HR

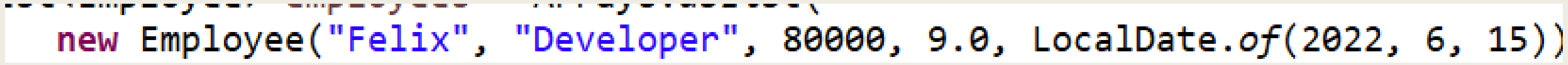
Ever Hinojosa Aguirre

HR

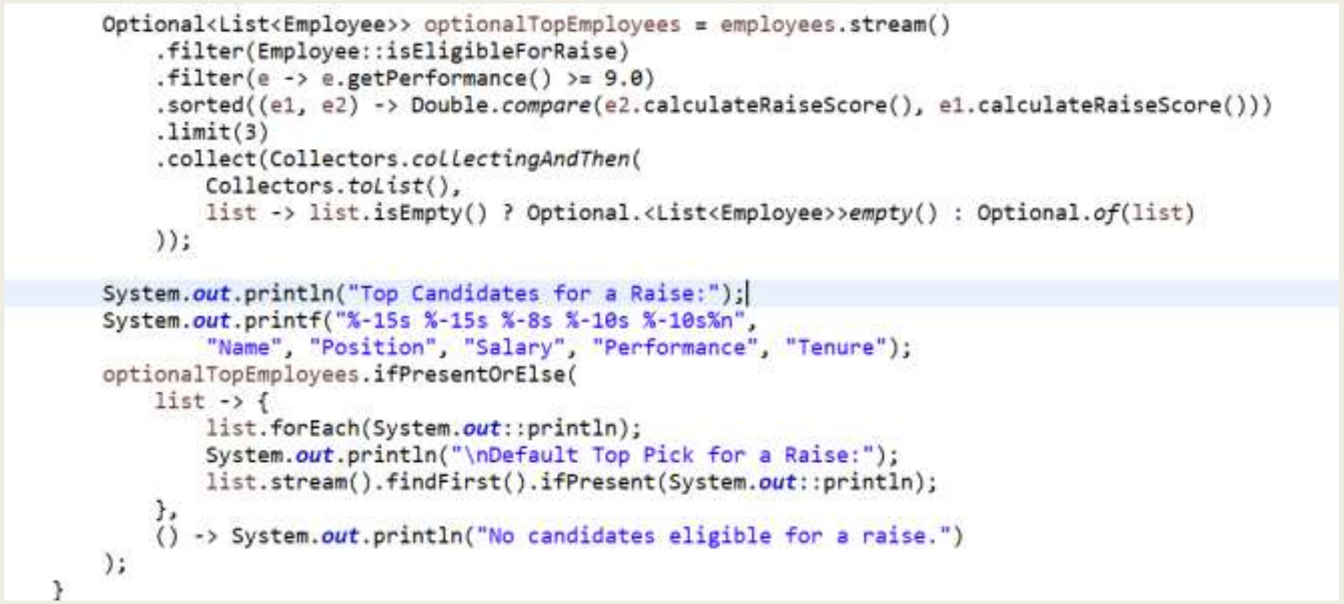
- i wanted to get a try at a HR code that by filtering out a list tries to find the most suitable employee to get a raise. The metrics chosen by me include the performance of the employee and the tenure. Even if the code gives out a default “Top pick” based on these metrics i know alot of more factors are envolved but for the proyect i took those in account.

HR.java

- The main again, here i define each of the employees givien each a name, position, salary, a “performance” and a Hire date to calculate the tenure.



```java
  new Employee("Felix", "Developer", 80000, 9.0, LocalDate.of(2022, 6, 15))
```



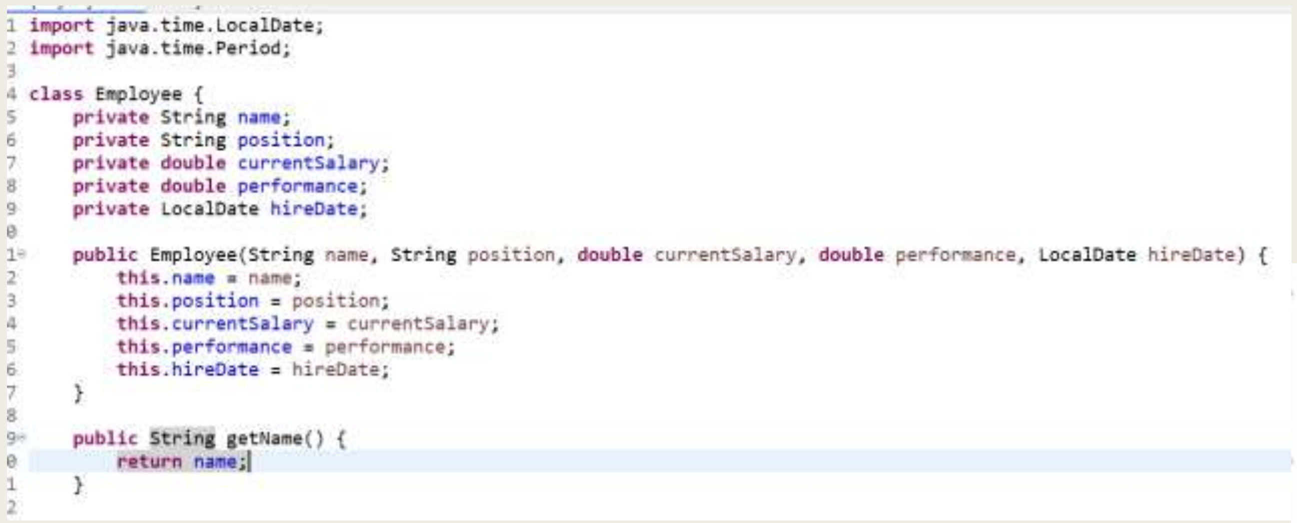
```java
        Optional<List<Employee>> optionalTopEmployees = employees.stream()
            .filter(Employee::isEligibleForRaise)
            .filter(e -> e.getPerformance() >= 9.0)
            .sorted((e1, e2) -> Double.compare(e2.calculateRaiseScore(), e1.calculateRaiseScore()))
            .limit(3)
            .collect(Collectors.collectingAndThen(
                Collectors.toList(),
                list -> list.isEmpty() ? Optional.<List<Employee>>empty() : Optional.of(list)
            ));

        System.out.println("Top Candidates for a Raise:");
        System.out.printf("%-15s %-15s %-8s %-10s %-10s%n",
                "Name", "Position", "Salary", "Performance", "Tenure");
        optionalTopEmployees.ifPresentOrElse(
            list -> {
                list.forEach(System.out::println);
                System.out.println("\nDefault Top Pick for a Raise:");
                list.stream().findFirst().ifPresent(System.out::println);
            },
            () -> System.out.println("No candidates eligible for a raise.")
        );
    }
```

- First for the stream i check if the employee has more tan a year in the Company to see if they are eligible for a raise
- Then i filter out any employee that has a lower performance tan 9.0
- Next i sort them out by their Raise score
- Now i limit the list for the top 3 and collect it to check if its empty .

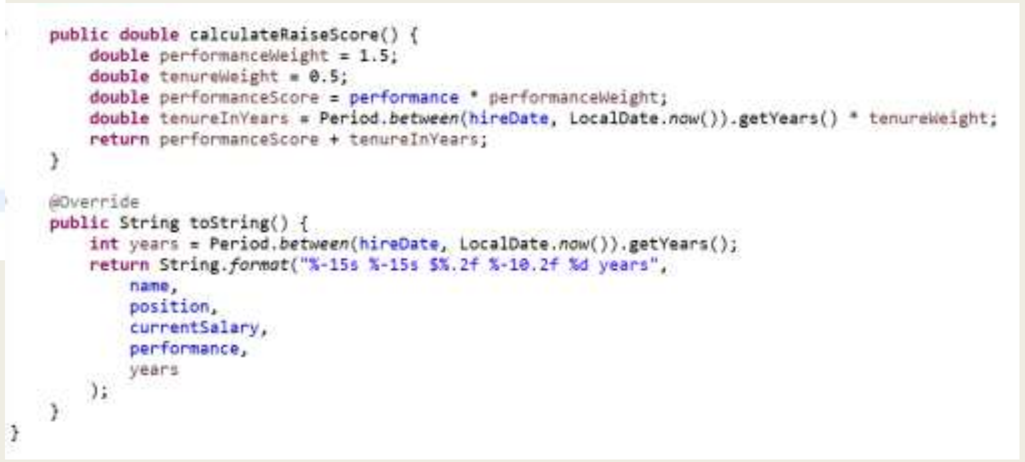
- Finally i check with an optional if any employees are present to print those 3 and a default top pick.

Employee.java

- The Employee class has its attributes: name, position, currentSalary, performance and hireDate. To get the Raise score I multiply the performance by 1.5 and the tenure by 0.5 and then add them up to make a score which takes both in account but gives more weight to the performance.



```java
import java.time.LocalDate;
import java.time.Period;

class Employee {
    private String name;
    private String position;
    private double currentSalary;
    private double performance;
    private LocalDate hireDate;

    public Employee(String name, String position, double currentSalary, double performance, LocalDate hireDate) {
        this.name = name;
        this.position = position;
        this.currentSalary = currentSalary;
        this.performance = performance;
        this.hireDate = hireDate;
    }

    public String getName() {
        return name;
    }

```



```java
    public double calculateRaiseScore() {
        double performanceWeight = 1.5;
        double tenureWeight = 0.5;
        double performanceScore = performance * performanceWeight;
        double tenureInYears = Period.between(hireDate, LocalDate.now()).getYears() * tenureWeight;
        return performanceScore + tenureInYears;
    }

    @Override
    public String toString() {
        int years = Period.between(hireDate, LocalDate.now()).getYears();
        return String.format("%-15s %-15s $%.2f %-10.2f %d years",
            name,
            position,
            currentSalary,
            performance,
            years
        );
    }
}
```