# Радио оператор

Напишете програма, която да моделира радио станция. Всяка радио станция има оператор и слушатели. Когато излезе прогнозата за времето, операторът съобщава това и всички слушатели биват оповестени.

Напишете:

* клас **RadioEvent**, който представя събитието „Операторът съобщава по радиото“.

Той трябва да има поле **message**, конструктор за общо ползване, getter и setter.

* Интерфейс **EventListener** – имплементацията на този интерфейс моделира човек, който слуша радиото. Напишете в него метод  
  void react(RadioEvent event);
* клас **RadioOperator** – операторът, който съобщава прогнозата. Той трябва да има абонирани слушатели, представете ги чрез масив от **EventListener.**

Напишете:

* + Клас данни
    - EventListener[] listeners
    - int currentIdx
  + Конструктор за общо ползване

public **RadioOperator**(int maximumListeners)

който инициализира масивът със слушатели с дължина **maximumListeners**. Стойността на currentIdx да е 0.

* + Метод

public void addListener(**EventListener** listener)

Който да добавя към масива от слушатели – listeners, слушателят listener

* + Метод

public void fireBroadcastEvent(**RadioEvent** event)

който оповестява всичките си слушатели за настъпилото събитие **event**

* Напишете клас **RadioTest**, в който да тествате програмата.
  + Създайте радио оператор, който може да има до 100 слушателя.
  + Използвайки анонимни класове, добавете 3 слушателя към него, така че:
    - Когато бъде оповестен за събитие, слушател 1 реагира с  
      „Listener 1 is happy of event: <event message>“
    - Когато бъде оповестен за събитие, слушател 2 реагира с  
      „Listener 2 is not happy with event: <event message>“
    - Когато бъде оповестен за събитие, слушател 1 реагира с  
      „Listener 1 is doesn’t care about event: <event message>“
  + Операторът оповестява всичките си слушатели, че ще вали.

Извикайте метод **fireBroadcastEvent**, с **RadioEvent** обект, предаващ съобщението "It's going to rain"

# Софтуерна Компания

Напишете програма, която да моделира софтуерна компания. Една компания се състои от различни служители – програмисти (developer), тестъри (QA) и HR-и. Трабва да предоставите начин на потребителите да търсят служители по въведен през конзолата критерий. Например, ако потребителят иска да намери всички програмисти, които са на ниво junior и работят във фирмата от поне 3 години, ще напише следната заявка:

position = DEVELOPER skillLevel = JUNIOR yearsOfService > 2

Това може да е заявка за повишяване на тези потребители.

* Напишете класове тип enum Sex, Position и SkillLevel. Техните стойности да са:
  + Sex: FEMALE, MALE
  + Position: DEVELOPER, QA, HR
  + SkillLevel: JUNIOR, MID, SENIOR
* Напишете клас **Employee**, който да има следните полета:
  + String name;
  + Sex sex;
  + Position position;
  + SkillLevel skillLevel;
  + int yearsOfService;
  + int salary;

Да се генерират конструкор, getter, setter и toString метод

* Напишете интерфейс **SearchCriteria** – имплементирайки този интерфейс, може да бъде зададено условие, което проверява всеки служител дали отговаря на него.
  + Да се напише метод

boolean matches(Employee employee);

* Напишете клас **Query –** този клас ще предоставя възможност за изпълняване на заявка върху всичките служители.
  + Напишете в него метод

public static Employee[] and(**Employee**[] employees, **SearchCriteria**[] criteria)

който да прилага **matches** метода на всяка една критерия върху всеки един employee обект. Критерията трябва да се комбинира с **&&**, т.е. един employee обект отговаря на критерията, ако  
 criteria[0].matches(employee) && criteria[1].matches(employee) && …Измислите как да напишете този алгоритъм.

* Напишете клас **CompanyUtils**
  + Напишете метод

public static Employee[] generateRandomEmployees(final int numberOfEmployees)

който да генерира numberOfEmployees наброй случайни служители

* Клас **CompanyTest**
  + Използвайки CompanyUtils, генерирайте 100 случайни служителя
  + Създайте два критерия за търсене, **isJunior**, проверяващ дали един служител е junior и **hasAtleastOneYear**, проверяващ дали служител има поне година стаж. Използвайте анонимни класове.
  + Използвайки **Scanner** обект, прочетете от конзолата заявката за търсене в String обект **searchQuery**
  + Заявката е във вид

<keyword> <operator> <value>

Използвайте **split** метода, за да разделите searchQuery на масив от думи, като за разделител използвайте празен интервал. Така разделени, могат да бъдат обхождани на групи по 3, като за всяка група, всеки първи елемент е keyword, всеки втори operator и всеки трети value.

* + Използвайки масива от горната подточка, създайте нов масив, от **SearchCriteria**. За всяка тройка <keyword> <operator> <value>, добавете в този масив подходяща имплементация на SearchCriteria, така че да се проверява дали клас данната <keyword>, принадлежаща на клас Employee, отговаря на стойността <value> според оператора <operator>. За **enum** клас данните е ясно, че операторът може да е само „=“. За salary и yearsOfService се поддържат оператори =, > и <. Ако се срещне невалидна група (например sex > MALE, name > “Ivan” и т.н.), да се изведе грешка. Използвайте статичният метод ***valueOf***на enum типовете, например Position.valueOf(“JUNIOR”)
  + Да се изведе на стандартния изход списъка от служители, отговарящи на критерия
* По желание:
  + \*Добавете функционалност за създаване на заявка, която връща всички служители, които отговарят на поне един от критериите. Например

ALL position = DEVELOPER skillLevel = JUNIOR yearsOfService > 2  
ANY position = DEVELOPER skillLevel = JUNIOR yearsOfService > 2

* + \*\*Добавете функционалност за промяна на служителите. Трябва да може да се зададе заявка, която да промени всички служители, отговарящи на критерия. Форматът трябва да е:

CHANGE <keyword> <value> FOR <заявка за търсене>

Например

CHANGE position MID FOR ALL position = DEVELOPER skillLevel = JUNIOR yearsOfService > 2

CHANGE salary 2000 FOR ANY skillLevel = JUNIOR yearsOfService < 2