Experiment 5

Student Name: Akriti Barthwal
UID: 22BCS14027
Branch: BE CSE
Section/Group: Epam-801 A
Semester: 06
Date of Performance: 23-2-25
Subject Name: Project Based Learning in Java
Subject Code: 22CSH-359

1. Aim-

Easy: 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: 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: 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.

2. Code-

EASY:

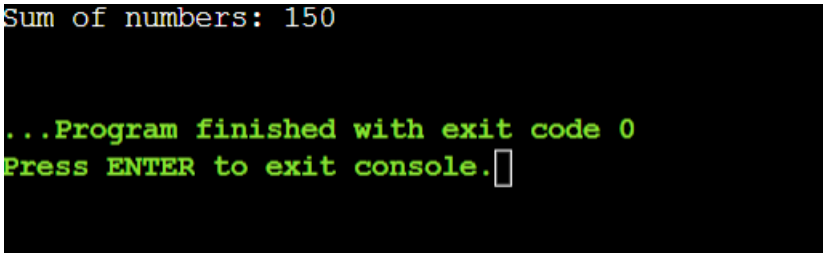
```
import java.util.ArrayList;

public class AutoBoxingExample {
    public static void main(String[] args) {
        // Creating an integer array
        int[] values = {10, 20, 30, 40, 50};

        // Creating a list of Integer objects (Autoboxing happens here)
        ArrayList<Integer> numbers = new ArrayList<>();
        for (int value : values) {
            numbers.add(value);  // Autoboxing: int → Integer
        }

        // Calculating the sum (Unboxing happens here)
        int sum = 0;
        for (Integer num : numbers) {
            sum += num;  // Unboxing: Integer → int
        }
```

```
        // Printing the sum
        System.out.println("Sum of numbers: " + sum);
    }
```



MEDIUM:

```
import java.io.*;

// Serializable Student class
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 String toString() {
        return "ID: " + id + ", Name: " + name + ", GPA: " + gpa;
    }
}

public class StudentSerialization {
    public static void main(String[] args) {
        Student student = new Student(101, "Akriti", 8.5); String
        filename = "student.ser";

        serializeStudent(student, filename);
        deserializeStudent(filename);
    }

    // Serialize Student object
    public static void serializeStudent(Student student, String filename) {
        try (ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(filename)))
{
            out.writeObject(student);
```

```
            System.out.println("Student serialized successfully!");
        } catch (Exception e) {
            System.out.println("Serialization Error: " + e.getMessage());
        }
    }

    // Deserialize Student object
    public static void deserializeStudent(String filename) {
        try (ObjectInputStream in = new ObjectInputStream(new FileInputStream(filename))) {
            Student student = (Student) in.readObject();
            System.out.println("Deserialized Student: " + student);
        } catch (Exception e) {
            System.out.println("Deserialization Error: " + e.getMessage());
        }
    }
}
```

```
Student serialized successfully!
Deserialized Student: ID: 101, Name: Akriti, GPA: 8.5


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

HARD:

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

class Employee implements Serializable {
    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;
    }

    public String toString() {
        return "ID: " + id + ", Name: " + name + ", Designation: " + designation + ", Salary: " + salary;
    }
}

public class EmployeeManager {
```

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

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    List<Employee> employeeList = loadEmployees();

    while (true) {
        System.out.println("\nMenu:");
        System.out.println("1. 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(); // Consume newline

        if (choice == 1) {
            addEmployee(scanner, employeeList);
        } else if (choice == 2) {
            displayEmployees(employeeList);
        } else if (choice == 3) {
            saveEmployees(employeeList);
            System.out.println("Exiting program...");
            scanner.close();
            System.exit(0);
        } else {
            System.out.println("Invalid choice! Please enter 1, 2, or 3.");
        }
    }
}

private static void addEmployee(Scanner scanner, List<Employee> employeeList) {
    System.out.print("Enter Employee ID: ");
    int id = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    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();

    employeeList.add(new Employee(id, name, designation, salary));
    System.out.println("Employee added successfully!");
}

private static void displayEmployees(List<Employee> employeeList) {
    if (employeeList.isEmpty()) {
        System.out.println("No employees found.");
    } else {
        System.out.println("\nEmployee List:");
        for (Employee emp : employeeList) {
            System.out.println(emp);
        }
    }
```

```
    }

    private static void saveEmployees(List<Employee> employeeList) {
        try {
            ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_NAME));
            oos.writeObject(employeeList);
            oos.close();
        } catch (IOException e) {
        }
    }

    private static List<Employee> loadEmployees() {
        File file = new File(FILE_NAME);
        if (!file.exists()) {
            return new ArrayList<>();
        }

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

```
Menu:
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee ID: 123
Enter Employee Name: akriti
Enter Designation: student
Enter Salary: 10000
Employee added successfully!

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

Employee List:
ID: 121, Name: klndks, Designation: wdnmewd, Salary: 1323.0
ID: 123, Name: akriti, Designation: student, Salary: 10000.0
```

3. Learning Outcomes-

- Autoboxing **& Unboxing:** Efficiently convert between primitive types and their wrapper classes in Java.
- Serialization **& Deserialization:** Store and retrieve object states using file handling.
- Object-**Oriented Design:** Implement classes with attributes and methods, demonstrating encapsulation.
- File **I/O Operations:** Read from and write to files for persistent data storage.
- Menu-**Driven Programming:** Build interactive console applications with dynamic user input handling.