NAME	Ajay Singh Pargai
UID	23BCS11963
CLASS	622-A

Experiment – 2.2 (Part – c) :

Title

Menu-Based Employee Management System Using File Handling.

Objective

To develop a menu-driven Java application for adding and displaying employee records using file handling mechanisms.

Task Description

This task combines user interaction, file operations, and object or text data storage. The application should:

- Display a menu with three options:
    1. **Add an Employee**
        - Prompt user to enter employee details (name, ID, designation, salary).

        - Save the data to a file either in serialized form or as formatted text.
2. **Display All Employees**
    - Read data from the file.
    - Display employee records in a readable format.
3. **Exit the Application**
    - End the program gracefully.

This exercise demonstrates:

- Practical usage of **file I/O classes** like FileWriter, BufferedWriter, FileReader, BufferedReader, or object streams.
- Structuring applications with loops and Scanner for repeated menu interaction.
- Efficient data management through persistent storage and retrieval.

Code :

```
import java.io.*;

import java.util.Scanner;


class Employee {

   int id;

   String name;

   String designation;

   double salary;


   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 id + "," + name
+ "," + designation + "," +
salary;
    }

    static              Employee
fromString(String line) {
        String[]      parts      =
line.split(",");
        if  (parts.length  !=  4)
return null;
        int           id          =
Integer.parseInt(parts[0]);
        String       name        =
parts[1];
        String   designation   =
parts[2];
```

```
        double salary = Double.parseDouble(parts[3]);
        return new Employee(id, name, designation, salary);
    }

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

public class EmployeeManagement {
    private static final String FILE_NAME = "employees.txt";
```

```
  public      static      void
addEmployee(Scanner  sc)
{
     try  (FileWriter  fw  =
new
FileWriter(FILE_NAME,
true);
        BufferedWriter  bw
= new BufferedWriter(fw))
{


System.out.print("Enter
Employee ID: ");
        int id = sc.nextInt();
        sc.nextLine();

System.out.print("Enter
Employee Name: ");
        String     name     =
sc.nextLine();

System.out.print("Enter
Employee Designation: ");
```

```
          String designation =
sc.nextLine();

System.out.print("Enter
Employee Salary: ");
          double    salary    =
sc.nextDouble();
          sc.nextLine();

          Employee    emp    =
new  Employee(id,  name,
designation, salary);

bw.write(emp.toString());
          bw.newLine();

System.out.println("Emplo
yee added successfully.");
      } catch (IOException e)
{
          e.printStackTrace();
      }
   }
```

```
  public static void
displayAllEmployees() {
    try (FileReader fr =
new
FileReader(FILE_NAME);
        BufferedReader br
= new BufferedReader(fr))
{

      String line;
      boolean found =
false;
      while ((line =
br.readLine()) != null) {
        Employee emp =
Employee.fromString(line)
;
        if (emp != null) {

System.out.println(emp.pr
ettyPrint());
          found = true;
        }
      }
      if (!found) {
```

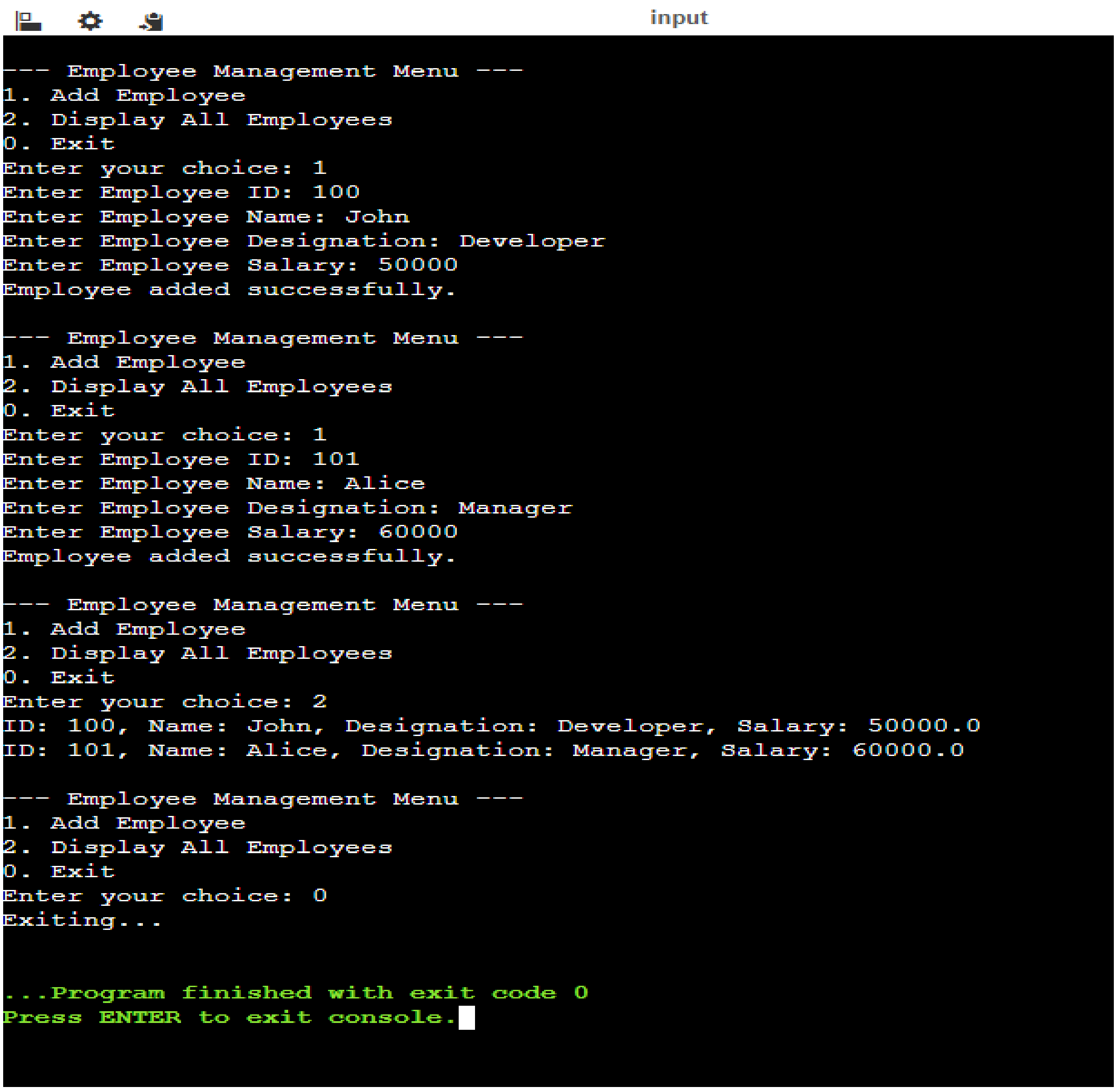
```
System.out.println("No
employees found.");
          }
      }                        catch
(FileNotFoundException e)
{

System.out.println("No
employees found. File not
created yet.");
      } catch (IOException e)
{
          e.printStackTrace();
      }
   }

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

      while (true) {
```

```
System.out.println("\n---
Employee      Management
Menu ---");

System.out.println("1. Add
Employee");

System.out.println("2.
Display All Employees");

System.out.println("0.
Exit");

System.out.print("Enter
your choice: ");

            if  (!sc.hasNextInt())
{
                sc.next();

System.out.println("Invali
d input. Try again.");
                continue;
            }
```

```
          choice                  =
sc.nextInt();
          sc.nextLine();

          switch (choice) {
             case 1:

addEmployee(sc);
                break;
             case 2:

displayAllEmployees();
                break;
             case 0:

System.out.println("Exitin
g...");
                sc.close();
                return;
             default:

System.out.println("Invali
d choice. Try again.");
          }
       }
```

```
    }
}
```

Output :



```
--- Employee Management Menu ---
1. Add Employee
2. Display All Employees
0. Exit
Enter your choice: 1
Enter Employee ID: 100
Enter Employee Name: John
Enter Employee Designation: Developer
Enter Employee Salary: 50000
Employee added successfully.

--- Employee Management Menu ---
1. Add Employee
2. Display All Employees
0. Exit
Enter your choice: 1
Enter Employee ID: 101
Enter Employee Name: Alice
Enter Employee Designation: Manager
Enter Employee Salary: 60000
Employee added successfully.

--- Employee Management Menu ---
1. Add Employee
2. Display All Employees
0. Exit
Enter your choice: 2
ID: 100, Name: John, Designation: Developer, Salary: 50000.0
ID: 101, Name: Alice, Designation: Manager, Salary: 60000.0

--- Employee Management Menu ---
1. Add Employee
2. Display All Employees
0. Exit
Enter your choice: 0
Exiting...


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