Experiment- 5

Student Name: Tejveer Singh **UID: 22BCS16439**
Branch: BE-CSE **Section/Group: IOT_631-A**
Semester:6th **Date of Performance: 28/02/2025**
Subject Name: Project Based Learning in Java with Lab **Subject Code: 22CSH-359**

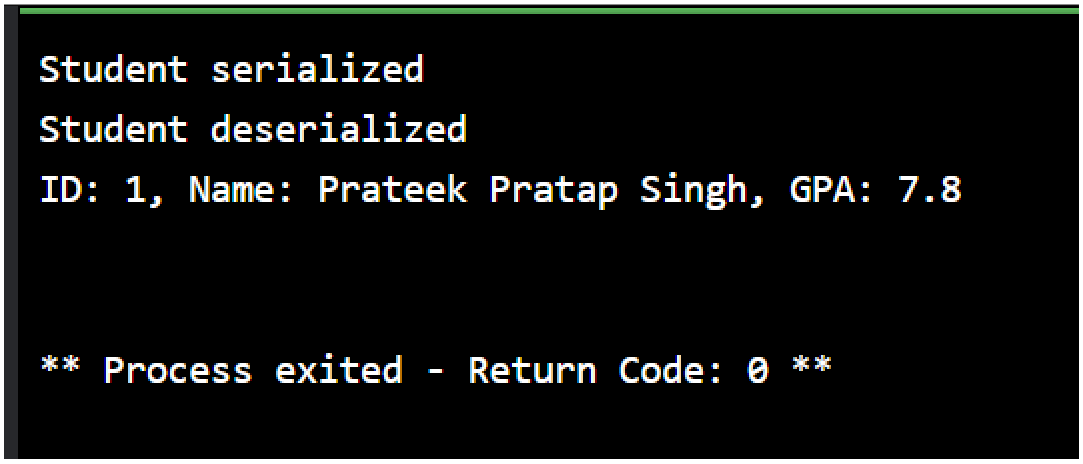
1. **(a) Aim:** Create a Java program to serialize and deserialize a Student object. The program should:
   Serialize a Student object (containing id, name, and GPA) and save it to a file.
   Deserialize the object from the file and display the student details.
   Handle FileNotFoundException, IOException,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

2. **Objective:**
   - To demonstrate object serialization and deserialization in Java.
   - To save a Student object (containing id, name, and GPA) to a file using ObjectOutputStream.
   - To retrieve the saved object from the file using ObjectInputStream and display its details.
   - To handle exceptions such as IOException and ClassNotFoundException.

3. **Implementation:**

```
import java.io.*;

class Student implements Serializable {
    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("ID: " + id + ", Name: " + name + ", GPA: " + gpa);
    }
}

public class Main {
    public static void main(String[] args) {
        String filename = "student.ser";
        try {
            Student student = new Student(1, "Prateek Pratap Singh", 7.8);
            ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream(filename));
            out.writeObject(student);
            out.close();
            System.out.println("Student serialized");
        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }

        try {
            ObjectInputStream in = new ObjectInputStream(new
FileInputStream(filename));
            Student student = (Student) in.readObject();
            in.close();
            System.out.println("Student deserialized");
            student.display();
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

4. Output:



```
Student serialized
Student deserialized
ID: 1, Name: Prateek Pratap Singh, GPA: 7.8


** Process exited - Return Code: 0 **
```

5. Learning Outcomes :

- Understand serialization and deserialization in Java.
- Learn how to use ObjectOutputStream and ObjectInputStream for object persistence.
- Gain hands-on experience with file handling (FileOutputStream and FileInputStream).
- Learn to handle exceptions (IOException and ClassNotFoundException) in Java.
- Understand how object state can be saved and restored from a file.

1. (b) Aim: 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.

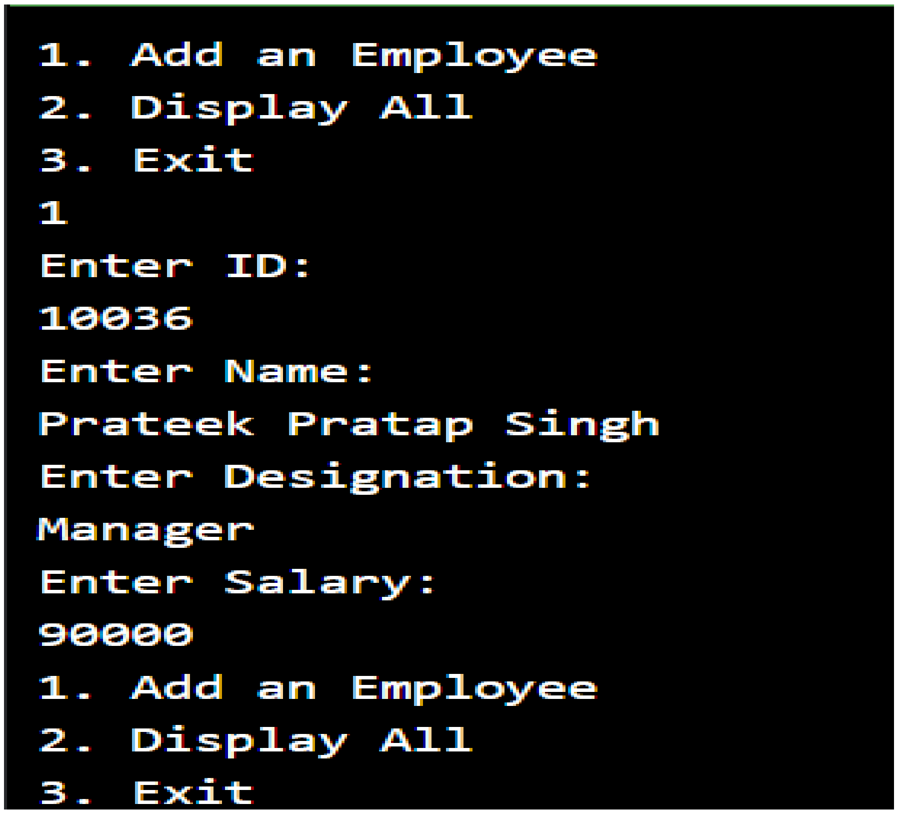
2. Objective :

- To create a menu-driven Java application that allows users to:
  - Add Employee details (id, name, designation, salary).
  - Display all Employees stored in an ArrayList.
  - Exit the program when the user chooses.
- To implement basic user interaction using Scanner for input handling.
- To demonstrate dynamic data storage using ArrayList without file operations.

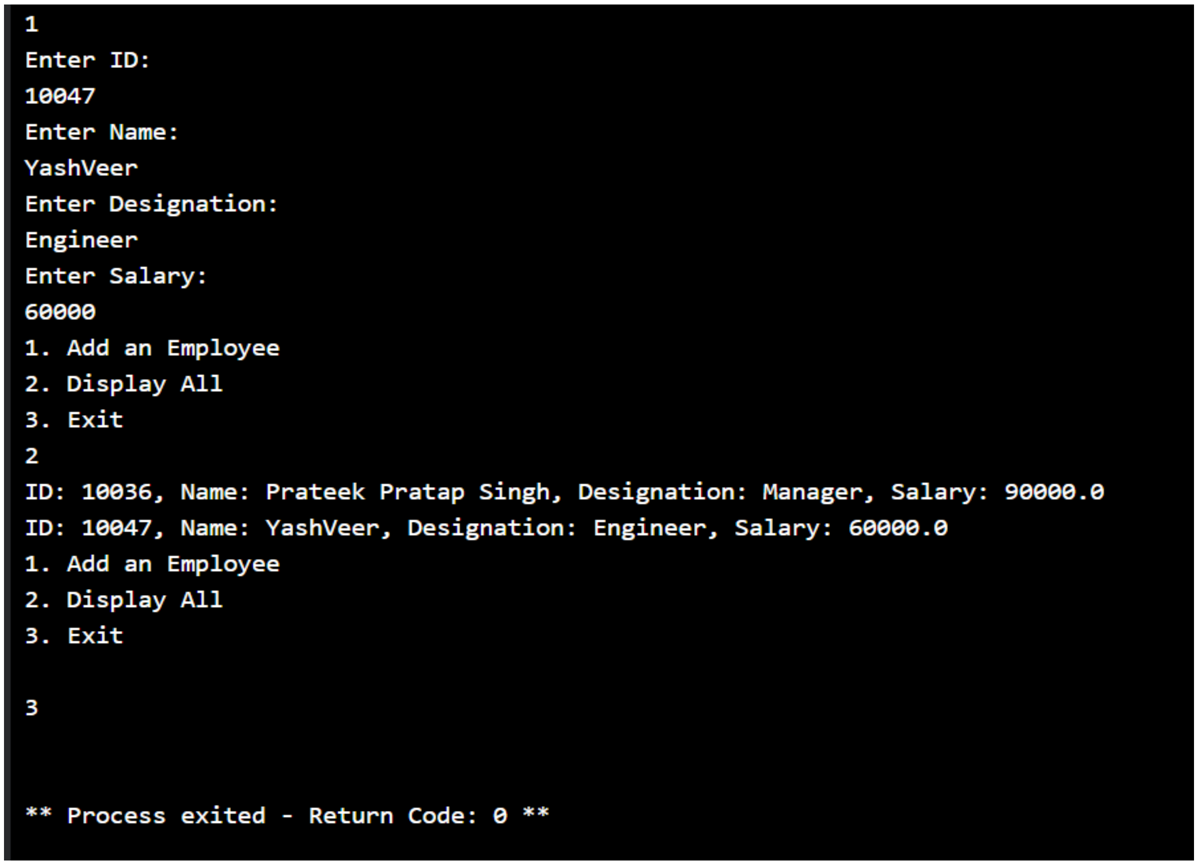
3. Implementation :

```
import java.util.*;

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

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        List<Employee> employees = new ArrayList<>();
        while (true) {
            System.out.println("1. Add an Employee\n2. Display All\n3. Exit");
            int choice = scanner.nextInt();
            scanner.nextLine();
            if (choice == 1) {
                System.out.print("Enter 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));
            } else if (choice == 2) {
                for (Employee e : employees) e.display();
            } else if (choice == 3) {
                break;
            }
        }
        scanner.close();
    }
    }
```

4. **Output :**



```
1. Add an Employee
2. Display All
3. Exit
1
Enter ID:
10036
Enter Name:
Prateek Pratap Singh
Enter Designation:
Manager
Enter Salary:
90000
1. Add an Employee
2. Display All
3. Exit
```



```
1
Enter ID:
10047
Enter Name:
YashVeer
Enter Designation:
Engineer
Enter Salary:
60000
1. Add an Employee
2. Display All
3. Exit
2
ID: 10036, Name: Prateek Pratap Singh, Designation: Manager, Salary: 90000.0
ID: 10047, Name: YashVeer, Designation: Engineer, Salary: 60000.0
1. Add an Employee
2. Display All
3. Exit

3


** Process exited - Return Code: 0 **
```

5. **Learning Outcomes:**
   - Learn how to create a menu-driven console application in Java.
   - Gain experience in taking user input using Scanner.
   - Understand how to store and manage multiple objects using ArrayList.
   - Develop skills in object-oriented programming (OOP) concepts like encapsulation and constructors.
   - Improve logical thinking for implementing control structures (if-else, while loop).