Employee Management System

```
package EMS;

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

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

    //Getters and Setters

    public int getId() {
        return id;
    }

    public String getName(){
        return name;
    }

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

    public String getDesignation() {
        return designation;
    }

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

    public double getSalary() {
        return salary;
    }

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

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

```java
package EMS;

public interface EmployeeManagement {
    void addEmployee(Employee employee);
    void viewAllEmployees();
    void updateEmployee(int empId);
    void deleteEmployee(int empId);
    Employee findEmployeeById(int empId);
}
```

```java
package EMS;
import java.util.List;
import java.util.ArrayList;
import java.util.Scanner;

public class EmployeeManagementImpl implements
EmployeeManagement {

    private List<Employee> employeeList = new ArrayList<>();

    //Add employee to the list
    @Override
    public void addEmployee(Employee employee){
        employeeList.add(employee);
        System.out.println("Employee Added Successfully");
    }

    //View all employees
    @Override
    public void viewAllEmployees(){
        if (employeeList.isEmpty()){
            System.out.println("No employees Found");
        } else {
            System.out.println("List of Employees :");
            for(Employee employee : employeeList){
                System.out.println(employee);
            }
        }
    }

    // Update employee details by ID
    @Override
    public void updateEmployee(int empId){
      Employee employee =findEmployeeById(empId);
      if(employee != null){
          Scanner scanner = new Scanner(System.in);
          System.out.println("Enter New Name : ");
          employee.setName(scanner.nextLine());
          System.out.println("Enter new designation");
```

```
            employee.setDesignation(scanner.nextLine());
            System.out.println("Enter new Salary : ");
            employee.setSalary(scanner.nextDouble());
            System.out.println("Employee updated");
        } else {
            System.out.println("Employee not found");
        }
    }

    //Delete employee by ID
    public void deleteEmployee(int empId){
        Employee employee  = findEmployeeById(empId);
        if(employee != null){
            employeeList.remove(employee);
            System.out.println("Employee Deleted");
        }else {
            System.out.println("Employee not found");
        }
    }

    //Find employee by ID
    @Override
    public Employee findEmployeeById(int empId){
        for(Employee employee : employeeList ){
            if(employee.getId() == empId){
                return employee;
            }
        }
        return null;
    }
}
```

```
package EMS;

import java.util.Scanner;

public class EmployeeManagementSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        EmployeeManagementImpl management = new
EmployeeManagementImpl();
        boolean exit = false;

        System.out.println("Welcome to Employee Management
System");
        while(!exit){
            System.out.println("\n Menu :");
            System.out.println("1. Add Employee");
            System.out.println("2. View All Employee");
            System.out.println("3. Update Employee");
            System.out.println("4. Delete Employee");
            System.out.println("5. Exit");
            System.out.println("Enter your choice : ");
            int choice = scanner.nextInt();

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

                    Employee employee = new Employee(id, name,
designation, salary);
                    management.addEmployee(employee);
                    break;

                case 2:
                    management.viewAllEmployees();
                    break;

                case 3:
                    System.out.println("Enter Employee ID to
update :");
                    int updateID = scanner.nextInt();
```

```
                    management.updateEmployee(updateID);
                    break;

                case 4:
                    System.out.println("Enter Employee ID to
delete :");
                    int deleteID = scanner.nextInt();
                    management.deleteEmployee(deleteID);
                    break;

                case 5:
                    exit = true;
                    System.out.println("GoodBye!");
                    break;

                default:
                    System.out.println("Invalid choice!");

            }
        }
        scanner.close();
    }
}
```

SortedSet Interface:

It is a sub-interface of set and it ensures elements are stored in a sorted order.

Key Features:

Maintains elements in ascending order by default.

Does not allow duplicate elements

TreeSet Class:

Thread-unsafe (use Collections.synchronizedSet for Synchronization)

```
package SortedSet;

import java.util.TreeSet;

public class TreeSetExample {
    public static void main(String[] args) {
        TreeSet<Integer> treeSet = new TreeSet<>();

        //Adding elements
        treeSet.add(50);
        treeSet.add(30);
        treeSet.add(40);
        treeSet.add(10);
        treeSet.add(20);

        System.out.println("TreeSet : " + treeSet);

        //Different methods
        System.out.println("First Element : "+
treeSet.first());
        System.out.println("Last Element : "+ treeSet.last());
        System.out.println("HeadSet(less than 30) :" +
treeSet.headSet(30));
        System.out.println("TailSet(greater than or equal to
30) : "+ treeSet.tailSet(30));
        System.out.println("Subset (20-40) : " +
treeSet.subSet(20,40));
    }
}
```

TreeSet (Java SE 21 & JDK 21)

Queue

It Follows FIFO(First In First Out) principle.

PriorityQueue:

A concrete implementation of queue.

```
package Queue;

import java.util.PriorityQueue;

public class PriorityQueueExample {
    public static void main(String[] args) {
        PriorityQueue<Integer> pq = new PriorityQueue<>();

        //Adding Elements
        pq.add(40);
        pq.add(30);
        pq.add(10);
        pq.add(20);

        System.out.println("Priority Queue : "+ pq);

        //Accessing an element
        System.out.println("Peek (Highest Priority) : " +
pq.peek());

        //Removing an element
        System.out.println("Poll : "+ pq.poll());

        System.out.println("Priority Queue : "+ pq);

    }
}
```