# Java 2

## Oefeningen

---

**Opleiding:** **Bachelor Toegepaste Informatica**

Kies een item.

**Academiejaar: 2017-2018**

# Inhoud

# Oefeningen Java 2

## 1. Lambda en streams

### 1.1.Comparator

Vervang volgende code door een kortere versie, mbv lambda expressie

```
List<String> names = Arrays.asList("peter", "anna", "mike", "xenia");

Collections.sort(names, new Comparator<String>() {
    @Override
    public int compare(String a, String b) {
        return b.compareTo(a);
    }
});
```

### 1.2.Listener

Vervang volgende code door een kortere versie, mbv lambda expressie

```
public void Listener()
{
    JButton testButton = new JButton("Test Button");
    testButton.addActionListener(new ActionListener(){
        @Override public void actionPerformed(ActionEvent ae){
           System.out.println("Clicked anonymous class definition");
        }
    });

    JFrame frame = new JFrame("Listener Test");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.add(testButton, BorderLayout.CENTER);
    frame.pack();
    frame.setVisible(true);
}
```

## 1.3.Foreach op collections

Herschrijf onderstaande code zodat je gebruik maakt van een lambda expressie en de foreach functie voorzien bij de collections.

```
List<Person> lectorenLijst = new ArrayList<>();
lectorenLijst.add(new Person("Kelly", "Casal", "kelly.casalmosteiro@ap.be"));
lectorenLijst.add(new Person("Olga", "Coutrin", "olga.coutrin@ap.be"));
lectorenLijst.add(new Person("Philippe", "Possemiers", "philippe.possemiers@ap.be"));

System.out.println("\n=== E-Mail ===");

for (Person l: lectorenLijst) {
    System.out.println(l.getEmail());
}
```

## 1.4.Streams en filter icm lambda expressies

Pas de volgende code zo aan dat je werkt met streams, filter en lambda

```
List<Person> lectorenLijst = new ArrayList<>();
lectorenLijst.add(new Person("Kelly", "Casal", "kelly.casalmosteiro@ap.be"));
lectorenLijst.add(new Person("Olga", "Coutrin", "olga.coutrin@ap.be"));
lectorenLijst.add(new Person("Philippe", "Possemiers", "philippe.possemiers@ap.be"));


System.out.println("\n=== print Name with stream ===");
for (Person lector : lectorenLijst)
{
    if (lector.getEmail().equals("philippe.possemiers@ap.be"))
    {
        System.out.println(lector.getName());
        break;
    }

}
```

## 1.5..collect()

Pas de vorige code zo aan, dat je een nieuwe lijst maakt ahv de filter uit de vorige oefening.

## 1.6..findAny() , orElse()

Pas onderstaande code aan zodat je met stream, filter, findany en orElse de code korter en overzichtelijker kan maken.

```
List<Person> lectorenLijst = new ArrayList<>();
lectorenLijst.add(new Person("Kelly", "Casal", "kelly.casalmosteiro@ap.be"));
lectorenLijst.add(new Person("Olga", "Coutrin", "olga.coutrin@ap.be"));
lectorenLijst.add(new Person("Philippe", "Possemiers", "philippe.possemiers@ap.be"));

for (Person lector : lectorenLijst)
{
    if (lector.getEmail().equals("philippe.possemiers@ap.be") &&
lector.getName().equals("Philippe Possemiers"))
    {
        return lector;
    }
}
return null;
```

## 1.7.File streams en lambda

Lees de file van de 4 letter woorden in (zie bijlage oefeningen op learning.ap.be). Maak een nieuwe file, dewelke de 4 letterwoorden bevat die een a bevatten, schrijf ze in uppercase.

## 1.8.File streams en lambda

Lees de file van de 4 letter woorden in. Maak een nieuwe file, dewelke de 4 letterwoorden bevat die een a bevatten, schrijf ze alfabetisch.

## 1.9.File streams en lambda

Lees de file van de 4 letter woorden in. Maak een nieuwe file,met enkel de palindromen.

## 1.10.Filteren van steden

Steden klasse :

```
public class City {

    private String name;
    private long nrOfPeople;
    private long nrOfCrimes;

    public City(String name, long nrOfPeople, long nrOfCrimes) {
        this.setName(name);
        this.setNrOfPeople(nrOfPeople);
        this.setNrOfCrimes(nrOfCrimes);
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public long getNrOfPeople() {
        return nrOfPeople;
    }

    public void setNrOfPeople(long nrOfPeople) {
        this.nrOfPeople = nrOfPeople;
    }

    public long getNrOfCrimes() {
        return nrOfCrimes;
    }

    public void setNrOfCrimes(long nrOfCrimes) {
        this.nrOfCrimes = nrOfCrimes;
    }

    @Override
    public String toString() {
        return "City{" +
                "name='" + name + '\'' +
                ", nrOfPeople=" + nrOfPeople +
                ", nrOfCrimes=" + nrOfCrimes +
                '}';
    }
}
```

Maak een lijst van steden aan :

```
List<City> cities = new ArrayList<>();
cities.add(new City("New York",8550405, 86443));
cities.add(new City("Boston",754987, 3864));
cities.add(new City("LA",5439768, 1043));
cities.add(new City("Chicago",2720546, 87));
cities.add(new City("San Diego",1394928, 86443));
```

Zoek de stad/steden met meer dan 4.000.000 inwoners en minder dan 8.000 misdrijven. Druk de naam van de stad af met stream en lambda.