Threads

Gerd Bohlender

Institut für Angewandte und
Numerische Mathematik

Vorlesung: Einstieg in die Informatik mit Java

27.06.07

Übersicht

Eigenschaften von Threads

- „Parallele“ Ausführung von Programmcode ähnlich wie bei Prozessen (z.B. mehreren „gleichzeitig“ laufenden Programmen)
- Im Unterschied zu Prozessen haben zusammengehörige Threads eines Programms gemeinsamen Speicher, etc.

Eigenschaften von Threads

- „Parallele“ Ausführung von Programmcode ähnlich wie bei Prozessen (z.B. mehreren „gleichzeitig“ laufenden Programmen)
- Im Unterschied zu Prozessen haben zusammengehörige Threads eines Programms gemeinsamen Speicher, etc.
- Vorteil: Dadurch ist die Verwaltung effizienter, der Wechsel einer CPU zwischen verschiedenen Threads geht sehr schnell

Eigenschaften von Threads

- „Parallele“ Ausführung von Programmcode ähnlich wie bei Prozessen (z.B. mehreren „gleichzeitig“ laufenden Programmen)
- Im Unterschied zu Prozessen haben zusammengehörige Threads eines Programms gemeinsamen Speicher, etc.
- Vorteil: Dadurch ist die Verwaltung effizienter, der Wechsel einer CPU zwischen verschiedenen Threads geht sehr schnell
- Nachteil: es gibt keinen Schutz, alle Thread dürfen auf die gleichen Variablen zugreifen (im Rahmen der Java-Sichtbarkeitsregeln)

Eigenschaften von Threads

- „Parallele“ Ausführung von Programmcode ähnlich wie bei Prozessen (z.B. mehreren „gleichzeitig“ laufenden Programmen)
- Im Unterschied zu Prozessen haben zusammengehörige Threads eines Programms gemeinsamen Speicher, etc.
- Vorteil: Dadurch ist die Verwaltung effizienter, der Wechsel einer CPU zwischen verschiedenen Threads geht sehr schnell
- Nachteil: es gibt keinen Schutz, alle Thread dürfen auf die gleichen Variablen zugreifen (im Rahmen der Java-Sichtbarkeitsregeln)

Arbeiten mit Threads

- Klasse ist von java.util.Thread abgeleitet
- Sie implementiert die Methode

```
public void run ()
```

Arbeiten mit Threads

- Klasse ist von java.util.Thread abgeleitet
- Sie implementiert die Methode

  ```
  public void run ()
  ```

- Mit einer Instanz der Klasse wird die Methode `start()` aufgerufen, damit wird `run()` „parallel“ ausgeführt

Arbeiten mit Threads

- Klasse ist von java.util.Thread abgeleitet
- Sie implementiert die Methode

  ```
  public void run ()
  ```

- Mit einer Instanz der Klasse wird die Methode `start()` aufgerufen, damit wird `run()` „parallel“ ausgeführt
- `run()` enthält eine Endlosschleife mit regelmäßig zu bearbeitenden Befehlen

Arbeiten mit Threads

- Klasse ist von java.util.Thread abgeleitet
- Sie implementiert die Methode
  ```
  public void run ()
  ```
- Mit einer Instanz der Klasse wird die Methode `start()` aufgerufen, damit wird `run()` „parallel“ ausgeführt
- `run()` enthält eine Endlosschleife mit regelmäßig zu bearbeitenden Befehlen
- Zur zeitlichen Ablaufsteuerung verwendet `run()` die Methode
  ```
  sleep(long);
  ```
  deren Parameter die gewünschte Pausenlänge in Millisekunden angibt

Arbeiten mit Threads

- Klasse ist von java.util.Thread abgeleitet
- Sie implementiert die Methode
  `public void run ()`
- Mit einer Instanz der Klasse wird die Methode `start()` aufgerufen, damit wird `run()` „parallel“ ausgeführt
- `run()` enthält eine Endlosschleife mit regelmäßig zu bearbeitenden Befehlen
- Zur zeitlichen Ablaufsteuerung verwendet `run()` die Methode
  `sleep(long);`
  deren Parameter die gewünschte Pausenlänge in Millisekunden angibt
- Dies ist meistens in einem `try ... catch` Block enthalten

Arbeiten mit Threads

- Klasse ist von java.util.Thread abgeleitet
- Sie implementiert die Methode
  `public void run ()`
- Mit einer Instanz der Klasse wird die Methode `start()` aufgerufen, damit wird `run()` „parallel“ ausgeführt
- `run()` enthält eine Endlosschleife mit regelmäßig zu bearbeitenden Befehlen
- Zur zeitlichen Ablaufsteuerung verwendet `run()` die Methode
  `sleep(long);`
  deren Parameter die gewünschte Pausenlänge in Millisekunden angibt
- Dies ist meistens in einem `try ... catch` Block enthalten

Typisches Thread-Applet

```
import java.util.*;
public class Uhrzeit extends Thread {
  public static void main (String[] args) {
    Uhrzeit uhr = new Uhrzeit();
    uhr.start();
  }
  public void run () {
    while (true) { // Endlosschleife
      long sek = System.currentTimeMillis() / 1000 % 86400;
      System.out.println ("Systemzeit in Sekunden: " + sek);
      try {
        sleep(1000); // 1 Sekunde warten
      } catch (InterruptedException e) { }
    }
  }
}
```

Das Interface Runnable

- Wird eine andere Basisklasse benötigt, kann Thread nicht angegeben werden – Beispiel: Applets
- Dann verwendet man das Interface Runnable

Das Interface Runnable

- Wird eine andere Basisklasse benötigt, kann Thread nicht angegeben werden – Beispiel: Applets
- Dann verwendet man das Interface Runnable
- Die Methode `run()` wird wie bei Thread implementiert

Das Interface Runnable

- Wird eine andere Basisklasse benötigt, kann Thread nicht angegeben werden – Beispiel: Applets
- Dann verwendet man das Interface Runnable
- Die Methode `run()` wird wie bei Thread implementiert
- In der Klasse erzeugt man ein Objekt vom Typ Thread und übergibt als Argument eine Referenz auf eine Instanz der gerade definierten Klasse, z.B.

```
Thread t = new Thread (this);
```

Das Interface Runnable

- Wird eine andere Basisklasse benötigt, kann Thread nicht angegeben werden – Beispiel: Applets
- Dann verwendet man das Interface Runnable
- Die Methode `run()` wird wie bei Thread implementiert
- In der Klasse erzeugt man ein Objekt vom Typ Thread und übergibt als Argument eine Referenz auf eine Instanz der gerade definierten Klasse, z.B.

```
Thread t = new Thread (this);
```

- Der Thread wird gestartet mit

```
t.start();
```

Das Interface Runnable

- Wird eine andere Basisklasse benötigt, kann Thread nicht angegeben werden – Beispiel: Applets
- Dann verwendet man das Interface Runnable
- Die Methode `run()` wird wie bei Thread implementiert
- In der Klasse erzeugt man ein Objekt vom Typ Thread und übergibt als Argument eine Referenz auf eine Instanz der gerade definierten Klasse, z.B.

```
Thread t = new Thread (this);
```

- Der Thread wird gestartet mit

```
t.start();
```

Typisches Beispiel zu Runnable

```
import java.applet.*;
import java.awt.*;
import java.util.*;
public class UhrzeitApplet extends Applet implements Runnable {
  Label label;
  Thread thread;
  public void init () {
    add (label = new Label ("00:00:00"));
    thread = new Thread(this);
    thread.start();
  }
  public void run () {
    while (true) { // Endlosschleife
      long sek = System.currentTimeMillis() / 1000 % 86400;
      label.setText ("Systemzeit in Sekunden: " + sek);     // Sekunden
      try {
        Thread.sleep(1000); // 1 Sekunde warten
      } catch (InterruptedException e) { }
    }
  }
}
```