Experiment-5

Student Name: Palak Khullar
UID: 22BCS13306
Branch: BE-CSE
Section/Group: 618-B
Semester: 6th
Date of Performance: 28/02/25
Subject Name: Project Based Learning in Java
Subject Code: 22CSH-359

EASY:

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())

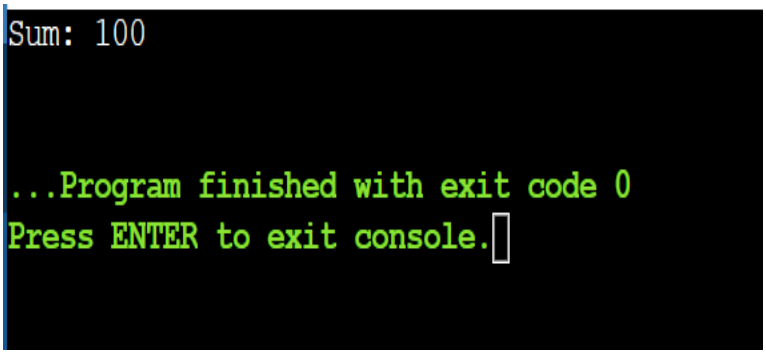
Objective: To implement a Java program that calculates the sum of a list of integers using autoboxing and unboxing. The program should also parse string inputs into their respective wrapper classes.

Implementation/Code:

```
import java.util.*;

class MyInteger {
    private int value;
    public MyInteger(int value) {
        this.value = value;
    }
    public int getValue() {
        return value;
    }

    public static MyInteger parseMyInt(String str) {
        return new MyInteger(Integer.parseInt(str));
    }
```

```
}

public class SumUsingWrapper {
    public static int sumList(List<String> numbers) {
        int sum = 0;
        for (String num : numbers) {
            sum += MyInteger.parseMyInt(num).getValue();
        }
        return sum;
    }

    public static void main(String[] args) {
        List<String> numbers = Arrays.asList("10", "20", "30", "40");
        System.out.println("Sum: " + sumList(numbers));
    }
}
```

Output



MEDIUM:

Aim: 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.

Objective: To develop a Java program that serializes and deserializes a Student object containing ID, name, and GPA. The program should handle file operations and exceptions like FileNotFoundException, IOException, and ClassNotFoundException.

Code/Implementation:

```
import java.io.*;


class MyString implements Serializable {
    private static final long serialVersionUID = 1L;
    private String value;

    public MyString(String value) {
        this.value = value;
    }

    public String getValue() {
        return value;
    }
}

class Student implements Serializable {
```

```
    private static final long serialVersionUID = 1L;
    int id;
    MyString name;
    double gpa;

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

    public void display() {
        System.out.println("ID: " + id + ", Name: " + name.getValue() + ", GPA: " + gpa);
    }
}

public class StudentSerialization {
    public static void main(String[] args) {
        String filename = "student.ser";

        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(filename))) {
            Student student = new Student(101, new MyString("Alice"), 3.8);
            oos.writeObject(student);
            System.out.println("Student object serialized successfully.");
        } catch (IOException e) {
            e.printStackTrace();
```

```
        }

        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filename))) {
            Student student = (Student) ois.readObject();
            System.out.println("Deserialized Student:");
            student.display();
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Student object serialized successfully.
Deserialized Student:
ID: 101, Name: Alice, GPA: 3.8


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

StudentSerializatio... | student.ser

```
1  ¬í··sr··Student··········D··gpaI··idL··namet·
2  LMyString;xp@·ffffff···esr··MyString··········L··valuet··Ljava/lang/String;xpt··Alice
```

HARD:

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.

Objective: To create a menu-driven Java application for managing employee records by adding and displaying employee details. The data should be stored in a file, retrieved when needed, and the program should exit when required.

Code/Implementation:

```
import java.io.*;

import java.util.*;


class MyDouble implements Serializable {

    private static final long serialVersionUID = 1L;

    private double value;


    public MyDouble(double value) {

        this.value = value;

    }


    public double getValue() {
```

```
        return value;
    }
}
class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    int id;
    String name, designation;
    MyDouble salary;

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

    public void display() {
        System.out.println("ID: " + id + ", Name: " + name + ", Designation: " +
designation + ", Salary: " + salary.getValue());
    }
}

public class EmployeeManagement {
    private static final String FILE_NAME = "employees.ser";
```

```
    private static List<Employee> employeeList = new ArrayList<>();


    public static void addEmployee() {
        Scanner sc = new Scanner(System.in);
        try {
            System.out.print("Enter ID: ");
            int id = sc.nextInt();
            sc.nextLine();
            System.out.print("Enter Name: ");
            String name = sc.nextLine();
            System.out.print("Enter Designation: ");
            String designation = sc.nextLine();
            System.out.print("Enter Salary: ");
            double salary = sc.nextDouble();


            Employee emp = new Employee(id, name, designation, new
MyDouble(salary));
            employeeList.add(emp);
            saveToFile();
            System.out.println("Employee added successfully!");
        } catch (InputMismatchException e) {
            System.out.println("Invalid input! Please enter the correct data type.");
            sc.nextLine();
        }
```

```
    }
    public static void displayEmployees() {
        loadFromFile();
        if (employeeList.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            System.out.println("Employee Details:");
            for (Employee emp : employeeList) {
                emp.display();
            }
        }
    }
    private static void saveToFile() {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            oos.writeObject(employeeList);
        } catch (IOException e) {
            System.out.println("Error saving employee data.");
        }
    }


    private static void loadFromFile() {
        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) {
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.println("\n1. Add Employee\n2. Display All\n3. Exit");
            System.out.print("Choose an option: ");
            int choice = sc.nextInt();
            switch (choice) {
                case 1:
                    addEmployee();
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    System.out.println("Exiting application.");
                    return;
                default:
                    System.out.println("Invalid choice! Try again.");
            }
```

```
        }
    }
}
```

Output:

```
1. Add Employee
2. Display All
3. Exit
Choose an option: 1
Enter ID: 101
Enter Name: Elena
Enter Designation: Developer
Enter Salary: 60000
Employee added successfully!

1. Add Employee
2. Display All
3. Exit
Choose an option: 1
Enter ID: 102
Enter Name: Caroline
Enter Designation: Intern
Enter Salary: 30000
Employee added successfully!

1. Add Employee
2. Display All
3. Exit
Choose an option: 2
Employee Details:
ID: 101, Name: Elena, Designation: Developer, Salary: 60000.0
ID: 102, Name: Caroline, Designation: Intern, Salary: 30000.0
```

Learning Outcome:

1. Understand the concepts of wrapper classes, autoboxing, unboxing, and object serialization in Java.
2. Gain hands-on experience in file handling, exception handling, and creating menu-driven applications for data storage and retrieval.