EXPERIMENT: 04

Easy: Autoboxing and Unboxing

Code/Implementation

```
import java.util.*;

public class WrapperExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter numbers separated by spaces:");
        String input = scanner.nextLine();
        String[] numbers = input.split(" ");

        int sum = 0;
        for (String num : numbers) {
            sum += Integer.parseInt(num);
        }

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

Medium Level: Serialization and Deserialization

Code/Implementation

```
import java.io.*;

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;

   }


   public void display() {

      System.out.println("ID: " + id + ", Name: " + name + ", GPA: " + gpa);

   }

}


public class SerializeDemo {

   public static void main(String[] args) {

      Student student = new Student(101, "Alice", 3.9);

      String filename = "student.ser";



      try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filename))) {

         oos.writeObject(student);

         System.out.println("Serialization successful.");
```

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

Hard Level: Menu-Based Employee Management

Code/Implementation

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

class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
    int id;
    String name, 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;
    }
    public void display() {
        System.out.println("ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " +
salary);
    }
}

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

    public static void main(String[] args) {
        loadEmployees(); // Load existing employees from file
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n1. Add 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:
                    addEmployee(scanner);
                    saveEmployees();
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    saveEmployees();
                    System.out.println("Exiting...");
                    scanner.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid choice! Try again.");
            }
        }
    }

    private static void addEmployee(Scanner scanner) {
        System.out.print("Enter ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter 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 displayEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
```

```
            return;
        }
        for (Employee emp : employees) {
            emp.display();
        }
    }

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

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