# Java OOPs Practice Problem: Employee Management

## Problem Statement:

Write a Java program to model a simple Employee management system using OOP concepts.

Create two classes:

1. Employee – represents individual employee details.
2. EmployeeTester – contains the main() method to test your class.

## Requirements:

### Class: Employee

##### 🏢 Static Members (shared across all employees):

* companyName – Name of the company (String)
* empCounter – Used to auto-generate employee IDs (int)
* A static block to initialize these values.

##### 👤 Instance Members (per employee):

* empId – Auto-generated using empCounter
* name – Employee name
* email – Email address
* phone – Phone number
* basicSalary – Base monthly salary
* skills[] – A list of skills the employee has

##### 🛠 Constructors:

1. Default constructor (prints welcome message)
2. Parameterized constructor: Initializes name, email, phone, salary, and skills

##### 🧮 Methods:

* calculateSalary() – Calculate and print gross salary by adding basic salary + 20% bonus
* displayDetails() – Display all employee details neatly using this
* Override toString() method to print a summary

## Sample Usage from EmployeeTester:

System.out.println(Employee.companyName);

Employee e1 = new Employee("Ram", "ram@gmail.com", 9876543210L, 30000.0f, new String[]{"Java", "Spring"});

System.out.println("Employee Info: " + e1);

Employee e2 = new Employee("Sita", "sita@gmail.com", 9876500000L, 35000.0f, new String[]{"UI/UX", "Figma", "HTML"});

e1.displayDetails();

e2.displayDetails();

## Output Sample (You can decide layout):

TCS

Welcome to TCS

Employee Info: Employee [company=TCS, empId=1001, name=Ram, email=ram@gmail.com]

Welcome to TCS

Employee Info: Employee [company=TCS, empId=1002, name=Sita, email=sita@gmail.com]

--- Employee Details ---

ID: 1001

Name: Ram

Email: ram@gmail.com

Phone: 9876543210

Skills: Java, Spring

Basic Salary: ₹30000.0

Gross Salary: ₹36000.0

## Concepts Students Will Practice:

* static keyword (variables + block)
* Constructor chaining using this()
* toString() override
* Working with arrays
* Using this for referencing fields
* System.out.println() vs toString()
* Optional: You can extend this with ArrayList for skills