WORKSHEET-5

Student Name: Meenansh Gupta **UID:** 22BCS16380

Branch: CSE **Section/Group:** NTPP-603-B

Semester: 6th **Date of Performance:** 19/3/25

Subject Name: PBLJ **Subject Code:** 22CSH-359

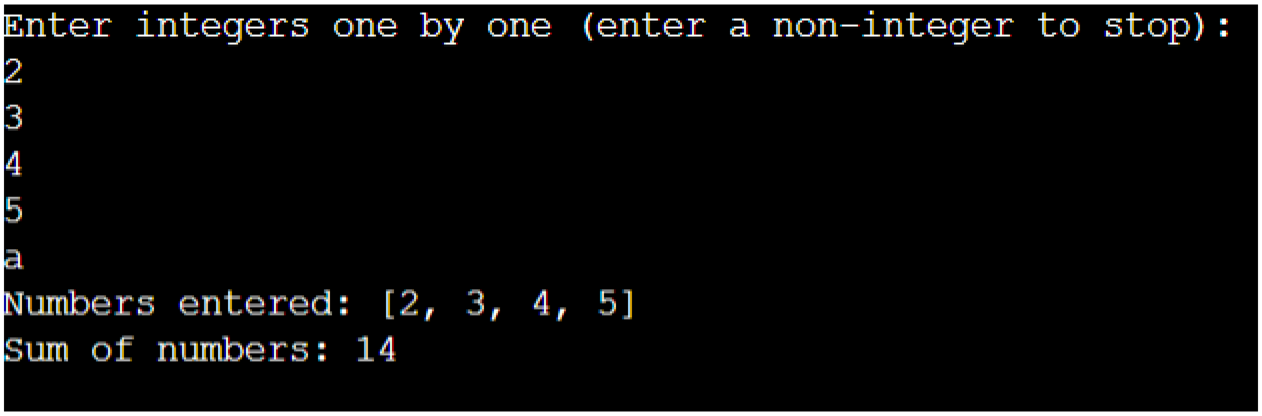
Aim(i): *Easy Level: Write a Java program to calculate the sum of a list of integers using autoboxing and unboxing. Include methods to parse strings into their respective wrapper classes (e.g., Integer.parseInt()).*

Source Code:

```
import java.util.*;
import java.util.stream.Collectors;

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

      System.out.println("Enter integers one by one (enter a non-integer to stop):");

      while (scanner.hasNextInt()) { // Reads only integer inputs
         numbers.add(scanner.nextInt()); // Autoboxing
      }

      // Using Stream API to calculate sum
      int sum = numbers.stream().mapToInt(Integer::intValue).sum();

      System.out.println("Numbers entered: " + numbers);
```

```
                System.out.println("Sum of numbers: " + sum);

                scanner.close();
            }
        }
```

OUTPUT:



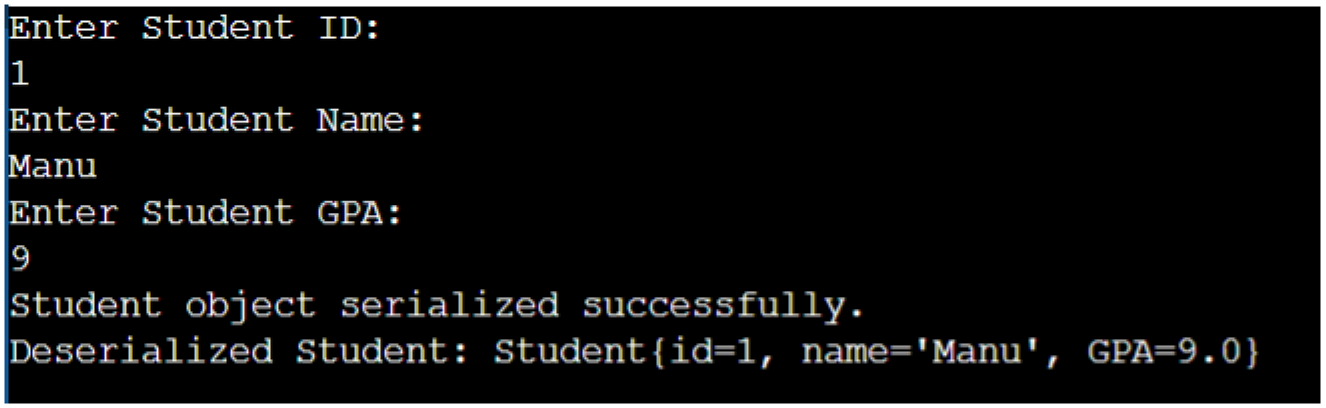
```
Enter integers one by one (enter a non-integer to stop):
2
3
4
5
a
Numbers entered: [2, 3, 4, 5]
Sum of numbers: 14
```

Aim(ii): *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.*

Source Code:

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

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;
    }

    @Override
    public String toString() {
        return "Student{id=" + id + ", name='" + name + "', GPA=" + gpa + "}";
    }
}

public class StudentSerialization {
    private static final String FILE_NAME = "student.ser";

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Taking input from user
        System.out.println("Enter Student ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.println("Enter Student Name: ");
        String name = scanner.nextLine();
        System.out.println("Enter Student GPA: ");
        double gpa = scanner.nextDouble();

        Student student = new Student(id, name, gpa);

        // Serialization
                try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            oos.writeObject(student);
            System.out.println("Student object serialized successfully.");
        } catch (IOException e) {
            e.printStackTrace();
        }

        // Deserialization
                try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            Student deserializedStudent = (Student) ois.readObject();
            System.out.println("Deserialized Student: " + deserializedStudent);
        } catch (FileNotFoundException e) {
            System.out.println("File not found: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } catch (ClassNotFoundException e) {
            System.out.println("Class not found: " + e.getMessage());
        }
        scanner.close();
    }
}
```

OUTPUT:



```
Enter Student ID:
1
Enter Student Name:
Manu
Enter Student GPA:
9
Student object serialized successfully.
Deserialized Student: Student{id=1, name='Manu', GPA=9.0}
```

Aim(iii):

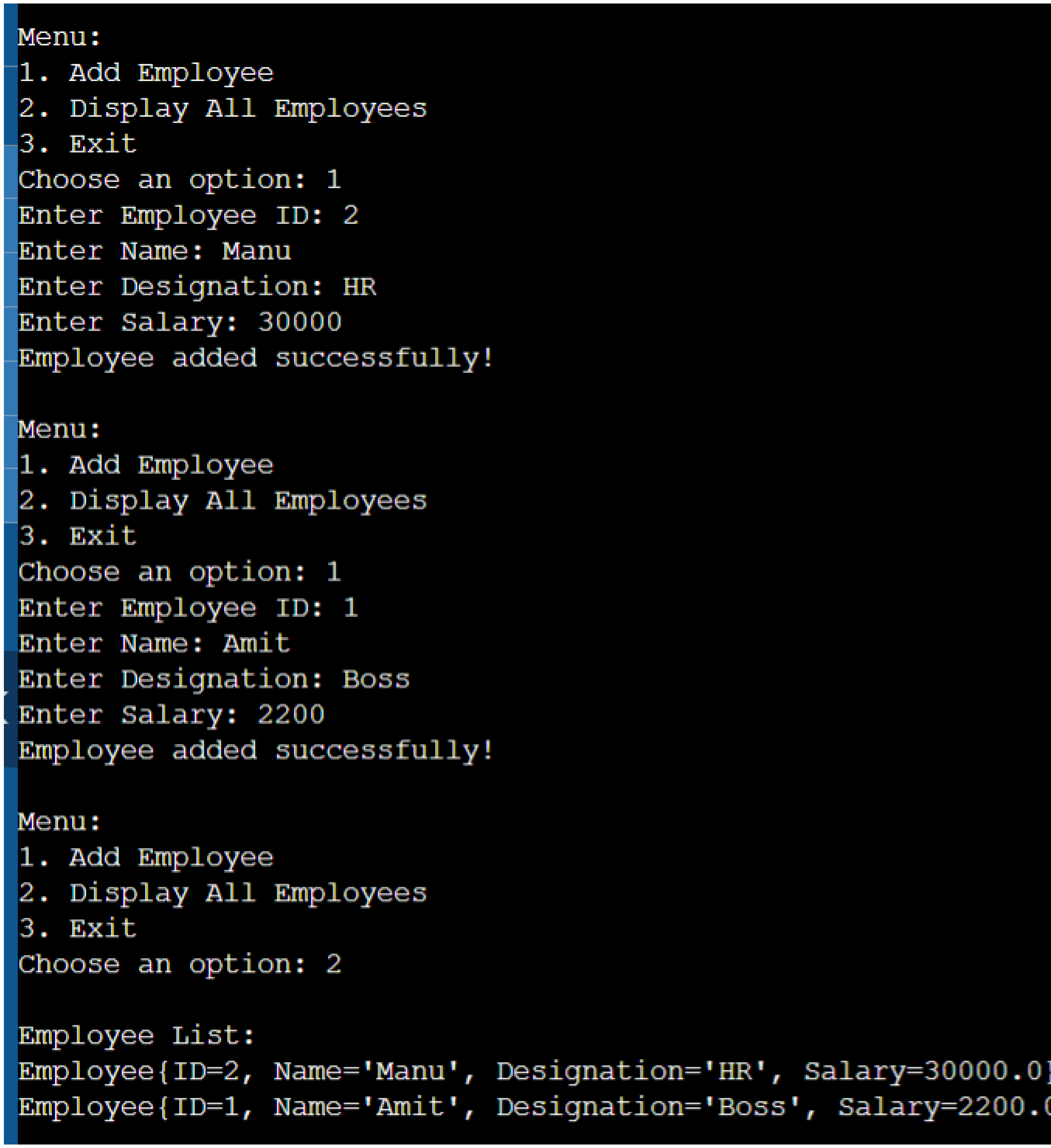
Hard Level: Create a menu-based Java application with the following options. 1.Add an Employee 2. Display All 3. Exit If option 1 is selected, the application should gather details of the employee like employee name, employee id, designation and salary and store it in a file. If option 2 is selected, the application should display all the employee details. If option 3 is selected the application should exit.

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

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

    public Employee(String empId, String name, String designation, double salary) {
        this.empId = empId;
        this.name = name;
        this.designation = designation;
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee{ID=" + empId + ", Name='" + name + "', Designation='" +
designation + "', Salary=" + salary + "}";
    }
}

public class EmployeeManagement {
    private static final String FILE_NAME = "employees.dat";
    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("\nMenu:");
        System.out.println("1. Add Employee");
        System.out.println("2. Display All Employees");
        System.out.println("3. Exit");
        System.out.print("Choose an option: ");
        int choice = scanner.nextInt();
        scanner.nextLine(); // Consume newline

        switch (choice) {
            case 1:
                addEmployee(scanner);
                break;
            case 2:
                displayEmployees();
                break;
            case 3:
                saveEmployees(); // Save employees before exiting
                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 Employee ID: ");
    String empId = 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();
        scanner.nextLine(); // Consume newline

        Employee employee = new Employee(empId, name, designation, salary);
        employees.add(employee);
        saveEmployees(); // Save immediately after adding
        System.out.println("Employee added successfully!");
    }

    private static void displayEmployees() {
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
            return;
        }
        System.out.println("\nEmployee List:");
        employees.forEach(System.out::println);
    }

    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());
        }
    }

    @SuppressWarnings("unchecked")
    private static void loadEmployees() {
        try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            employees = (List<Employee>) ois.readObject();
        } catch (FileNotFoundException e) {
            // No previous data, so ignore
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error loading employees: " + e.getMessage());
        }
    }
}
```

OUTPUT:



```
Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 1
Enter Employee ID: 2
Enter Name: Manu
Enter Designation: HR
Enter Salary: 30000
Employee added successfully!

Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 1
Enter Employee ID: 1
Enter Name: Amit
Enter Designation: Boss
Enter Salary: 2200
Employee added successfully!

Menu:
1. Add Employee
2. Display All Employees
3. Exit
Choose an option: 2

Employee List:
Employee{ID=2, Name='Manu', Designation='HR', Salary=30000.0
Employee{ID=1, Name='Amit', Designation='Boss', Salary=2200.
```

Learning Outcomes

1. We learnt about File Handling.
2. We learnt about Serialization.
3. We learnt about Autoboxing, Unboxing.