Experiment 5

Student Name: Mayank Mani
UID: 22BET10076
Branch: BE-IT
Section/Group: IOT-701/A
Semester: 6th
Date of Performance: 18/02/2025
Subject Name: PBLJ
Subject Code: 22ITH-259

Problem: 1

1. **Aim:** To implement a Java program that calculates the sum of a list of integers using autoboxing and unboxing while parsing strings into wrapper classes.

2. **Objective:**
   - To understand and apply autoboxing and unboxing in Java.
   - To implement data parsing using Java wrapper classes.
3. **Implementation:**

```
package com.studyopedia;

import java.util.*;

public class Project5_easy {

    public static List<Integer> parseStringToIntegers(List<String> strNumbers) {

        List<Integer> intNumbers = new ArrayList<>();

        for (String str : strNumbers) {

            try {

                intNumbers.add(Integer.parseInt(str));

            } catch (NumberFormatException e) {

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

            }

        }

        return intNumbers;

    }

    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);
        System.out.println("Enter numbers separated by spaces:");
        String input = scanner.nextLine();
        scanner.close();
        List<String> strNumbers = Arrays.asList(input.split(" "));
        List<Integer> numbers = parseStringToIntegers(strNumbers);
        int sum = calculateSum(numbers);
        System.out.println("Sum of numbers: " + sum);
    }
}
```

4. Output

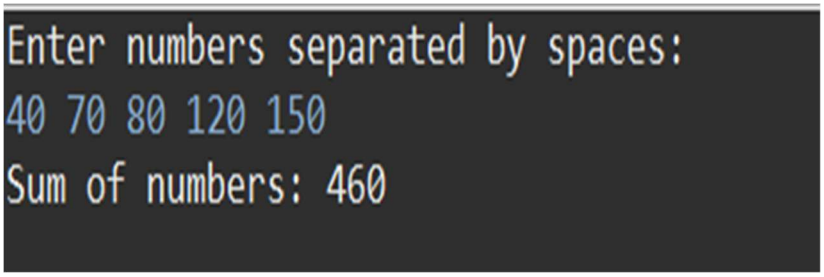

Fig. 1 Output of the Problem 1

5. Learning Outcome

- Ability to convert string inputs into integer values using Integer.parseInt().
- Understanding of how autoboxing and unboxing work in Java for primitive and wrapper class conversions.

Problem: 2

1. **Aim:** To develop a Java program that serializes a Student object (containing ID, name, and GPA) and saves it to a file, then deserializes and retrieves the data.

2. **Objective:**
   - To implement Java object serialization and deserialization using the Serializable interface.
   - To handle exceptions such as FileNotFoundException, IOException, and ClassNotFoundException effectively.
3. **Implementation:**

```
package com.studyopedia;

import java.io.*;

import java.util.Scanner;

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 display() {

      System.out.println("Student ID: " + id);

      System.out.println("Name: " + name);

      System.out.println("GPA: " + gpa);

   }
```

```
}
public class Project5_medium {
    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 during serialization: " + e.getMessage());
        }
    }
    public static Student deserializeStudent() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILE_NAME)))
{
            return (Student) ois.readObject();
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("Error during deserialization: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.out.println("Class not found: " + e.getMessage());
        }
        return null;
    }
    public static void main(String[] args) {
         Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Student ID: ");
```

```
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Enter Student Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Student GPA: ");
        double gpa = scanner.nextDouble();
        scanner.close();
        Student student = new Student(id, name, gpa);
        serializeStudent(student);
        Student deserializedStudent = deserializeStudent();
        if (deserializedStudent != null) {
            System.out.println("\nDeserialized Student Details:");
            deserializedStudent.display();
        }
    }
}
```

4. Output

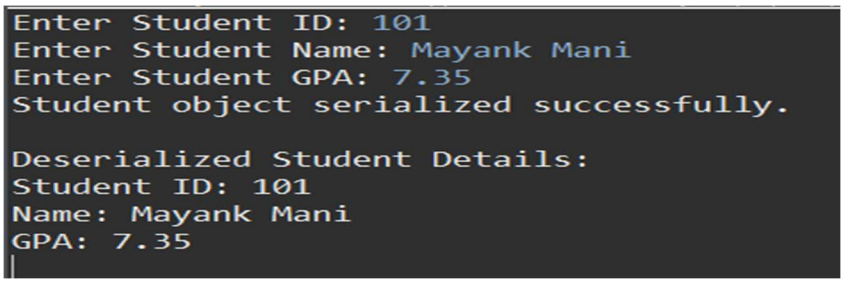

Fig. 2 Output of the Problem 2

5. Learning Outcome

- Understanding of object persistence through serialization and deserialization in Java.
- Ability to handle file operations and exceptions related to file input/output.

Question: 3

1. **Aim:** To create a menu-driven Java application that allows users to add employee details, save them to a file, and display all stored employee records.

2. **Objective:**
   - To implement a file-based employee management system with CRUD operations.
   - To use Java collections and file handling for storing and retrieving employee details.
3. **Implementation:**

```
package com.studyopedia;

import java.io.*;

import java.util.Scanner;

public class Project5_hard {

    private static final String FILE_NAME = "employees.txt";

    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {

        boolean running = true;

        while (running) {

            printMenu();

            int choice = getChoice();

            switch (choice) {

                case 1:

                    addEmployee();

                    break;

                case 2:

                    displayEmployees();

                    break;

                case 3:

                    running = false;
```

```
                System.out.println("Exiting application...");
                break;
            default:
                System.out.println("Invalid option! Please try again.");
        }
    }
    scanner.close();
}
private static void printMenu() {
    System.out.println("\n--- Employee Management System ---");
    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: ");
}
private static int getChoice() {
    while (!scanner.hasNextInt()) {
        System.out.print("Invalid input! Please enter a number (1-3): ");
        scanner.next();
    }
    int choice = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    return choice;
}
private static void addEmployee() {
    System.out.println("\nEnter Employee Details:");
    System.out.print("Name: ");
```

```
    String name = scanner.nextLine();
    System.out.print("Employee ID: ");
    String id = scanner.nextLine();
    System.out.print("Designation: ");
    String designation = scanner.nextLine();
    System.out.print("Salary: ");
    String salary = scanner.nextLine();
    try (PrintWriter writer = new PrintWriter(new FileWriter(FILE_NAME, true))) {
        writer.printf("%s,%s,%s,%s%n", name, id, designation, salary);
        System.out.println("Employee added successfully!");
    } catch (IOException e) {
        System.out.println("Error saving employee data: " + e.getMessage());
    }
}
private static void displayEmployees() {
    System.out.println("\n--- Employee List ---");
    try (BufferedReader reader = new BufferedReader(new FileReader(FILE_NAME))) {
        String line;
        int count = 1;
        while ((line = reader.readLine()) != null) {
            String[] data = line.split(",");
            if (data.length == 4) {
                System.out.println("Employee " + count++ + ":");
                System.out.println("Name: " + data[0]);
                System.out.println("ID: " + data[1]);
                System.out.println("Designation: " + data[2]);
                System.out.println("Salary: " + data[3]);
```

```
                System.out.println("----------------------");
            }
        }
        if (count == 1) {
            System.out.println("No employees found!");
        }
    } catch (FileNotFoundException e) {
        System.out.println("File Not found!");
    } catch (IOException e) {
        System.out.println("Error reading employee data: " + e.getMessage());
    }
  }
}
```

4. **Output**

```
Name: Mayank Mani
Employee ID: 101
Designation: Developer
Salary: 150000
Employee added successfully!

--- Employee Management System ---
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1

Enter Employee Details:
Name: Rajni
Employee ID: 102
Designation: Designer
Salary: 150000
Employee added successfully!

--- Employee Management System ---
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 2

--- Employee List ---
Employee 1:
Name: Mayank Mani
ID: 101
Designation: Developer
Salary: 150000
----------------------
Employee 2:
Name: Rajni
ID: 102
Designation: Designer
Salary: 150000
```

Fig. 3 Output of the Problem 3

5. **Learning Outcome**

- Ability to store and retrieve structured data using file handling in Java.
- Understanding of menu-driven programs and exception handling for user-friendly applications.