Experiment -5

Student Name: Shivangi **UID: 22BCS16953**

Branch: BE-CSE **Date of Performance: 03-03-2025**

Semester: 6th **Section/Group: 22BCS_EPAM-801/ B**

Subject Name: Project based learning in java with lab **Subject Code: 22CSH-359**

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

2. **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:
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.

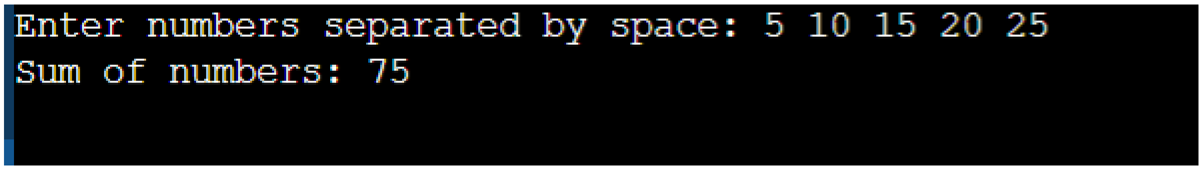
Hard Level:

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.

3. Implementation/Code:

Easy Level:

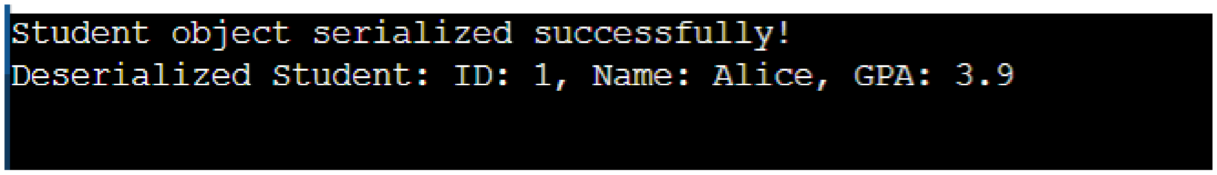
```
import java.util.ArrayList;
import java.util.Scanner;

public class Autoboxing {
    public static int calculateSum(ArrayList<Integer> numbers) {
        int sum = 0;
        for (Integer num : numbers) {
            sum += num;
        }
        return sum;
    }

    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter numbers separated by space: ");
        String input = scanner.nextLine();
        String[] tokens = input.split(" ");

        for (String token : tokens) {
            numbers.add(Integer.parseInt(token));
        }

        int sum = calculateSum(numbers);
        System.out.println("Sum of numbers: " + sum);

        scanner.close();
    }
}
```



```
Enter numbers separated by space: 5 10 15 20 25
Sum of numbers: 75
```

Medium Level:

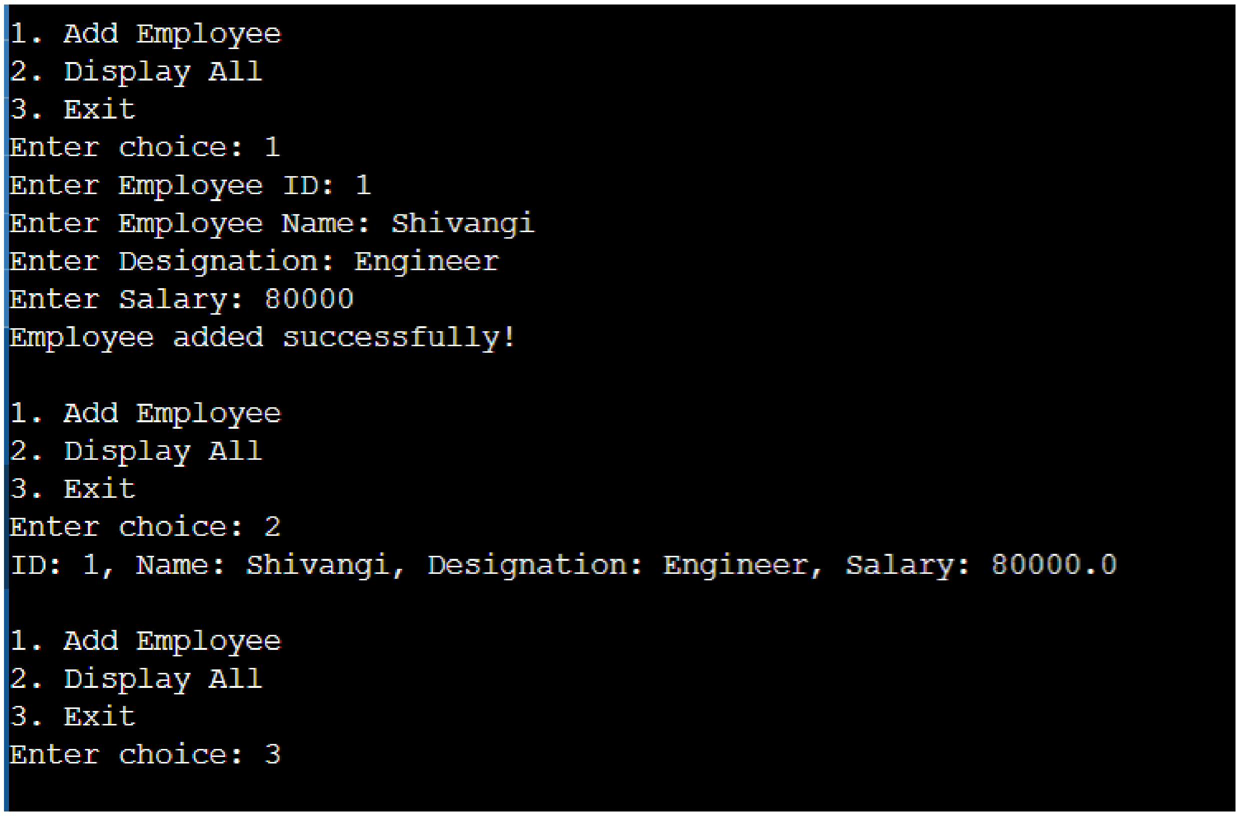
```
import java.io.*;
class Student implements Serializable {
    private static final long serialVersionUID = 1L;
    int id;
    String name;
    double gpa;
    Student(int id, String name, double gpa) {
        this.id = id;
        this.name = name;
        this.gpa = gpa;
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Name: " + name + ", GPA: " + gpa;
    }
}
public class StudentSerialization {
    private static final String FILE_NAME = "student.ser";

    public static void serializeStudent(Student student) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            oos.writeObject(student);
            System.out.println("Student object serialized successfully!");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
    public static void deserializeStudent() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME)))
{
            Student student = (Student) ois.readObject();
            System.out.println("Deserialized Student: " + student);
        } catch (FileNotFoundException e) {
            System.out.println("File not found!");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void main(String[] args) {
        Student student = new Student(1, "Alice", 3.9);
        serializeStudent(student);
        deserializeStudent();
    }
}
```



```
Student object serialized successfully!
Deserialized Student: ID: 1, Name: Alice, GPA: 3.9
```

Hard Level:

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

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

    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 {
    private static final String FILE_NAME = "employees.dat";

    public static void addEmployee() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Enter Employee Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Designation: ");
        String designation = scanner.nextLine();
        System.out.print("Enter Salary: ");
        double salary = scanner.nextDouble();
        Employee emp = new Employee(id, name, designation, salary);
        List<Employee> employees = readEmployees();
        employees.add(emp);
        writeEmployees(employees);

        System.out.println("Employee added successfully!");
    }
    public static List<Employee> readEmployees() {
        List<Employee> employees = new ArrayList<>();
        File file = new File(FILE_NAME);

        if (!file.exists()) {
            return employees;
        }
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME)))
{
            employees = (List<Employee>) ois.readObject();
        } catch (EOFException e) {
            System.out.println("File is empty. No employees found.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error reading employees: " + e.getMessage());
        }
        return employees;
    }

    public static void writeEmployees(List<Employee> employees) {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            oos.writeObject(employees);
        } catch (IOException e) {
            System.out.println("Error writing employees: " + e.getMessage());
        }
    }

    public static void displayEmployees() {
        List<Employee> employees = readEmployees();
        if (employees.isEmpty()) {
            System.out.println("No employees found!");
        } else {
            for (Employee emp : employees) {
                System.out.println(emp);
            }
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n1. Add Employee\n2. Display All\n3. Exit");
            System.out.print("Enter choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    addEmployee();
                    break;
                case 2:
                    displayEmployees();
                    break;
                case 3:
                    scanner.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid option, try again.");
            }
        }
    }
}
```



```
1. Add Employee
2. Display All
3. Exit
Enter choice: 1
Enter Employee ID: 1
Enter Employee Name: Shivangi
Enter Designation: Engineer
Enter Salary: 80000
Employee added successfully!

1. Add Employee
2. Display All
3. Exit
Enter choice: 2
ID: 1, Name: Shivangi, Designation: Engineer, Salary: 80000.0

1. Add Employee
2. Display All
3. Exit
Enter choice: 3
```

4. Learning Outcomes:

- **Autoboxing & Unboxing:** Understand how primitive types are automatically converted into their wrapper classes and vice versa.
- **String Parsing:** Learn to convert string values into numeric types using wrapper class methods (e.g., Integer.parseInt()).
- **Serialization & Deserialization:** Gain hands-on experience in saving and retrieving objects using ObjectOutputStream and ObjectInputStream.
- **Exception Handling:** Learn to handle file-related exceptions like FileNotFoundException, IOException, and ClassNotFoundException.
- **File Handling:** Understand reading and writing data to files using streams (FileInputStream, FileOutputStream).
- **Efficient Data Management:** Implement list-based storage and update mechanisms to avoid file corruption in serialization.