Experiment4

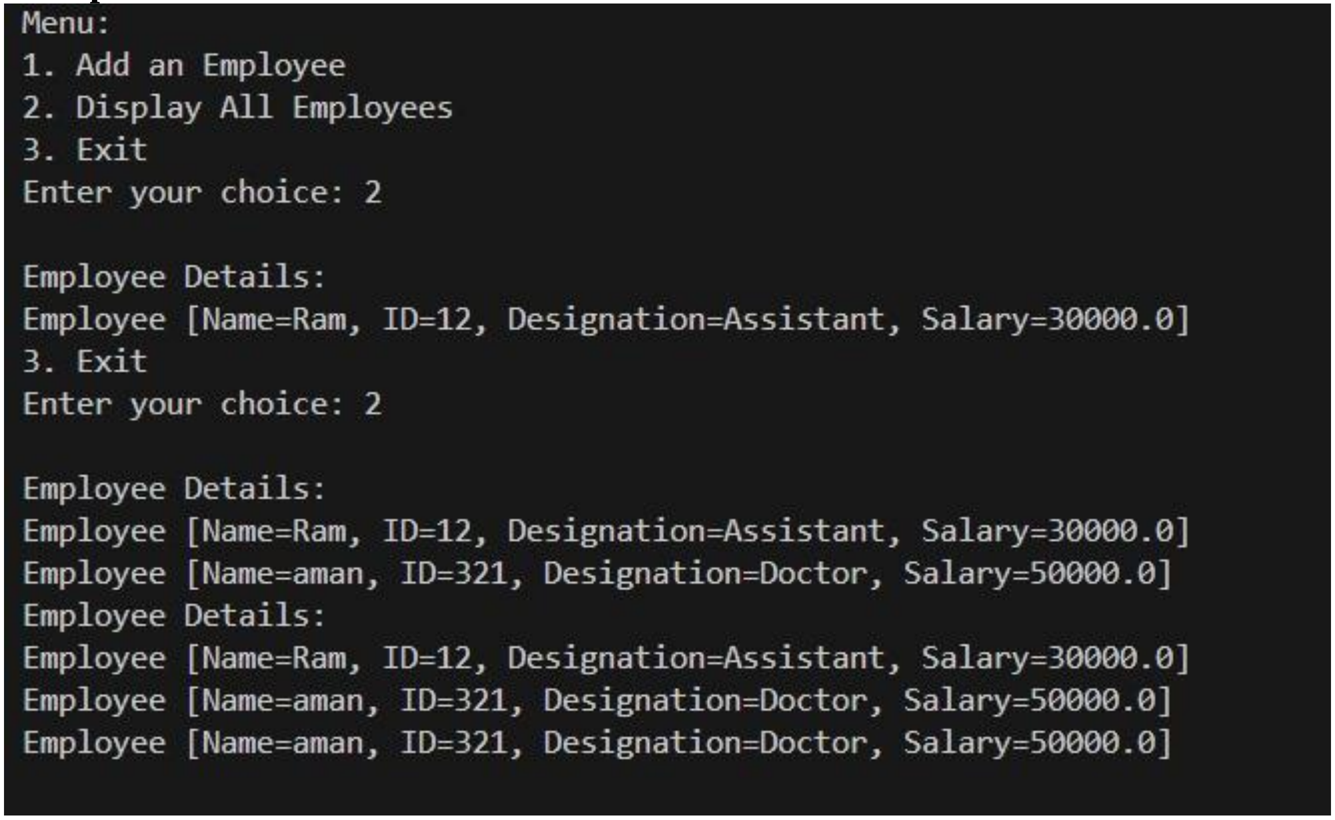
Student Name: Shivang Mehla **UID: 22BCS10748**
Branch :BE-CSE **Section/Group:643-B**
Semester:6th **Date of Performance:20/02/25**
Subject Name: PBLJ **Subject Code:22CSH-359**

1. **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. Code

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

public class EmployeeManagementApp {
    private static final String FILE_NAME = "employees.dat";
    private List<Employee> employees;

    public EmployeeManagementApp() {
        employees = new ArrayList<>();
        loadEmployees();
    }

    public void addEmployee(Employee employee) {
        employees.add(employee);
        saveEmployees();
    }

    public List<Employee> getAllEmployees() {
        return employees;
    }

    private void saveEmployees() {
        try (ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(FILE_NAME))) {
            oos.writeObject(employees);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    @SuppressWarnings("unchecked")
    private void loadEmployees() {
        File file = new File(FILE_NAME);
        if (file.exists()) {
            try (ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(FILE_NAME))) {
                employees = (List<Employee>) ois.readObject();
            } catch (IOException | ClassNotFoundException e) {
                e.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        EmployeeManagementApp management = new EmployeeManagementApp();
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nMenu:");
            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(); // Consume newline

            switch (choice) {
                case 1:
                    System.out.print("Enter Employee Name: ");
                    String name = scanner.nextLine();
                    System.out.print("Enter Employee ID: ");
                    int id = scanner.nextInt();
                    scanner.nextLine(); // Consume newline
                    System.out.print("Enter Employee Designation: ");
                    String designation = scanner.nextLine();
                    System.out.print("Enter Employee Salary: ");
                    double salary = scanner.nextDouble();
                    scanner.nextLine(); // Consume newline

                    Employee employee = new Employee(name, id, designation, salary);
                    management.addEmployee(employee);
                    System.out.println("Employee added successfully.");
                    break;
                case 2:
                    List<Employee> employees = management.getAllEmployees();
                    System.out.println("\nEmployee Details:");
                    for (Employee emp : employees) {
                        System.out.println(emp);
                    }
                    break;
                case 3:
                    System.out.println("Exiting...");
                    scanner.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }
}

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

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

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public String getDesignation() {
        return designation;
    }

    public double getSalary() {
        return salary;
    }

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

3.Output:



```
Menu:
1. Add an Employee
2. Display All Employees
3. Exit
Enter your choice: 2

Employee Details:
Employee [Name=Ram, ID=12, Designation=Assistant, Salary=30000.0]
3. Exit
Enter your choice: 2

Employee Details:
Employee [Name=Ram, ID=12, Designation=Assistant, Salary=30000.0]
Employee [Name=aman, ID=321, Designation=Doctor, Salary=50000.0]
Employee Details:
Employee [Name=Ram, ID=12, Designation=Assistant, Salary=30000.0]
Employee [Name=aman, ID=321, Designation=Doctor, Salary=50000.0]
Employee [Name=aman, ID=321, Designation=Doctor, Salary=50000.0]
```

Learning Outcomes

- Understand how to use maps(dictionaries)for efficient data storage and retrieval.

- Learn to group and organized at a based on a key attribute.

Gain experience in handling user input and storing objects dynamicall