# Handzettel 5 „ProgressBar,“ Seite 1 von 3

Tutorium Programmiersprachen 2, WS 2006/07
Arne Johannessen
Hochschule Karlsruhe – Technik und Wirtschaft

```java
/* Dateiname: ProgressBar.java
 * Programmiersprachen 2, Hochschule Karlsruhe
 * ProgressBar: Vorlage zu Kapitel 3, Aufgabe 2
 * erstellt von: Arne Johannessen, 2006-11-13
 */

// verwendete Pakete importieren
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

// diese Klasse (ProgressBar) ist ein JFrame
public class ProgressBar extends JFrame {

    // Deklaration von Instanzvariablen
    Container contentPane = super.getContentPane();
    ButtonListener buttonListener = new ButtonListener();

    // Deklaration der Steuerelemente als Instanzvariablen
    JButton startKnopf;
    JProgressBar fortschrittsAnzeige;

    // Konstruktor
    public ProgressBar () {
        super();

        // Fenster initialisieren, erster Teil
        super.setTitle("JFrame-ProgressBar");
        super.setSize(350, 250);
        super.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        contentPane.setLayout(new FlowLayout());

        // neue Instanzen der Steuerelemente erstellen
        startKnopf = new JButton("Start");
        fortschrittsAnzeige = new JProgressBar();

        // Fortschritts-Anzeige initialisieren
        // ...

        // Steuerelemente zum Fensterinhalt hinzufügen
        contentPane.add(startKnopf);
        contentPane.add(fortschrittsAnzeige);

        // Steuerelemente für Ereignis-Behandlung registrieren
        startKnopf.addActionListener(buttonListener);

    }

    // Ereignis-Behandlung ist hier mit innerer Klasse gelöst
    class ButtonListener implements ActionListener {

        // Java ruft die actionPerformed-Methode auf, wenn die angeklickten
        // Knöpfe zuvor mit addActionListener registriert worden sind
        public void actionPerformed (ActionEvent ereignis) {

            // prüfen, bei welchem Steuerelement das Ereignis stattgefunden hat
            if (ereignis.getSource() == startKnopf) {
                zeigeFortschritt();
            }

        }
    }

    // Fortschritts-Anzeige einmal von links nach rechts laufen lassen
    public void zeigeFortschritt () {
        // ...
    }

    // main-Methode; wird ausgeführt, wenn man diese Klasse als Programm startet
    public static void main (String[] args) {

        // neue Fenster-Instanz erstellen
        ProgressBar hauptfenster = new ProgressBar();  // Konstruktor-Aufruf!

        // Fenster initialisieren, zweiter Teil
        hauptfenster.setLocation(250, 350);
        hauptfenster.setVisible(true);
    }

}
```

Abschnitt Instanzvariablen:
- JProgressBar-Variable deklarieren

Abschnitt Konstruktor:
- neue JProgressBar-Instanz erstellen und sie der Variablen zuweisen
- ggf. weitere Initialisierungen der neuen JProgressBar vornehmen (Maximalwert, Abmessungen etc.)
- neue JProgressBar zum Fensterinhalt hinzufügen

Abschnitt Ereignis-Behandlung:
(prinzipiell richtiger Ansatz)

zeigeFortschritt() ist die Methode, in der das komplette Ändern der Progress-Bar stattfindet; benötigt werden darin:
- eine Schleife zum Hochzählen des Zustands
- eine Verzögerung, damit die Schleife nicht sofort fertig ist
  → `Thread.sleep(10);` verzögert um 10 ms

Problem dieses prinzipell richtigen Ansatzes: Temperaturstreifen bewegt sich nicht sichtbar, sondern „springt“ nach Terminierung der Schleife abrupt vom Anfang zum Ende

Tutorium Programmiersprachen 2, WS 2006/07
Arne Johannessen
Hochschule Karlsruhe – Technik und Wirtschaft

Grund für das Problem:
Java arbeitet bei graphischen Oberflächen grundsätzlich mit verschiedenen Nebenläufen (engl. Thread). Es ist eine Besonderheit von Java, dass aus dem Ereignis-Behandlungs–Thread heraus die graphische Oberfläche nicht direkt aktualisiert werden kann. Eine mögliche Lösung ist, auf Threads komplett zu verzichten (dabei muss auch auf Interaktivität und Ereignis-Behandlung verzichtet werden):

```
/* Dateiname: ProgressBarSynchron.java
 * Programmiersprachen 2, Hochschule Karlsruhe
 * ProgressBarSynchron: synchrone Lösung zu Kapitel 3, Aufgabe 2
 * erstellt von: Arne Johannessen, 2006-11-13
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ProgressBarSynchron extends JFrame {

    // Deklaration von Instanzvariablen
    Container contentPane = super.getContentPane();
//  ButtonListener buttonListener = new ButtonListener();
//  JButton startKnopf;
    JProgressBar fortschrittsAnzeige;

    // Konstruktor
    public ProgressBarSynchron () {
        super();

        // Fenster initialisieren, erster Teil
        super.setTitle("JFrame-ProgressBar synchron");
        super.setSize(350, 250);
        super.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        contentPane.setLayout(new FlowLayout());

        // neue Instanzen der Steuerelemente erstellen
//      startKnopf = new JButton("Start");
        fortschrittsAnzeige = new JProgressBar();

        // Fortschritts-Anzeige initialisieren
        // ...

        // Steuerelemente zum Fensterinhalt hinzufügen
//      contentPane.add(startKnopf);
        contentPane.add(fortschrittsAnzeige);

        // Steuerelemente für Ereignis-Behandlung registrieren
//      startKnopf.addActionListener(buttonListener);

    }

/*  class ButtonListener implements ActionListener {
        public void actionPerformed (ActionEvent ereignis) {
            if (ereignis.getSource() == startKnopf) {
                zeigeFortschritt();
            }
        }
    }
*/
    // Fortschritts-Anzeige einmal von links nach rechts laufen lassen
    public void zeigeFortschritt () {
        // ...
    }

    public static void main (String[] args) {
        ProgressBarSynchron hauptfenster = new ProgressBarSynchron();

        // Fenster initialisieren, zweiter Teil
        hauptfenster.setLocation(250, 350);
        hauptfenster.setVisible(true);

        // Fortschrittsanzeige synchron laufen lassen
        hauptfenster.zeigeFortschritt();
    }

}
```

Abschnitt Instanzvariablen:
- startKnopf brauchen wir nicht mehr und damit auch keinen ButtonListener

Abschnitt Konstruktor:
keine besonderen Änderungen über das Entfernen des Buttons startKnopf hinaus

Abschnitt Ereignis-Behandlung:
gibt es nicht, da keine Ereignisse durch Steuerelemente ausgelöst werden

zeigeFortschritt() liegt jetzt direkt in der main-Methode; weil es während der main-Methode noch keine Nebenläufe gibt, wird das Problem umgangen

Tutorium Programmiersprachen 2, WS 2006/07
Arne Johannessen
Hochschule Karlsruhe – Technik und Wirtschaft

Beste Lösung:
Statt Nebenläufigkeit zu verhindern, lieber einen eigenen Nebenlauf nur für die Progress-Bar erzeugen. Ist einfacher, als es klingt.

```
/* Dateiname: ProgressBarThread.java
 * Programmiersprachen 2, Hochschule Karlsruhe
 * ProgressBarThread: nebenläufige Lösung zu Kapitel 3, Aufgabe 2
 * erstellt von: Arne Johannessen, 2006-11-13
 */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ProgressBarThread extends JFrame implements Runnable {

    // Deklaration von Instanzvariablen
    Container contentPane = super.getContentPane();
    ButtonListener buttonListener = new ButtonListener();
    JButton startKnopf;
    JProgressBar fortschrittsAnzeige;

    // Konstruktor
    public ProgressBarThread () {
        super();

        super.setTitle("JFrame-ProgressBar nebenläufig");
        super.setSize(350, 250);
        super.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        contentPane.setLayout(new FlowLayout());

        startKnopf = new JButton("Start");
        fortschrittsAnzeige = new JProgressBar();
        // ...
        contentPane.add(startKnopf);
        contentPane.add(fortschrittsAnzeige);
        startKnopf.addActionListener(buttonListener);
    }

    // Ereignis-Behandlung ist hier mit innerer Klasse gelöst
    class ButtonListener implements ActionListener {

        public void actionPerformed (ActionEvent ereignis) {
            if (ereignis.getSource() == startKnopf) {
                // Nebenlauf (Thread) starten
                Thread thread = new Thread(ProgressBarThread.this);
                thread.start();
            }
        }
    }

    // Implementation des Nebenlaufs (Threads) dieser Klasse
    public void run () {
        zeigeFortschritt();
    }

    // Fortschritts-Anzeige einmal von links nach rechts laufen lassen
    public void zeigeFortschritt () {
        // ...
    }

    public static void main (String[] args) {
        ProgressBarThread hauptfenster = new ProgressBarThread();
        hauptfenster.setLocation(250, 350);
        hauptfenster.setVisible(true);
    }

}
```

Klassendeklaration:
- um diese Klasse als eigenen Thread nutzen zu können, muss sie das Interface „Runnable“ implementieren

Abschnitt Ereignis-Behandlung:
- statt direkt die Progress-Bar laufen zu lassen, wird jetzt hier der Thread erzeugt und dann gestartet
- `ProgressBarThread.this` ist eine Referenz auf die aktuelle Instanz der Klasse ProgressBarThread (eine Besonderheit von inneren Klassen)

zusätzliche Methode:
- das Interface Runnable erzwingt eine Methode mit der Signatur `public void run ()`; diese Methode wird beim Start des Threads nebenläufig aufgerufen
- hierher gehört das Ändern der Progress-Bar (mit `zeigeFortschritt()`)