PCET's

Pimpri Chinchwad University

A

Project
Report On

"Employee Management System"

PRESENTED BY:

Name of Student : GAYATRI A. KADAM
Enrollment No : SOEL24201040033

GUIDED BY

Dr. Sagar Dhanraj Pande

SUBMITTED TO
PCET's
Pimpri Chinchwad University
OF

SECOND YEAR

DEPARTMENT OF COMPUTER ENGINEERING AND ENGG.

(Academic Year: 2024-25)

PCET's

Pimpri Chinchwad University

Certificate

This is to certify that

Name : GAYATRI A. KADAM
Enrollment No. SOEL24201040033

Have successfully completed the Java Mini project work titled "**Employee Management System**" in a satisfactory manner as part of the fulfillment of the requirements for the Second Year in Computer Science and Engineering for the academic year 2024-25.

Dr. Sagar Dhanraj Pande — Project Guide

Dr.V.N.Patil — Head Of Department

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this Database Management System project. First and foremost, I extend my heartfelt thanks to **PCET’s Pimpri Chinchwad University** for providing me the opportunity to undertake this mini project. I am deeply grateful to my guide **Dr. Sagar D Pande**, for his encouragement, continuous guidance, and valuable feedback throughout the development of the project. His expertise and suggestions were instrumental in shaping the success of this work. Lastly, I also thank my peers and all those who directly or indirectly supported and motivated me during this project.

[GAYATRI KADAM] [BTECH CSE]
Date :-

ABSTRACT

This Employee Management System Project application stores all the employee’s information in a database. It is an application developed in Java GUI technology and database used is SQLite. It contains employee information like employee id, first name, surname, and age. It is an easy to use application and has a user-friendly interface. It is totally built at the administrative end which means that only the admin has access rights to change or modify any records. So this makes it safe and reliable application to use. The main aim of developing this application was to reduce the errors that occur in the manual system. One can search the details easily by just entering employee id. All the details are stored,It is easy to update any employee details.

This will be a GUI-based program administrators will be able to perform the following :

- Login
- View all employees
- Create employee
- Edit employee
- Delete employee
- Save data

It contains employee information like employee id, first name, surname, and age. It is an easy to use application and has a user-friendly interface. It is totally built at the administrative end which means that only the admin has access rights to change or modify any records.

Structure Overview:

Employee class: Defines the properties and behavior of an employee (with getters, setters, and a toString method).

EmployeeService class: Contains business logic to manage the collection of employees.

Main class: Handles user interaction and menu navigation via console.

INDEX

Introduction :-

This Employee Management System Project application stores all the employee's information in a database.

- ▶ java.util.Scanner: This is crucial for getting user input from the console.
- ▶ java.util.HashSet: This is essential for storing employee objects without duplicates.

1. **Employee Class:**

- ➢ This class represents an employee with attributes like id, name, age, designation, department, and salary.
- ➢ private access modifiers ensure that these attributes can only be accessed or modified through the class's methods (encapsulation).
- ➢ Getter and Setter Methods: These methods (getId(), setId(), getName(), setName(), etc.) provide controlled access to the private attributes.
- ➢ toString() Method: Returns a string representation of the Employee object, making it easy to print employee details.
- ➢ Constructor: Initializes the Employee object with the provided values when a new employee is created.

2. **EmployeeService Class:**

- ➢ this class manages employee data and operations.
- ➢ HashSet<Employee> empset: A HashSet to store Employee objects. HashSet ensures that each employee is unique (no duplicate IDs).
- ➢ Employee emp1, emp2, emp3, emp4: Pre-populated employee objects for testing.
- ➢ Scanner sc: A Scanner object to read user input.
- ➢ found, id, name, age, department, designation, sal: Variables to store temporary data during operations.
- ➢ Constructor: Adds the pre-populated employee objects to the empset.
    - ➢ viewAllEmps():
    - ➢ viewEmp():
    - ➢ updateEmployee():
    - ➢ deleteEmp():
    - ➢ addEmp():

3. **Main Class:**

- ➢ This is the entry point of the program.
- ➢ EmployeeService service: Creates an instance of the EmployeeService class.
- ➢ static boolean ordering = true: controls the do while loop.
- ➢ menu(): Prints the menu options to the console.

- main():
- Creates a Scanner object to read user input.
- Creates an EmployeeService object.
- Uses a do-while loop to display the menu and handle user choices.
- A switch statement calls the appropriate method from the EmployeeService based on the user's choice.
- Exits the program when the user chooses option 6.
- Handles invalid user input.

Objectives :-

The objective of the provided Java code is to implement a console-based Employee Management System that allows users to perform basic CRUD (Create, Read, Update, Delete) operations on employee records using a HashSet. The system uses object-oriented principles and is operated through a simple menu-driven interface.

System Architecture :-

- Frontend: Java

Source Code :-

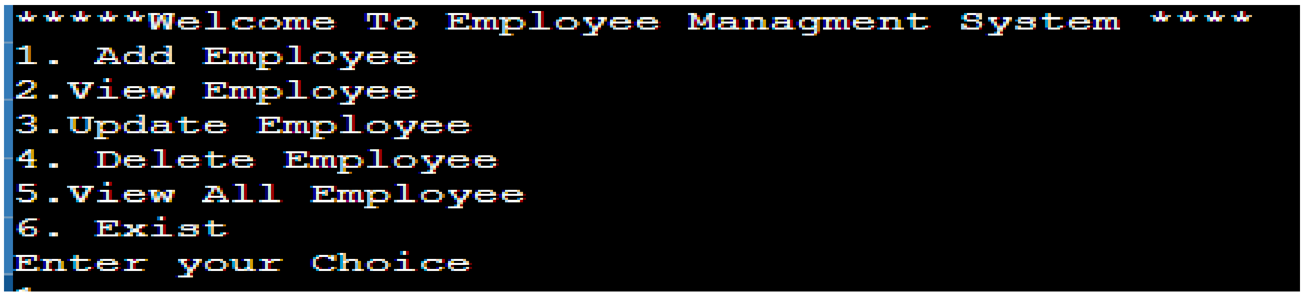
```
import java.util.Scanner; //user input from console
import java.util.HashSet; //for storing employee objects


class Employee {
    private int id;
    private String name;
    private int age;
    private String designation;
    private String department;
    private double salary;

    public int getId() {
        return id;
    }
```

```
public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public String getDesignation() {
    return designation;
}

public void setDesiganation(String designation) {
    this.designation = designation;
}

public String getDepartment() {
    return department;
}

public void setDepartment(String department) {
    this.department = department;
}

public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public String toString() {
        return "Employee [id=" + id + "\n name=" + name + "\n age=" + age + "\n
designation="
                + designation + "\n department=" + department + "\n salary=" + salary
+ "]";
    }

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

class EmployeeService {
    HashSet<Employee> empset = new HashSet<Employee>();
    Employee emp1 = new Employee(101, "ABC", 24, "Developer", "IT",
60000);
    Employee emp2 = new Employee(102, "EFG", 26, "Tester", "CO", 62000);
    Employee emp3 = new Employee(103, "NEHA", 20, "Manager", "IT",
50000);
    Employee emp4 = new Employee(104, "XYZ", 27, "System Eng", "CO",
50000);
    Scanner sc = new Scanner(System.in);
    boolean found = false;
    int id;
```

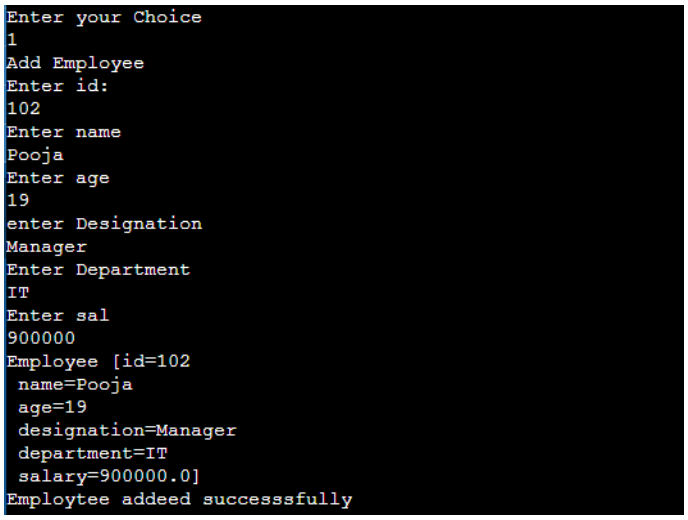
```
String name;
int age;
String department;
String designation;
double sal;

public EmployeeService() {
    empset.add(emp1);
    empset.add(emp2);
    empset.add(emp3);
    empset.add(emp4);
}

// view all employees
public void viewAllEmps() {
    for (Employee emp : empset) {
        System.out.println(emp);
    }
}

public void viewEmp() {
    System.out.println("Enter id: ");
    id = sc.nextInt();
    for (Employee emp : empset) {
        if (emp.getId() == id) {
            System.out.println(emp);
            found = true;
        }
    }
    if (!found)
        System.out.println("Employee with this id is not present");
}

public void updateEmployee() {
    System.out.println("Enter id: ");
    id = sc.nextInt();
    boolean found = false;
    for (Employee emp : empset) {
        if (emp.getId() == id) {
            System.out.println("Enter name: ");
            name = sc.next();
            System.out.println("Enter new Salary");
            sal = sc.nextDouble();
            emp.setName(name);
            emp.setSalary(sal);
            System.out.println("Updated Details of employee are: ");
            System.out.println(emp);
            found = true;
        }
    }
    if (!found) {
        System.out.println("Employee is not present");
    } else {
        System.out.println("Employee details updated successfully !!");
    }
}

// delete emp
public void deleteEmp() {
    System.out.println("Enter id");
    id = sc.nextInt();
    boolean found = false;
    Employee empdelete = null;
    for (Employee emp : empset) {
        if (emp.getId() == id) {
            empdelete = emp;
            found = true;
        }
    }
    if (!found) {
        System.out.println("Employee is not present");
    } else {
        empset.remove(empdelete);
        System.out.println("Employee deleted successfully!!");
    }
}
```

```
    public void addEmp() {
        System.out.println("Enter id:");
        id = sc.nextInt();
        System.out.println("Enter name");
        name = sc.next();
        System.out.println("Enter age");
        age = sc.nextInt();
        System.out.println("enter Designation");
        designation = sc.next();
        System.out.println("Enter Department");
        department = sc.next();
        System.out.println("Enter sal");
        sal = sc.nextDouble();
        Employee emp = new Employee(id, name, age, designation, department,
sal);
        empset.add(emp);
        System.out.println(emp);
        System.out.println("Employee addeed successsfully");
    }
}

public class Main {
    EmployeeService service = new EmployeeService();
    static boolean ordering = true;

    public static void menu() {
        System.out.println("*****Welcome To Employee Managment System
**** "
                + "\n1. Add Employee "
                + "\n2.View Employee"
                + "\n3.Update Employee"
                + "\n4. Delete Employee"
                + "\n5.View All Employee"
                + "\n6. Exist ");
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        EmployeeService service = new EmployeeService();
        do {
            menu();
            System.out.println("Enter your Choice");
            int choice = sc.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Add Employee");
                    service.addEmp();
                    break;
                case 2:
                    System.out.println("View Employee");
                    service.viewEmp();
                    break;
                case 3:
                    System.out.println("Update Employee");
                    service.updateEmployee();
                    break;
                case 4:
                    System.out.println("Delete Employee");
                    service.deleteEmp();
                    break;
                case 5:
                    System.out.println("view All Employee");
                    service.viewAllEmps();
                    break;
                case 6:
                    System.out.println("Thank you for using application!!");
                    System.exit(0);
                default:
                    System.out.println("Please enter valid choice");
                    break;
            }

        } while (ordering);
    }
}
```

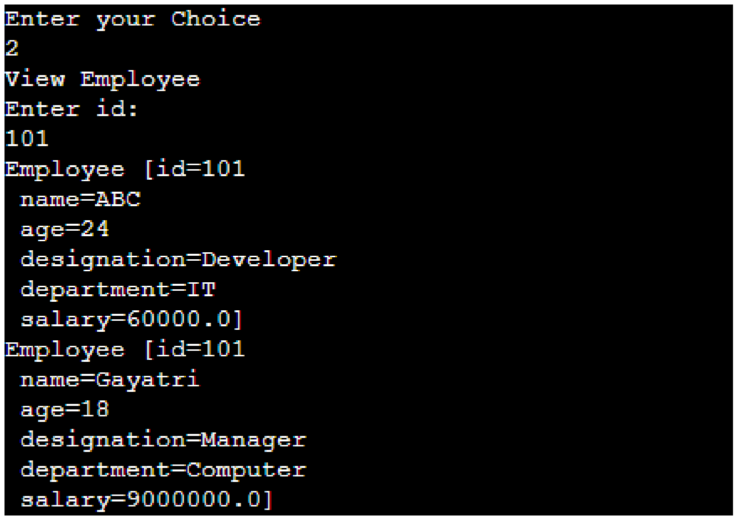
Output :-



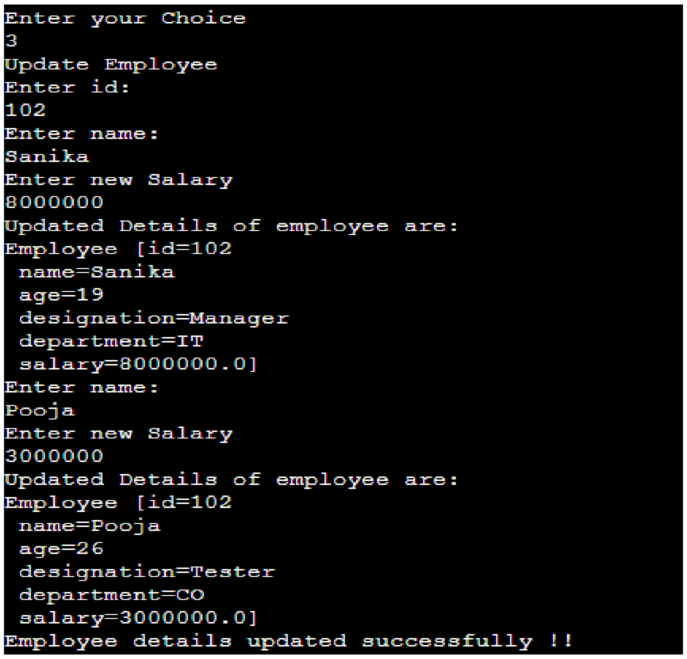
```
*****Welcome To Employee Managment System ****
1. Add Employee
2.View Employee
3.Update Employee
4. Delete Employee
5.View All Employee
6. Exist
Enter your Choice
```



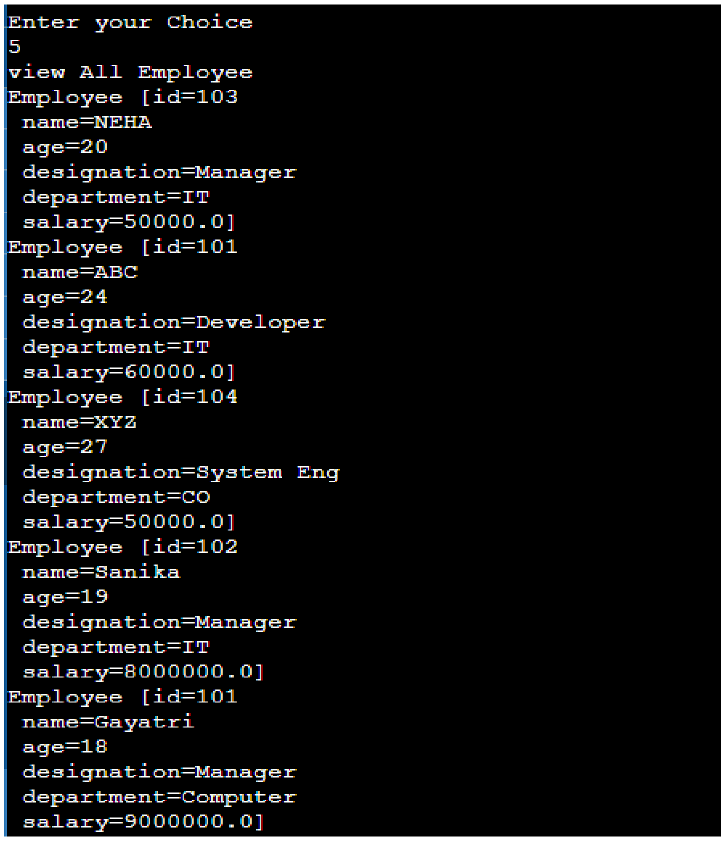
```
Enter your Choice
1
Add Employee
Enter id:
102
Enter name
Pooja
Enter age
19
enter Designation
Manager
Enter Department
IT
Enter sal
900000
Employee [id=102
 name=Pooja
 age=19
 designation=Manager
 department=IT
 salary=900000.0]
Employtee addeed successsfully
```



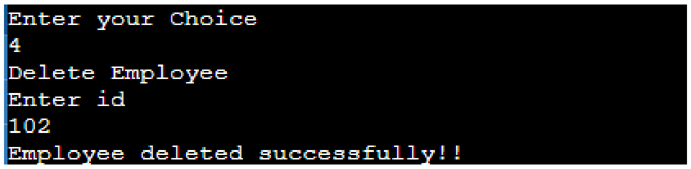
```
Enter your Choice
2
View Employee
Enter id:
101
Employee [id=101
 name=ABC
 age=24
 designation=Developer
 department=IT
 salary=60000.0]
Employee [id=101
 name=Gayatri
 age=18
 designation=Manager
 department=Computer
 salary=9000000.0]
```



```
Enter your Choice
3
Update Employee
Enter id:
102
Enter name:
Sanika
Enter new Salary
8000000
Updated Details of employee are:
Employee [id=102
 name=Sanika
 age=19
 designation=Manager
 department=IT
 salary=8000000.0]
Enter name:
Pooja
Enter new Salary
3000000
Updated Details of employee are:
Employee [id=102
 name=Pooja
 age=26
 designation=Tester
 department=CO
 salary=3000000.0]
Employee details updated successfully !!
```



```
Enter your Choice
5
view All Employee
Employee [id=103
 name=NEHA
 age=20
 designation=Manager
 department=IT
 salary=50000.0]
Employee [id=101
 name=ABC
 age=24
 designation=Developer
 department=IT
 salary=60000.0]
Employee [id=104
 name=XYZ
 age=27
 designation=System Eng
 department=CO
 salary=50000.0]
Employee [id=102
 name=Sanika
 age=19
 designation=Manager
 department=IT
 salary=8000000.0]
Employee [id=101
 name=Gayatri
 age=18
 designation=Manager
 department=Computer
 salary=9000000.0]
```



```
Enter your Choice
4
Delete Employee
Enter id
102
Employee deleted successfully!!
```

Advantages :-

1. Object-Oriented Design: Uses classes and objects to model real-world employee entities, promoting reusability and modularity.
2. Basic CRUD Operations:Provides essential operations like Add, View, Update, and Delete for managing employee records.
3. Data Handling Using Collections:Uses HashSet for storing employee objects, which avoids duplicate entries and provides efficient operations.
4. Interactive Menu Interface:Menu-driven interface makes it easy for users to interact with the system.
5. Extensible Structure:Code can be easily extended to add features like file handling, sorting, or searching by different attributes.

Limitations:-

1. Incorrect Method Placement & Scope Issues
2. Incorrect Setter Method Name
3. Missing ordering = false Condition
4. Improper Salary Input in addEmp()
6. No Persistence

Future Scope :-

1. File Handling / Database Integration
2. Graphical User Interface (GUI)
3. Mobile Application Support

Conclusion :-

The code provides a foundational implementation of an Employee Management System using core Java features like classes, collections, and console I/O. It demonstrates key programming concepts but contains a few syntax, logic, and structural issues that must be resolved for the application to compile and function correctly.

With proper fixes and enhancements—such as file/database integration, error handling, and UI improvements—this program could serve as a base for more complex management systems.