Experiment - 5

Student Name: Ankush Thakur **UID:** 22BCS11815
Branch: BE- CSE **Section/Group:** 22BCS_IOT-637 (A)
Semester: 6th **Date of Performance:** 21/2/2025
Subject Name: Project Based Learning in Java **Subject Code:** 22CSH-359

Problem - 1

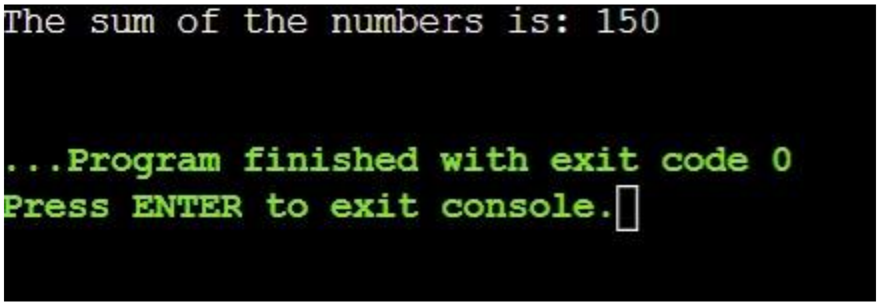
1. **Aim:** Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.

2. **Objective:** 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()).

3. **Implementation/Code:**

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

public class AutoboxingExample {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();

        // Example input numbers in String format
        String[] inputStrings = {"10", "20", "30", "40", "50"};

        // Parse strings into Integer objects and add to list
        for (String str : inputStrings) {
            numbers.add(parseInteger(str));
        }

        // Calculate sum using autoboxing and unboxing
        int sum = calculateSum(numbers);

        // Print result
        System.out.println("The sum of the numbers is: " + sum);
    }

    // Method to parse a string into an Integer object
    private static Integer parseInteger(String str) {
        return Integer.parseInt(str); // Autoboxing
    }

    // Method to calculate sum of a list of integers
    private static int calculateSum(List<Integer> numbers) {
        int sum = 0;
        for (Integer num : numbers) {
            sum += num; // Unboxing
        }
        return sum;
    }
}
```

4. Output:



```
The sum of the numbers is: 150


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

Problem – 2

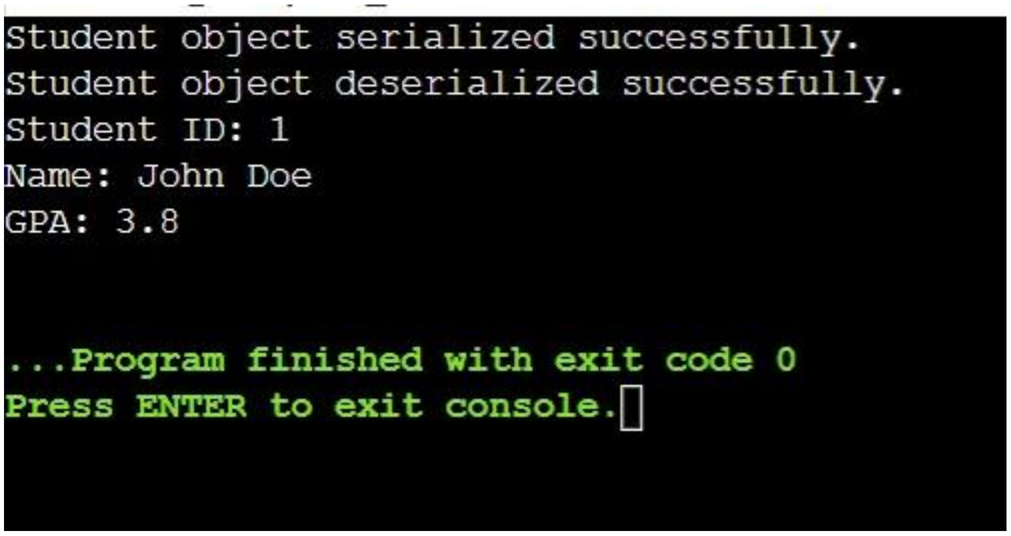
1. **Aim:** Develop Java programs using autoboxing, serialization, file handling, and efficient data processing and management.

2. **Objective:** Create a Java program to serialize and deserialize a Student object. The program should:
   a. Serialize a Student object (containing id, name, and GPA) and save it to a file.
   b. Deserialize the object from the file and display the student details.
   c. Handle FileNotFoundException, IOException, and ClassNotFoundException using exception handling.

3. **Implementation/Code:**

```
import java.util.*;
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);
        System.out.println("Name: " + name);
        System.out.println("GPA: " + gpa);
    }
}

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

    public static void main(String[] args) {
        Student student = new Student(1, "John Doe", 3.8);
        serializeStudent(student);
        Student deserializedStudent = deserializeStudent();

        if (deserializedStudent != null) {
            deserializedStudent.displayStudent();
        }
    }

    private 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.err.println("Error     during     serialization:     "     +
e.getMessage());
        }
    }

    private static Student deserializeStudent() {
        try    (ObjectInputStream    ois    =    new    ObjectInputStream(new
FileInputStream(FILE_NAME))) {
            System.out.println("Student object deserialized successfully.");
            return (Student) ois.readObject();
        } catch (FileNotFoundException e) {
            System.err.println("File not found: " + e.getMessage());
        } catch (IOException e) {
            System.err.println("Error    during    deserialization:    "    +
e.getMessage());
        } catch (ClassNotFoundException e) {
            System.err.println("Class not found: " + e.getMessage());
        }
        return null;
    }
}
```

4. Output:



```
Student object serialized successfully.
Student object deserialized successfully.
Student ID: 1
Name: John Doe
GPA: 3.8


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

Problem – 3

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

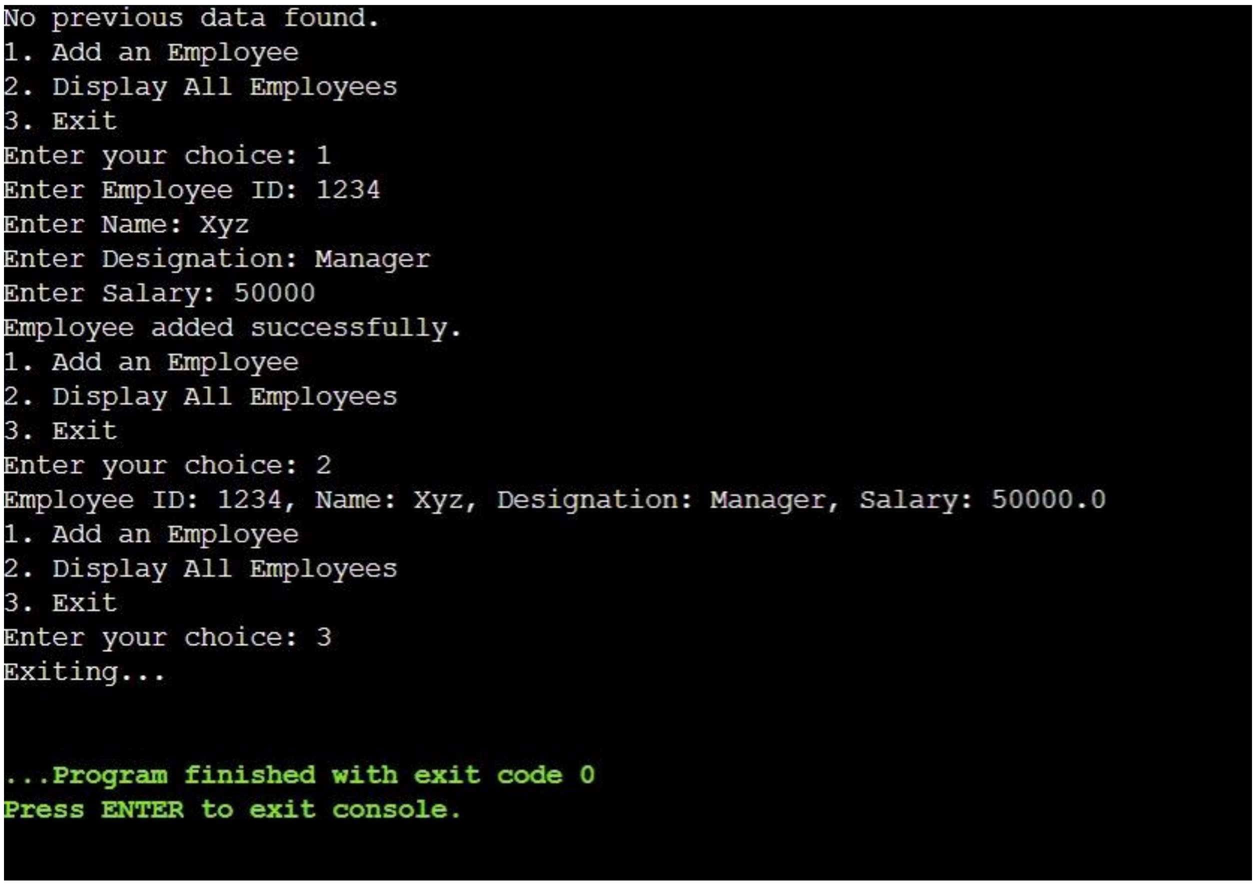
2. **Objective:** Create a menu-based Java application with the following options. 1.Add an Employee 2. Display All 3. Exit If option 1 is selected, the application should gather details of the employee like employee name, employee id, designation and salary and store it in a file. If option 2 is selected,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

3. **Implementation/Code:**

```
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;
    }

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

public class EmployeeManagement {
    private static final String FILE_NAME = "employees.ser";
    private static List<Employee> employees = new ArrayList<>();

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

```
            do {
                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: ");
                choice = scanner.nextInt();
                scanner.nextLine();
                switch (choice) {
                    case 1:
                        addEmployee(scanner);
                        break;
                    case 2:
                        displayEmployees();
                        break;
                    case 3:
                        saveEmployees();
                        System.out.println("Exiting...");
                        break;
                    default:
                        System.out.println("Invalid choice! Please try again.");
                }
            } while (choice != 3);
            scanner.close();
        }

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

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

        private static void displayEmployees() {
            if (employees.isEmpty()) {
                System.out.println("No employees found.");
            } else {
                for (Employee emp : employees) {
                    System.out.println(emp);
                }
            }
        }

        private static void saveEmployees() {
            try    (ObjectOutputStream    oos    =    new    ObjectOutputStream(new
    FileOutputStream(FILE_NAME))) {
            oos.writeObject(employees);
        } catch (IOException e) {
            System.err.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) {
            System.out.println("No previous data found.");
        } catch (IOException | ClassNotFoundException e) {
            System.err.println("Error loading employees: " + e.getMessage());
        }
    }
}
```

4. Output:



```
No previous data found.
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 1
Enter Employee ID: 1234
Enter Name: Xyz
Enter Designation: Manager
Enter Salary: 50000
Employee added successfully.
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 2
Employee ID: 1234, Name: Xyz, Designation: Manager, Salary: 50000.0
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 3
Exiting...


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

Learning Outcomes:

i. I learned how to serialize and deserialize a Java object.
ii. I successfully saved a Student object containing id, name, and GPA to a file.
iii. I deserialized the Student object from the file and displayed its details.
iv. I handled FileNotFoundException, IOException, and ClassNotFoundException using exception handling.
v. I used autoboxing and unboxing to calculate the sum of a list of integers.
vi. I converted string representations of numbers into their respective wrapper classes using methods like Integer.parseInt().
vii. I built a menu-driven Java application with options to add an employee, display all employees, and exit.
viii. I gathered and stored employee details such as employee name, employee id, designation, and salary in a file.
ix. I retrieved and displayed all stored employee details when requested.
x. I ensured the application exited when the exit option was selected.