**Case Study: Employee Management System**

**Objective**

To understand how **classes, constructors, and inheritance** can be used to design a simple **real-world application** — an Employee Management System.

**1. Problem Statement**

A company wants to maintain employee records in a digital system.

They have **different types of employees**:

* Regular full-time employees
* Contract-based employees
* Managers who handle a team

Each employee should have:

* A unique ID
* Name
* Basic salary

Managers and Contract employees have **additional attributes**.

You are asked to:

* Create a class hierarchy using **inheritance**
* Use **constructors** to initialize data
* Use **method overriding** where needed
* Print details of each type of employee

**2. Class Design (UML Overview)**

┌──────────────────┐

│ Employee │

│------------------│

│ - id: int │

│ - name: String │

│ - salary: double │

│------------------│

│ + showDetails() │

└────────▲─────────┘

│

┌──────────────────┼───────────────────┐

│ │

│ │

▼ ▼

┌────────────────────┐ ┌─────────────────────┐

│ RegularEmployee │ │ ContractEmployee │

│--------------------│ │---------------------│

│ + hra: double │ │ + contractPeriod: int │

+ pf :double

│--------------------│ │---------------------│

│ + showDetails() │ │ + showDetails() │

└────────────────────┘ └─────────────────────┘

▼

┌────────────────────┐

│ Manager │

│--------------------│

│ + teamSize: int │

│--------------------│

│ + showDetails() │

└────────────────────┘

public class EmployeeManagement {

public static void main(String[] args) {

RegularEmployee e1 = new RegularEmployee(101, "Ravi", 45000, 5000);

ContractEmployee e2 = new ContractEmployee(102, "Meena", 30000, 12);

Manager m1 = new Manager(103, "Anil", 60000, 7000, 5);

System.out.println("=== Employee Details ===");

e1.showDetails();

e2.showDetails();

m1.showDetails();

}

}