        Student deserializedStudent = deserializeStudent(filename);
        System.out.println("Deserialized Student: " + deserializedStudent);
    }
}
```

output 1.2 -

```
Serialization successful!
Deserialized Student: Student{name='Nikhil Sharma', age=21, department='CSE'}
```

Problem 1.3: Employee Management System (Menu-Based Application)

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

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

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

    @Override
    public String toString() {
        return "Employee{id=" + id + ", name='" + name + "', salary=" + salary + "}";
    }
}

public class EmployeeManagement {
    private static final String FILE_NAME = "employees.ser";
    private static List<Employee> employeeList = new ArrayList<>();

    public static void addEmployee(Employee emp) {
```

```
    employeeList.add(emp);
    saveEmployees();
}

public static void displayEmployees() {
    if (employeeList.isEmpty()) {
        System.out.println("No employees found.");
    } else {
        for (Employee emp : employeeList) {
            System.out.println(emp);
        }
    }
}

public static void saveEmployees() {
    try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME))) {
        oos.writeObject(employeeList);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void loadEmployees() {
    try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME))) {
        employeeList = (List<Employee>) ois.readObject();
    } catch (IOException | ClassNotFoundException e) {
        employeeList = new ArrayList<>();
    }
}

public static void main(String[] args) {
    loadEmployees();
    Scanner scanner = new Scanner(System.in);
    while (true) {
        System.out.println("\nEmployee Management System");
        System.out.println("1. Add Employee");
        System.out.println("2. Display Employees");
        System.out.println("3. Exit");
        System.out.print("Enter your choice: ");
        int choice = scanner.nextInt();
        switch (choice) {
            case 1:
                System.out.print("Enter ID: ");
                int id = scanner.nextInt();
                scanner.nextLine(); // Consume newline
                System.out.print("Enter Name: ");
                String name = scanner.nextLine();
                System.out.print("Enter Salary: ");
                double salary = scanner.nextDouble();
                addEmployee(new Employee(id, name, salary));
```

```
                System.out.println("Employee added successfully!");
                break;
            case 2:
                displayEmployees();
                break;
            case 3:
                System.out.println("Exiting...");
                scanner.close();
                System.exit(0);
            default:
                System.out.println("Invalid choice! Try again.");
        }
    }
}
```

OUTPUT 1.3 -

```
Employee Management System
1. Add Employee
2. Display Employees
3. Exit
Enter your choice: 1
Enter ID: 101
Enter Name: John Doe
Enter Salary: 50000
Employee added successfully!

Employee Management System
1. Add Employee
2. Display Employees
3. Exit
Enter your choice: 2
Employee{id=101, name='John Doe', salary=50000.0}

Employee Management System
1. Add Employee
2. Display Employees
3. Exit
Enter your choice: 3
Exiting...
```

Learning Outcomes:

1. Object Serialization & Deserialization -Understanding how to save and retrieve objects using Java's Serializable interface to ensure data persistence.
2. File Handling in Java -Learning how to read and write objects to files using ObjectOutputStream and ObjectInputStream.
3. Menu-Driven Application Development - Implementing a user-friendly console-based menu system with Scanner for user input handling.
4. Collection Framework Usage -Gaining hands-on experience with Java's ArrayList to store and manage dynamic collections of Employee objects.