



Experiment 5.3

Student Name: Akshat Srivastava

UID: 22BCS11740

Branch: BE CSE

Section/Group: 22BCS_IOT_618_A

Semester: 6th

DoP: 21/02/2025

Subject Name: PBLJ Lab

Subject Code: 22CSH-359

1. **Aim:** To create a menu-based Java application that allows adding employee details, displaying all employees, and exiting the application, with employee data stored and retrieved from a file using serialization and deserialization.

2. Objective:

- Design an `Employee` class with name, id, designation, and salary fields.
- Implement a menu with options to add employees, display all employees, and exit the program.
- Store employee data in a file using `ObjectOutputStream` in append mode.
- Retrieve and display employee data using `ObjectInputStream`.
- Handle exceptions related to file input and output operations.

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;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

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

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

```
    public static void addEmployee() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        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);
        employees.add(employee);
        saveEmployees();
        System.out.println("Employee added successfully!");
    }
```

```
    public static void displayAllEmployees() {
        loadEmployees();
        if (employees.isEmpty()) {
            System.out.println("No employees found.");
        } else {
            for (Employee employee : employees) {
                System.out.println(employee);
            }
        }
    }

    private static void saveEmployees() {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
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) {
        employees = new ArrayList<>();
    } catch (IOException | ClassNotFoundException e) {
        System.err.println("Error loading employees: " + e.getMessage());
    }
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    while (true) {
        System.out.println("\nEmployee 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: ");
        int choice = scanner.nextInt();
        scanner.nextLine();

        switch (choice) {
            case 1:
                addEmployee();
                break;
            case 2:
                displayAllEmployees();
                break;
            case 3:
                System.out.println("Exiting...");
        }
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        return;  
    default:  
        System.out.println("Invalid choice! Please try again.");  
    }  
}  
}  
}
```

4. Output

```
Employee Management System  
1. Add an Employee  
2. Display All Employees  
3. Exit  
Enter your choice: 1  
Enter Employee ID: 132  
Enter Employee Name: Anwar  
Enter Designation: HR  
Enter Salary: 75000  
Employee added successfully!  
  
Employee Management System  
1. Add an Employee  
2. Display All Employees  
3. Exit  
Enter your choice: 1  
Enter Employee ID: 125  
Enter Employee Name: Vedant  
Enter Designation: Director  
Enter Salary: 100000  
Employee added successfully!  
  
Employee Management System  
1. Add an Employee  
2. Display All Employees  
3. Exit  
Enter your choice: 2  
Employee ID: 132, Name: Anwar, Designation: HR, Salary: 75000.0  
Employee ID: 125, Name: Vedant, Designation: Director, Salary: 100000.0
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

5. Learning Outcome:

- Understand file handling and serialization in Java to store and retrieve objects persistently.
- Learn how to implement a menu-driven console application using loops and conditional statements.
- Gain experience in object-oriented programming (OOP) by defining and managing Employee objects.