**QUESTION 40:**

A white text with black text

AI-generated content may be incorrect.

**CODE:**

**Employee.java (Base Class)**

public class Employee {

protected String empId;

protected String name;

protected String role;

protected String dept;

protected String reviewer;

public Employee(String empId, String name, String role, String dept, String reviewer) {

this.empId = empId;

this.name = name;

this.role = role;

this.dept = dept;

this.reviewer = reviewer;

}

public String getEmpId() { return empId; }

public String getDept() { return dept; }

public double finalRating() {

return 0.0; // Base employee has no specific rule

}

public String display() {

return empId + ": " + name + " | " + role + " | Dept: " + dept;

}

}

**Manager.java (Extends Employee)**

public class Manager extends Employee {

public Manager(String empId, String name, String dept) {

super(empId, name, "Manager", dept, null);

}

@Override

public double finalRating() {

return 5.0; // Managers have default high rating

}

}

**Objective.java**

import java.util.ArrayList;

import java.util.List;

public class Objective {

String objId;

String title;

String description;

double weightage;

String status;

List<KeyResult> keyResults;

public Objective(String objId, String title, String description, double weightage) {

this.objId = objId;

this.title = title;

this.description = description;

this.weightage = weightage;

this.status = "Open";

this.keyResults = new ArrayList<>();

}

public void addKeyResult(KeyResult kr) {

keyResults.add(kr);

}

public double calculateProgress() {

return keyResults.stream().mapToDouble(kr -> kr.progress).average().orElse(0.0);

}

public void finalizeStatus() {

status = calculateProgress() >= 80 ? "Achieved" : "Pending";

}

}

**KeyResult.java**

public class KeyResult {

String krId;

String objectiveId;

String metric;

double target;

double progress;

public KeyResult(String krId, String objectiveId, String metric, double target) {

this.krId = krId;

this.objectiveId = objectiveId;

this.metric = metric;

this.target = target;

this.progress = 0.0;

}

public void updateProgress(double value) {

progress = value;

}

public void updateProgress(int percent) {

progress = target \* percent / 100.0;

}

}

**ReviewService.java**

import java.util.ArrayList;

import java.util.List;

public class ReviewService {

List<Employee> employees = new ArrayList<>();

List<Objective> objectives = new ArrayList<>();

public void addEmployee(Employee emp) {

employees.add(emp);

}

public void addObjective(Objective obj) {

objectives.add(obj);

}

public void submitReview(Objective obj) {

obj.finalizeStatus();

}

public void printScorecards() {

System.out.println("\n--- Employee Scorecards ---");

for (Employee e : employees) {

System.out.printf("%s | Final Rating: %.1f\n", e.display(), e.finalRating());

}

}

public void deptWiseRatingSummary(String dept) {

long count = employees.stream().filter(e -> e.dept.equals(dept)).count();

System.out.printf("\nDept %s has %d employees\n", dept, count);

}

}

**OKRAppMain.java**

public class OKRAppMain {

public static void main(String[] args) {

ReviewService service = new ReviewService();

Employee emp1 = new Employee("E001", "Alice", "Developer", "IT", "M001");

Manager mgr1 = new Manager("M001", "Bob", "IT");

service.addEmployee(emp1);

service.addEmployee(mgr1);

Objective obj1 = new Objective("O001", "Improve App Performance", "Speed up app by 30%", 50);

KeyResult kr1 = new KeyResult("KR001", "O001", "Response Time (ms)", 100);

kr1.updateProgress(85); // Set progress by value

obj1.addKeyResult(kr1);

service.addObjective(obj1);

service.submitReview(obj1);

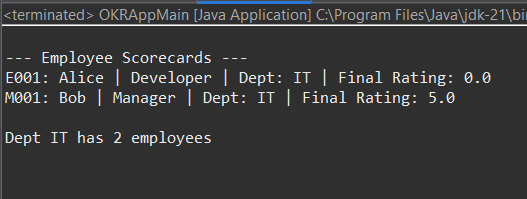
service.printScorecards();

service.deptWiseRatingSummary("IT");

}

}

**OUTPUT:**

****