10. Übungsblatt „Programmiersprachen“ WS 97/98

Leif Kornstaedt, Prof. Gert Smolka

Ausgabe: 22. Januar 1998 **Abgabe:** 2. Februar 1998

Auf diesem Übungsblatt sollen Sie ein Java-Programm schreiben, das sowohl als Applet als auch als Applikation gestartet werden kann. Eine beliebig einstellbare Anzahl nebenläufiger Threads soll Aufträge aus einem beschränkten Puffer lesen und abarbeiten. Jeder Auftrag wird dabei durch eine Zeichenkette und eine ganze Zahl gegeben; die Abarbeitung besteht darin, den Text als Laufschrift in einer graphischen Komponente anzuzeigen, wobei die ganze Zahl als Dauer in Millisekunden interpretiert wird und die Geschwindigkeit der Laufschrift regelt. Über eine Benutzeroberfläche können Aufträge generiert und in den beschränkten Puffer eingetragen werden.

Um diese Aufgabe lösen zu können, werden Sie kaum darum herum kommen, Klassen und Methoden in den Referenzdokumenten zum Java-API nachzuschlagen. Dieses finden Sie im WWW zum Beispiel unter der Adresse `http://java.sun.com/products/jdk/1.1/docs/api/packages.html`. Bitte geben Sie acht, daß Sie das API in der Version 1.1 verwenden. Auf den CIP-Rechnern ist dieses installiert.

X.1: Ein Laufschrift-Thread (12 Punkte)

Implementieren Sie eine Klasse `Scroller` mit folgendem Konstruktor:

```
public Scroller(BoundedBuffer buffer) { ... }
```

Der Konstruktor soll eine `Canvas`-Komponente erzeugen und einen neuen Thread starten, der in einer Schleife Aufträge aus dem `BoundedBuffer` anfordert. Jeder Auftrag besteht aus einem Paar (`String` s, `long` p). Der Text s soll vertikal in der Mitte des `Canvas` angezeigt werden; die horizontale Position soll sich periodisch ändern, so daß der Text von rechts in die Komponente hinein„*scroll*“t und links verschwindet. Die Periode ist durch p in Millisekunden gegeben. Verwenden Sie die `FontMetrics` des aktuellen Zeichensatzes, um die Position des Textes zu berechnen.

Wenn der Text nicht mehr in der graphischen Komponente sichtbar ist, soll der nächste Auftrag angefordert werden.

Bemerkungen:

- Es bietet sich an, die Klasse `Scroller` von `java.awt.Canvas` abzuleiten und zusätzlich die Schnittstelle `Runnable` zu implementieren. Dann reicht es, wenn Sie die Methoden `paint`, `run` sowie eventuell `update` und `getPreferredSize` überschreiben.
- Eine nebenläufige Implementierung der Klasse `BoundedBuffer` wurde in der Vorlesung vorgestellt. Sie können diese für ihre Implementierung wiederverwenden oder auch – falls nötig – modifizieren.

- Um eine flüssige Animation zu gewährleisten, können Sie die Technik des *double buffering* anwenden, die beispielsweise im Buch *The Java Tutorial* von Mary Campione und Kathy Walrath beschrieben wird, erschienen in *The Java Series*.

X.2: Benutzerschnittstelle (7 Punkte)

Definieren Sie eine Klasse `QueueControls` wie folgt:

```
class QueueControls extends java.awt.Panel {
  public QueueControls(BoundedBuffer buffer) { ... }
  ...
}
```

Der Konstruktor erzeuge in dem `Panel` zwei Textfelder (`java.awt.TextField`) und einen `Button`. Wird der Button gedrückt, so wird aus dem einen Textfeld eine Zeichenkette gelesen, aus dem anderen ein `long`-Wert, und ein Paar aus den beiden in den als Argument übergebenen beschränkten Puffer geschrieben. Über diese Klasse lassen sich also Aufträge erteilen.

X.3: Applets und Applikationen ($9 + 5$ Punkte)

In dieser Aufgabe sollen die in den vorigen Aufgaben bereitgestellten Teile zu einem Ganzen zusammengefügt werden.

a. Schreiben Sie eine Klasse `ScrollQueue`, die von `java.applet.Applet` abgeleitet ist und die zugehörigen Methoden implementiert. Dieses Applet soll zwei Parameter namens `n` und `m` erhalten; `n` gibt die Anzahl der Laufschrift-Threads an, die erzeugt werden sollen, und `m` die Größe des beschränkten Puffers, über den die Klassen aus den beiden vorhergehenden Aufgaben kommunizieren. Die `init`-Methode soll den beschränkten Puffer, die Laufschrift-Threads und `QueueControls` instanziieren und die graphischen Komponenten im Applet plazieren. Nachdem die Methode `start` aufgerufen wurde, soll alles zur Steuerung des Systems bereit sein.

b. Implementieren Sie eine Applikation, die sich genauso verhält wie das Applet aus der vorigen Teilaufgabe. Verwenden Sie dabei soviele Methoden des Applets wie möglich, um Code-Duplikation zu vermeiden. Ihre Applikation sollte die Parameter `n` und `m` auf der Kommandozeile erwarten und sinnvoll darauf reagieren, wenn sie mit einer ungültigen Kommandozeile aufgerufen wird.

   Hinweis: Bedenken Sie, daß Sie hierzu das Toplevel-Fenster der Applikation selbst erzeugen müssen. Dies ist in der Vorlesung gezeigt worden.