

Java Threads

Lesen Sie die folgende Internetseite zu Threads und führen Sie die Beispiele durch:

https://www.w3schools.com/java/java_threads.asp

Erstellen Sie in Eclipse ein Projekt Threads.

Aufgabe 1)

Im Projekt Threads erstellen Sie ein Paket counting und implementieren Sie die Klasse **Zaehler** und die weiteren Klassen zu dieser Aufgabe.

```
class Zaehler extends Thread ❶
{
    private int zahl;

    Zaehler(String name, int z) ❷
    {
        super(name);
        zahl = z;
    }

    @Override public void run() ❸
    {
        for (int i = 1; i <= zahl; i++) ❹
        {
            System.out.println(getName() + ": " + i);
            try {
                TimeUnit.MILLISECONDS.sleep(100);
            }
            catch (InterruptedException ignore) {}
        }
    }
}
```

- ❶ Soll die Funktionalität einer Klasse parallel ablaufen können, muss sie Attribute und Operationen der Superklasse Thread erben.
- ❷ Im Konstruktor werden der Name des Threads und die Zahl gesetzt, bis zu der hochgezählt werden soll.
- ❸ Beim Aufruf der Klasse als Thread kommt per Definition ihre run-Operation zum Einsatz, die an dieser Stelle überschrieben wird.
- ❹ Die Operation zählt von 1 bis zahl. Nach jedem Hochzählen legt sie sich selbst für 100 Millisekunden schlafen.

Implementieren Sie die Klasse Aktivität, die eine main-Methode enthält. In der Main-Methode wird zuerst die Ausgabe „Zählen beginnt“ gemacht.

Deklarieren Sie zwei Instanzen **zaehler1** und **zaehler2** der Klasse **Zaehler**. Für den

Konstruktor verwenden Sie jeweils die folgenden Werten: „Thread A“ und 3 für `zaehler1` und „Thread B“ und 6 für `zaehler2`.

Lassen Sie `zaehler1` und `zaehler2` diese Methode `start()` aufrufen. Siehe auch www.3schools.com.... (oben).

Danach rufen `zaehler1` und `zaehler2` die Methode `join()`.

Jeder `join()`-Aufruf wartet, bis die entsprechende `run()`-Methode durchgelaufen ist. Nach dem `join()` auf beiden Threads ist garantiert, dass beide fertig sind.

Die letzte Anweisung ist die Ausgabe von „Zählen beendet“.

Die Ausgabe sieht z.B. so aus:

```
Zählen beginnt  
Thread A: 1  
Thread B: 1  
Thread A: 2  
Thread B: 2  
Thread A: 3  
Thread B: 3  
Thread B: 4  
Thread B: 5  
Thread B: 6  
Zählen beendet
```

- Was können wir feststellen?
- Ändern Sie die Werte für `Z`, was bemerken Sie?
- Ändern Sie auch die Werte in den Timer.

Implementieren Sie einen Konstruktor mit drei Parametern. Der Konstruktor soll zusätzlich einen Zeitwert für den Timer.

Aufgabe 2)

Erstellen Sie ein Paket `countinganddating`, fügen Sie in dieses Paket die Klassen zu dieser Aufgabe.

Implementieren Sie die folgenden Klassen:

a. Klasse `CounterCommand`

```

public class CounterCommand extends Thread{

    @Override
    public void run() {
        for(int i= 0; i <= 20; i++) {
            System.out.println("Zahl : " + i);
            try {
                TimeUnit.MILLISECONDS.sleep(75);
            }
            catch (InterruptedException ignore) {}
        }
    }
}

```

b. Implementieren Sie die Klasse **DateCommand** ähnlich wie die Klasse **CounterCommand**. Unterschied: Die Ausgabe in der **for**-Schleife soll die Ausgabe lauten „Date and Time“ plus die Ausgabe des aktuellen Datums und der aktuellen Uhrzeit.

Testen Sie die beiden Klassen. Starten Sie beide Threads, am Ende rufen beide Objekte die **join()**-Methode auf und die Ausgabe „Done“ wird ausgegeben.

Um das Wissen zu vertiefen, lesen Sie diese Seite:

<http://www.scalingbits.com/java/javakurs2/programmieren2/threads/programmierung>

Aufgabe 3)

Erstellen Sie im Eclipse im Projekt **Threads** ein Paket **threads1** und implementieren Sie in diesem Paket die Klassen aus den Beispielen von [scalingbits.com](http://www.scalingbits.com).

Aufgabe 4)

a) Implementieren Sie die Klasse **RamdomNumbers**, die von der Klasse **Thread** erbt. Diese Klasse hat die Instanzvariablen **amount** und **maxNumber** von Typ **Integer**. Über den Konstruktor werden den Variablen die Werte zugewiesen. **amount** gibt die Anzahl der Zufallszahlen, die erzeugt werden sollen. **maxNumber** gibt die Obergrenze für die Zufallszahlen. Ist **maxNumber** 100, dann sollen die Double-Zahlen von 1 bis 100 erzeugt werden.

Informieren Sie sich über die Klasse **Random** um die Zufallszahlen zu erzeugen.

Die überschriebene Methode **run()** gibt die Zufallszahlen aus wie folgt aus:

```

95.05987296272437
49.00616609346676
75.17605987591133
65.65418659865982
61.658230240875504

```

Nach jeder Zeile soll im try-Block 800 Millisekunden gewartet werden und im catch-Block der Fehler ignoriert werden.

Implementieren Sie die Testklasse und testen Sie die Ausgabe der Klasse **RamdomNumbers**.

b)

Implementieren Sie die Klasse **CharArray**, die von der Klasse **Thread** erbt. Der parameterlose Konstruktor erzeugt einen Instanzarray **charArray** der Größe 5x5. In dem Array werden die Buchstaben ,a‘ bis ,z‘ und ,A‘ bis ,Z‘ gespeichert werden. Jeder Platz im **charArray** enthält genau einen Buchstaben, der per Zufall ausgewählt wird.

Die überschriebene Method **run()** gibt die im Array gespeicherten Buchstaben wie folgt aus:

Y	M	e	A	b
I	p	h	q	q
d	s	W	V	f
U	v	d	Q	T
p	c	D	p	H

Nach jeder Zeile soll im try-Block 500 Millisekunden gewartet werden und im catch-Block der Fehler ignoriert werden.

Testen Sie die Ausgabe der Klasse **RamdomNumbers**.

c) In der Testklasse wird mit der Methode **start()** die **run()**-Methode aufgerufen. Wenn die Methode **join()** von den Threads aufgerufen wird, dann muss die Methode **main()** die Anweisung **throws InterruptedException** enthalten.