



Multithreading (deel 2)



**DE HOGESCHOOL
MET HET NETWERK**

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



1. Thread synchronization

1	169.150.56.187
2	55.160.224.224
3	142.17.210.166
4	161.65.49.56
5	188.120.26.52
6	254.225.197.92
7	243.50.97.113
8	226.98.115.212
9	89.30.197.140
10	86.127.239.167

1. Gegeven: bestanden met ip-adressen.
2. Bestudeer de klasse `NumberCounter` en test de functionaliteit uit voor het bestand `ip_addresses_20181201.csv`.
3. Zorg er nu voor dat de functionaliteit van de klasse `LetterCounter` in een aparte thread uitgevoerd kan worden.



1. Thread synchronization (2)

1	169.150.56.187
2	55.160.224.224
3	142.17.210.166
4	161.65.49.56
5	188.120.26.52
6	254.225.197.92
7	243.50.97.113
8	226.98.115.212
9	89.30.197.140
10	86.127.239.167

4. Start nu voor ieder bestand in de directory "resources/testdata" een nieuwe thread.

Tip: bekijk de volgende methoden in de klasse Files:

<code>static DirectoryStream<Path></code>	<code>newDirectoryStream(Path dir)</code> Opens a directory, returning a <code>DirectoryStream</code> to iterate over all entries in the directory.
<code>static DirectoryStream<Path></code>	<code>newDirectoryStream(Path dir, DirectoryStream.Filter<? super Path> filter)</code> Opens a directory, returning a <code>DirectoryStream</code> to iterate over the entries in the directory.

5. Sommeer de resultaten van alle threads en druk dit resultaat af.



2. Object locking

```
public class Koekjesdoos {  
    private int aantalKoekjes;  
  
    public Koekjesdoos(int aantalKoekjes) {  
        this.aantalKoekjes = aantalKoekjes;  
    }  
  
    public boolean neemKoekje() {  
        if (aantalKoekjes > 0) {  
            aantalKoekjes--;  
            return true;  
        }  
        return false;  
    }  
}
```



```

public class Kind extends Thread {
    private int aantalKoekjes;
    private Koekjesdoos koekjesdoos;
    private String naam;

    public Kind(String naam, Koekjesdoos koekjesdoos) {
        this.koekjesdoos = koekjesdoos;
        this.naam = naam;
    }

    @Override
    public void run() {
        while (koekjesdoos.neemKoekje()) {
            aantalKoekjes++;
            try {
                Thread.sleep(5);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println(naam + " at " + aantalKoekjes + " koekjes");
    }

    public int getAantalKoekjes() {
        return aantalKoekjes;
    }
}

```

```
public class KoekjesEten {
    public static void main(String[] args) {
        Koekjesdoos koekjesdoos = new Koekjesdoos(50);
        Kind[] kinderen = {
            new Kind("Bram", koekjesdoos),
            new Kind("Sophie", koekjesdoos),
            new Kind("Elke", koekjesdoos),
            new Kind("Robin", koekjesdoos),
            new Kind("Sammy", koekjesdoos),
            new Kind("Max", koekjesdoos)};
        for (int i = 0; i < kinderen.length; i++) {
            kinderen[i].start();
        }
        for (int i = 0; i < kinderen.length; i++) {
            try {
                kinderen[i].join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println("De kinderen aten: " +
            Arrays.stream(kinderen)
                .mapToInt(kind -> kind.getAantalKoekjes())
                .sum());
    }
}
```

```
KoekjesEten x
"C:\Program Files\Java\jdk-11\bin\java.exe" "-ja
Bram at 9 koekjes
Sophie at 9 koekjes
Max at 9 koekjes
Elke at 9 koekjes
Martien at 9 koekjes
Robin at 9 koekjes
De kinderen aten: 54

Process finished with exit code 0
```

aantalKoekjes = 1

neemKoekje()

if (aantalKoekjes > 0)

neemKoekje()

if (aantalKoekjes > 0)

aantalKoekjes--

[aantalKoekjes -> 0]

aantalKoekjes--

[aantalKoekjes -> -1]




```
KoekjesEten x
"C:\Program Files\Java\jdk-11\bin\java.exe" "-ja
Bram at 9 koekjes
Sophie at 9 koekjes
Max at 9 koekjes
Elke at 9 koekjes
Martien at 9 koekjes
Robin at 9 koekjes
De kinderen aten: 54

Process finished with exit code 0
```

```
KoekjesEten x
"C:\Program Files\Java\jd
Elke at 8 koekjes
Robin at 8 koekjes
Martien at 9 koekjes
Max at 9 koekjes
Sophie at 8 koekjes
Bram at 8 koekjes
De kinderen aten: 50
```

```
public class Koekjesdoos {
    private int aantalKoekjes;

    public Koekjesdoos(int aantalKoekjes) {
        this.aantalKoekjes = aantalKoekjes;
    }

    public synchronized boolean neemKoekje() {
        if (aantalKoekjes > 0) {
            aantalKoekjes--;
            return true;
        }
        return false;
    }
}
```

3. Timer en TimerTask

```
public class RepeatTask {
    public static void main(String[] args) {
        TimerTask repeatedTask = new TimerTask() {
            public void run() {
                System.out.println("Task performed on " +
                                   LocalDateTime.now());
            }
        };
        Timer timer = new Timer("Timer");
        long delay = 5000L;
        long period = 10000L;
        timer.scheduleAtFixedRate(repeatedTask, delay, period);
        System.out.println("Timer started " +
                           LocalDateTime.now());
    }
}
```



4. Parallel streams

```
public class ParallellStreams {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<Employee>();
        for (int i = 0; i < 100; i++) {
            employees.add(new Employee("A", 20000));
            employees.add(new Employee("B", 3000));
            employees.add(new Employee("C", 15002));
            employees.add(new Employee("D", 7856));
            employees.add(new Employee("E", 200));
            employees.add(new Employee("F", 50000));
        }
        long t1 = System.currentTimeMillis();
        System.out.println("Sequential Stream Count?= " +
            employees.stream().filter(e -> e.getSalary() > 15000).count());
        long t2 = System.currentTimeMillis();
        System.out.println("Sequential Stream Time Taken?= " + (t2 - t1) + "\n");

        t1 = System.currentTimeMillis();
        System.out.println("Parallel Stream Count?= " +
            employees.parallelStream().filter(e -> e.getSalary() > 15000).count());
        t2 = System.currentTimeMillis();
        System.out.println("Parallel Stream Time Taken?= " + (t2 - t1));
    }
}
```