Experiment No: 2.4

Student Name: Kamal Ale Magar **UID: 21BCS10155**
Branch: CSE **Section/Group: 616/B**
Semester: 6th **Date of Performance: 04/03/24**
Subject: Project Based Learning in Java with Lab **Subject Code: 21CSH-319**

1. Aim:

Employee Management System Createa 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 likeemployee 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.

Task:

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.

Sample Output: Main Menu
```
1. Add an Employee
2. Display All
3. Exit
1
Enter Employee ID: 120
Enter Employee Name: Sudhir
Enter Employee Age:  33
Enter Employee Salary: 90000
Main Menu
1. Add an Employee
2. Display All
3. Exit
1
Enter Employee ID :130
Enter Employee Name :Selvan
Enter Employee Age :40
Enter Employee Salary :100000
Main Menu
1. Add an Employee
2. Display All
3. Exit
2
1
2
......Report........
120 Sudhir 33 90000.0
130 Selvan 40 100000.0
----End of Report-----
Main Menu
1. Add an Employee
2. Display All
3. Exit
3
Exiting the System.
```

2. Objectives:

- To learn about concept of File Handling in java.
- To learn about LinkedList, Exception Handling in java.

3. Input/Apparatus Used:

Hardware Requirements: - Minimum 384MB RAM, 100 GB hard Disk, processor with 2.1 MHz
Software Requirements: - Eclipse, NetBeans, IntelliJ, Online Java Compiler etc.

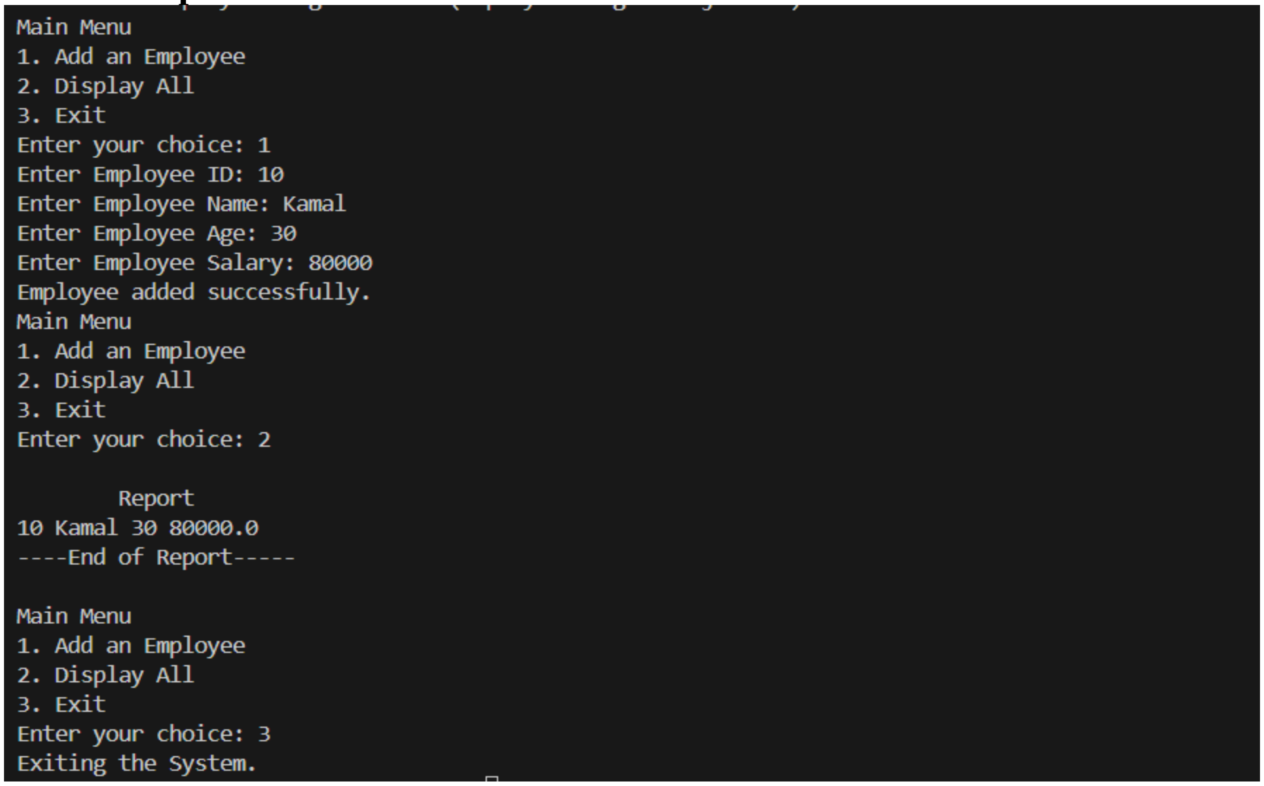
4. Procedure/Algorithm/Pseudocode:

```
Step1: Start execution.
Step2: Declare 4 ArrayList to store employee name, empoyee id, designation andsalary.
Step3: Using the constructor add values to the arraylist.
Step4: Make a display function to Display the contents of each arraylist using a forloop.
Step5: In main function take choices as input inside a switch statement.
Step6: Call the relevant functions as per the entered choices.
Step7: Stop execution.
```

5. Code:

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

class Employee implements Serializable {
    private int id;
    private String name;
    private int age;
    private double salary;

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

    @Override
    public String toString() {
        return id + " " + name + " " + age + " " + salary;
    }
}

public class EmployeeManagement {
    private static final String FILENAME = "employees.dat";
    private static final Scanner scanner = new Scanner(System.in);
    private static final List<Employee> employees = new ArrayList<>();

    public static void main(String[] args) {
        loadEmployees(); // Load existing employees from file
        int choice;
        do {
            System.out.println("Main Menu");
            System.out.println("1. Add an Employee");
            System.out.println("2. Display All");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    addEmployee();
                    break;
                case 2:
                    displayAllEmployees();
                    break;
                case 3:
                    saveEmployees();
                    System.out.println("Exiting the System.");
break;
                default:
                    System.out.println("Invalid choice. Please enter again.");
            }
        } while (choice != 3);
        scanner.close();
    }

    private static void addEmployee() {
        System.out.print("Enter Employee ID: ");
        int id = scanner.nextInt();
        scanner.nextLine(); // Consume newline
        System.out.print("Enter Employee Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Employee Age: ");
        int age = scanner.nextInt();
        System.out.print("Enter Employee Salary: ");
        double salary = scanner.nextDouble();

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

    private static void displayAllEmployees() {
        System.out.println("\n\tReport");
        for (Employee emp : employees) {
            System.out.println(emp);
        }
        System.out.println("----End of Report-----\n");
    }

    private static void loadEmployees() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(FILENAME))) {
            while (true) {
                employees.add((Employee) ois.readObject());
            }
        } catch (EOFException e) {
            // Reached end of file
        } catch (IOException | ClassNotFoundException e) {
            // Handle exceptions
            e.printStackTrace();
        }
    }

    private static void saveEmployees() {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILENAME))) {
            for (Employee emp : employees) {
                oos.writeObject(emp);
            }
        } catch (IOException e) {
            // Handle exceptions
                e.printStackTrace();


            }
              }
            }
```

6. Result/Output:



```
Main Menu
1. Add an Employee
2. Display All
3. Exit
Enter your choice: 1
Enter Employee ID: 10
Enter Employee Name: Kamal
Enter Employee Age: 30
Enter Employee Salary: 80000
Employee added successfully.
Main Menu
1. Add an Employee
2. Display All
3. Exit
Enter your choice: 2

        Report
10 Kamal 30 80000.0
----End of Report-----

Main Menu
1. Add an Employee
2. Display All
3. Exit
Enter your choice: 3
Exiting the System.
```

7. Learning Outcomes:

- Gain proficiency in reading from and writing to files in Java.
- Learnt how to design and implement a menu-driven user interface in Java..
- Learnt the importance of validating user input and handling potential errors gracefully.
- Understood to apply object-oriented principles such as encapsulation, inheritance, and polymorphism in designing the Employee Management System.
- Gained insight into basic CRUD (Create, Read, Update, Delete) operations in software development.