



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment-5

Student Name: Prithvi Singh

Branch: CSE

Semester: 6th

Subject Name: Java Lab

UID: 22BCS12368

Section/Group: IOT-618/A

Date of Performance: 21/02/25

Subject Code: 22CSH-359

Problem-1 (Easy)

1. Aim:

writing a Java program to calculate the sum of a list of integers using autoboxing and unboxing, along with methods to parse strings into their respective wrapper classes (e.g., Integer.parseInt()).

2. Implementation/Code:

```
//Prithvi Singh
```

```
//22BCS12368
```

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

```
import java.util.List;
```

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

```
public class IntegerSumCalculator {
```

```
    public static Integer parseStringToInteger(String str) {
```

```
        try {
```

```
            return Integer.parseInt(str);
```

```
        } catch (NumberFormatException e) {
```

```
            System.out.println("Invalid number format: " + str);
```

```
            return 0;
```

```
    }  
}  
  
    public static int calculateSum(List<Integer> numbers)  
{  
    int sum = 0;    for (Integer num : numbers) {  
sum += num;  
    }  
return sum;  
}  
  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        List<Integer> numbers = new ArrayList<>();  
  
        System.out.print("Enter the number of values you want to input: ");  
        int n = scanner.nextInt();  
  
        System.out.println("Enter " + n + " numbers:");  
  
        for (int i = 0; i < n; i++) {            String input  
= scanner.next();  
        numbers.add(parseStringToInteger(input));  
    }  
  
        scanner.close();  
  
        int totalSum = calculateSum(numbers);  
        System.out.println("The sum of the list is: " + totalSum);  
    }  
}
```

```
}  
}
```

3. Output:

```
Enter the number of values you want to input: 5  
Enter 5 numbers:  
10 20 30 40 50  
The sum of the list is: 150
```

```
Enter the number of values you want to input: 3  
Enter 3 numbers:  
100 200 300  
The sum of the list is: 600
```

Problem-2 (Medium)

1. Aim:

Java program that serializes and deserializes a Student object. It saves the Student object to a file and then reads it back, displaying the student details.

The program handles exceptions like FileNotFoundException, IOException, and ClassNotFoundException.

2. Implementation/Code:

```
//Prithvi Singh //22BCS12368  
import java.io.*;  
class Student implements Serializable {    private  
    static final long serialVersionUID = 1L;    private  
    int id;    private String name;    private double  
    gpa;  
  
    public Student(int id, String name, double gpa) {  
        this.id = id;        this.name = name;        this.gpa =  
        gpa;
```

```
    }

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

public class StudentSerialization {

    public static void serializeStudent(Student student, String filename) {
        try (ObjectOutputStream out = new ObjectOutputStream(new
        FileOutputStream(filename))) {
            out.writeObject(student);
            System.out.println("Student object has been serialized and saved to file.");
        } catch (IOException e) {
            System.out.println("Error during serialization: " + e.getMessage());
        }
    }

    public static Student deserializeStudent(String filename) {
        try (ObjectInputStream in = new ObjectInputStream(new
        FileInputStream(filename))) {
            System.out.println("Student object has been deserialized.");
            return (Student) in.readObject();
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
        } catch (IOException e) {
            System.out.println("Error during deserialization: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.out.println("Error: Class not found.");
        }
    }
    return null;
}

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

    // Creating a student object
    Student student = new Student(1, "John Doe", 3.75);
```

```
// Serializing the student object
serializeStudent(student, filename);

// Deserializing the student object
Student deserializedStudent = deserializeStudent(filename);

// Display student details if deserialization was successful
if (deserializedStudent != null) {
    System.out.println("Deserialized Student Details:");
    deserializedStudent.displayStudent();
}
}
```

3. Output:

```
Student object has been serialized and saved to file.
Student object has been deserialized.
Deserialized Student Details:
Student ID: 1, Name: John Doe, GPA: 3.75
```

Problem-3 (Hard)

1. Aim:

Menu-based Java application that allows you to add employee details, display all employees, and exit. The employee details will be stored in a file, and the program will read the file to display the stored employee information.

2. Implementation/Code:

```
//Prithvi Singh //22BCS12368
import java.io.*; import
java.util.*;

class Employee implements Serializable {
    private static final long serialVersionUID = 1L;
```

```
private int id;   private String name;   private
String designation;   private 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 displayEmployee() {
        System.out.println("Employee ID: " + id + ", Name: " + name +
            ", Designation: " + designation + ", Salary: " + salary);
    }
}

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

    // Method to add an employee   public static
void addEmployee() {   Scanner scanner =
new Scanner(System.in);
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();

        Employee employee = new Employee(id, name, designation, salary);
        saveEmployeeToFile(employee);
        System.out.println("Employee added successfully!");
    }
}
```

```
// Method to save an employee to file (serialization)    public
static void saveEmployeeToFile(Employee employee) {
    List<Employee> employees = readEmployeesFromFile(); // Read existing
employees    employees.add(employee); // Add new employee

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

// Method to read employees from file (deserialization)
public static List<Employee> readEmployeesFromFile() {
    List<Employee> employees = new ArrayList<>();
    File file = new File(FILE_NAME);

    if (!file.exists()) {        return employees; // Return
empty list if file doesn't exist
    }

    try (ObjectInputStream in = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
        employees = (List<Employee>) in.readObject();
    } catch (EOFException e) {
        // End of file reached (no employees)
    } catch (IOException | ClassNotFoundException e) {
        System.out.println("Error reading employees: " + e.getMessage());
    }
    return employees;
}

// Method to display all employees
public static void displayAllEmployees() {
    List<Employee> employees = readEmployeesFromFile();
    if (employees.isEmpty()) {
        System.out.println("No employees found.");
    }
}
```

```
        } else {
            System.out.println("Employee Details:");
            for (Employee emp : employees) {
                emp.displayEmployee();
            }
        }
    }

    public static void main(String[] args) {
        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 choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    addEmployee();
                    break;
                case 2:
                    displayAllEmployees();
                    break;
                case 3:
                    System.out.println("Exiting program.");
                    return;
                default:
                    System.out.println("Invalid choice. Please enter 1, 2, or 3.");
            }
        }
    }
}
```

3. Output:



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
1. Add Employee
2. Display All Employees
3. Exit
Enter choice: 1
Enter Employee ID: 101
Enter Employee Name: John Doe
Enter Designation: Software Engineer
Enter Salary: 50000
Employee added successfully!

1. Add Employee
2. Display All Employees
3. Exit
Enter choice: 2
Employee Details:
Employee ID: 101, Name: JHON, Designation: Manager, Salary: 50000.0
Employee ID: 101, Name: John Doe, Designation: Software Engineer, Salary: 50000.0
```