**Background**: A payroll system calculates salaries for different employee types. A general Employee class calculates salary based on hours worked, while ContractEmployee (a derived class) might have a different payment calculation, like a fixed contract rate. The payroll system processes all employee types and expects consistent results. In the future we might need more salary calculation logic.

**Question**: How would you design the Employee and ContractEmployee classes to ensure the payroll system handles both without needing special conditions?

**enum SALARY\_TYPE {**

**HOURLY = "Hourly",**

**CONTRACT = "contract",**

**}**

**abstract class Employee {**

**protected name: string;**

**protected type: SALARY\_TYPE;**

**protected joiningDate: string;**

**constructor(name: string, type: SALARY\_TYPE, joiningDate: string) {**

**this.name = name;**

**this.type = type;**

**this.joiningDate = joiningDate;**

**}**

**abstract calculateSalary(): void;**

**}**

**class HourlyEmployee extends Employee {**

**private hourlyRate: number;**

**private hours: number;**

**constructor(**

**hourlyRate: number,**

**hours: number,**

**name: string,**

**type: SALARY\_TYPE,**

**joinDate: string**

**) {**

**super(name, type, joinDate);**

**this.hourlyRate = hourlyRate;**

**this.hours = hours;**

**}**

**calculateSalary(): void {**

**console.log(`${this.name} Salary: $${this.hourlyRate \* this.hours}`);**

**}**

**}**

**class FixedEmployee extends Employee {**

**private fixedSalary: number;**

**constructor(**

**fixedSalary: number,**

**name: string,**

**type: SALARY\_TYPE,**

**joinDate: string**

**) {**

**super(name, type, joinDate);**

**this.fixedSalary = fixedSalary;**

**}**

**calculateSalary(): void {**

**console.log(`${this.name} Salary: $${this.fixedSalary}`);**

**}**

**}**

**function main() {**

**const employees: Employee[] = [**

**new HourlyEmployee(60, 160, "Munira", SALARY\_TYPE.HOURLY, "12-12-2020"),**

**new FixedEmployee(50000, "Hasan", SALARY\_TYPE.CONTRACT, "12-12-2020"),**

**];**

**employees.forEach((employee) => employee.calculateSalary());**

**}**