# Übung Woche 5

Patrick Bucher

23.03.2017

## Inhaltsverzeichnis

**1 Ballspiele** **1**
Warum überhaupt Threads verwenden? . . . . . 1
Wie werden die Threads erzeugt? . . . . . 2
Wann "sterben" die Treads? . . . . . 2
Wie wird die Darstellung der Bälle aktualisiert? . . . . . 2

**2 Hello World** **3**
Benötigen Sie überhaupt einen oder mehrere Threads? . . . . . 3
Welche Thread-Implementierung verwenden Sie? . . . . . 3
Wo starten Sie den/die Thread(s)? . . . . . 3
Wie viele Threads starten Sie? . . . . . 3
Wie wird die Darstellung der Welt aktualisiert? . . . . . 3
Wann sterben die Threads? . . . . . 3
Was passiert, wenn die Applikation gestoppt wird? . . . . . 3

**3 Bankgeschäfte** **4**

**4 Das Ende eines Threads** **4**

**5 Optional: JoinAndSleep** **4**

# 1 Ballspiele

## Warum überhaupt Threads verwenden?

Würden die Bälle synchron animiert, könnte das User-Interface beim Zeichnen keine Click-Events mehr entgegennehmen. Man könnte also, bei vielen zu zeichnenden Bällen, nicht mehr zuverlässig weitere Bälle hinzufügen.

Warum es für jeden Ball einen Thread geben soll, erschliesst sich mir jedoch nicht. Bei meinem Beispiel führt dies dazu, dass jeder Ball periodisch die ganze Zeichenfläche neu aufbauen lässt. Das ist ineffizient und hat den Nebeneffekt, dass die Animation bei vielen Bällen flüssiger läuft als bei wenigen, weil dann mehr Aktualisierungen nötig sind.

Würde man nur einen Zeichen-Thread verwenden, müsste weniger aktualisiert werden.

## Wie werden die Threads erzeugt?

Die Methode `createBall()` wird von einem Click-Event ausgelöst:

```
public void createBall(int x, int y) {
    int radius = getRandomNumber(20, 50);
    Color color = getRandomColor();
    Ball ball = new Ball(x, y, radius, color, this);
    balls.add(ball);
    ball.start();
    System.out.println("added ball, now " + balls.size() + " balls");
}
```

## Wann "sterben" die Treads?

Das "Leben" eines Balls hat zwei Phasen: das Fallen und das Erblassen. Sobald diese beiden Phasen durchschritten sind, läuft die Schleife in der `run()`-Methode des Balls nicht mehr weiter.

Wird ein Thread unterbrochen, wird das `interrupted`-Flag auf `true` gesetzt, wodurch auch die Schleife in der `run()`-Methode unterbrochen wird.

Die `paintComponent()`-Methode der Zeichenfläche prüft, ob der jeweils zu zeichnende Ball-Thread schon "erledigt", d.h. unterbrochen oder abgearbeitet, ist. Ist das der Fall, wird er aus der Collection mit zu zeichnenden Bällen entfernt.

## Wie wird die Darstellung der Bälle aktualisiert?

Jeder Ball ist zugleich ein Thread, der in einer Schleife die Zeichenfläche aktualisieren lässt und danach einige Millisekunden wartet (`Thread.sleep(10)`). Dazu ruft er die Methode `repaint()` von `JPanel` (bzw. dessen Vererbung `DrawingArea`) auf.

# 2 Hello World

## Benötigen Sie überhaupt einen oder mehrere Threads?

Ich benötige einen Thread. Ohne zusätzlichen Thread wäre die Benutzeroberfläche ständig mit Zeichnen und Warten beschäftigt und könnte so keine Klicks entgegennehmen.

## Welche Thread-Implementierung verwenden Sie?

Ich erbe von der Klasse `Thread` und implementiere das Interface `Runnable`. Das ist die einfachste Lösung für eine einfache Aufgabe. Eine Alternative dazu wäre der `SwingWorker`.

## Wo starten Sie den/die Thread(s)?

Ich starte den Thread, wenn das Fenster komplett geladen ist. Dazu implementiere ich einen `WindowListener`, der auf den `windowActivated`-Event des Fensters wartet und dann den Thread erstellt und startet.

Der Thread darf nicht im Konstruktor gestartet werden, da zu diesem Zeitpunkt noch nicht alle Ressourcen bereit sein könnten.

## Wie viele Threads starten Sie?

Ich starte nur einen Thread.

## Wie wird die Darstellung der Welt aktualisiert?

In der `run()`-Methode des Threads wird je nach Laufrichtung der Index der Bildliste hoch- oder runtergezählt. Das so ermittelte Bild wird der Zeichenfläche als Parameter übergeben, welche dann ihre Anzeige mittels `repaint()` und `paintComponent()` aktualisiert.

## Wann sterben die Threads?

Der Thread kann nur sterben, wenn er von aussen unterbrochen wird. Das sollte grundsätzlich nie der Fall sein. Somit läuft der Thread so lange wie die Applikation läuft.

## Was passiert, wenn die Applikation gestoppt wird?

Mittels `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` wird das Programm beendet, wenn das Fenster geschlossen wird. Da der gestartete Thread eine Eigenschaft eines der GUI-Komponenten ist, wird dieser mit dem GUI zusammen “entsorgt”.

## 3 Bankgeschäfte

TODO

## 4 Das Ende eines Threads

TODO

## 5 Optional: JoinAndSleep