Timers en Timertasks in Java

Auteur: drs. M.S.L.F. Manssen

http://www.manssen.eu/

De TimerTask class in java.util maakt het mogelijk om een uit te voeren taak te beschrijven. Deze taak kan daarna op bepaalde tijden worden uitgevoerd met behulp van een Timer.

Om een bepaalde taak vast te leggen met een TimerTask, dient een subclass van TimerTask gemaakt te worden, met daarin de method `run`. Deze method `run` is in de Timerclass abstract, en moet dus gedefinieerd worden in de aan te maken subclass. In de `run` method wordt de uit te voeren taak gespecificeerd.

```
// Maak een subclass van TimerTask aan:
public class OnzeTimerTask extends TimerTask {

	OnzeTimerTaks() {
		// Gebruik de constructor voor voorbereidend werk.
	}

	void run() {
		// Hier komt de taak die moet worden uitgevoerd.
	}
}
```

Nu we een taak hebben vastgelegd, kunnen we deze gaan schedulen. Hiervoor maken we een timer aan:

```
timer = new Timer();
```

Nu zijn er verschillende mogelijkheden om `OnzeTimerTask` te schedulen. Het gaat vooral om overloaded methods. We zullen ze één voor één behandelen.

```
void schedule( TimerTask task, Date time )
```

De taak wordt eenmalig op de aangegeven tijd uitgevoerd.

```
void schedule ( TimerTask task, Date firsttime, long period )
```

De taak wordt voor het eerst uitgevoerd op `firsttime`, en daarna telkens herhaald uitgevoerd als er weer het aantal milliseconden verstreken is zoals aangegeven in `period`.

```
void schedule ( TimerTask task, long delay )
```

De taak wordt eenmalig uitgevoerd na de aangegeven `delay` in milliseconden.

```
void schedule ( TimerTask task, long delay, long period )
```

De taak wordt voor het eerst uitgevoerd na het door `delay` aangegeven aantal milliseconden, en daarna telkens herhaald uitgevoerd als het aantal milliseconden is verstreken dat in `period` is aangegeven.

```
void scheduleAtFixedRate ( TimerTask task, Date firsttime, long period )
```

De taak wordt voor het eerst uitgevoerd op `firsttime`, en daarna telkens herhaald uitgevoerd als er weer het aantal milliseconden verstreken is zoals aangegeven in `period`. Als er vertraging optreed tijdens het schedulen, worden de geplande taken later ingehaald.

```
void scheduleAtFixedRate( TimerTask task, long delay, long period )
```

De taak wordt voor het eerst uitgevoerd na het aantal milliseconden zoals aangegeven in `delay`, en

daarna telkens herhaald uitgevoerd als er weer het aantal milliseconden verstreken is zoals aangegeven in `period`. Als er vertraging optreed tijdens het schedulen, worden de geplande taken later ingehaald.

Laten we de TimerTask en Timer samenvoegen in een voorbeeld:

```
// Bestand: Voorbeeld.java

// Niet vergeten de juiste package te importeren.
// Benodigd voor Timer en Timertask:
import java.util.*;
// Benodigd om de naar System.out te schrijven:
import java.lang.*;

// Maak een subclass van TimerTask aan:
class OnzeTimerTask extends TimerTask {
	// We houden bij hoevaak de timertask is geactiveerd:
	private int ronde;

	OnzeTimerTask() {
		// In de constructor initialiseren we de ronde op 1
		ronde = 1;
	}

	public void run() {
		System.out.println( ronde );
		ronde++;
	}
}



public class Voorbeeld {
	static Timer timer;
	public static void main(String[] args) {
		timer = new Timer();
		// Laat de taak voor het eerst uitvoeren na 10 seconden (10000
		// milliseconden ), en daarna telkens na 1 seconde (1000
		// milliseconden )
		//
		timer.schedule ( new OnzeTimerTask(), 10*1000, 1*1000 );

	}

}
```