Prof. Kastens und Mitarbeiter

Vorlesung Software-Entwicklung II SS 2004 - Lösung zu Blatt 6

Lösung zu Aufgabe 12

a) An `start`, `resume`, `join` und `stop` sind immer 2 Prozesse beteiligt.

Da Prozesse sich auch selbst anhalten können, können bei `suspend` ein oder auch zwei Prozesse beteiligt sein.

Bei `run` und `sleep` ist immer nur ein Prozeß beteiligt.

b) Auszug aus der Beschreibung der Java-Standardbibliothek:

- Applets:
  - start: Called by the browser or applet viewer to inform this applet that it should start its execution. It is called after the init method and each time the applet is revisited in a Web page.
  - stop: Called by the browser or applet viewer to inform this applet that it should stop its execution. It is called when the Web page that contains this applet has been replaced by another page, and also just before the applet is to be destroyed.
- Threads:
  - start: Causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread. The result is that two threads are running concurrently: the current thread (which returns from the call to the start method) and the other thread (which executes its run method).
  - stop: Forces the thread to stop executing.

c) Aus der Beschreibung der Java-Standardbibliothek:

Deprecated. This method is inherently unsafe. Stopping a thread with Thread.stop causes it to unlock all of the monitors that it has locked. If any of the objects previously protected by these monitors were in an inconsistent state, the damaged objects become visible to other threads, potentially resulting in arbitrary behavior.

Alternative:

Many uses of stop should be replaced by code that simply modifies some variable to indicate that the target thread should stop running. The target thread should check this variable regularly, and return from its run method in an orderly fashion if the variable indicates that it is to stop running.

Siehe auch Folie Folie 135

d) push und pop ändern die Verzeigerung der Stack-Elemente. Damit sie konsistent bleibt, müssen die Methoden als synchronized charakterisiert werden.

top und empty ändern die Verzeigerung der Stack-Elemente nicht. Deshalb brauchen sie nicht als synchronized charakterisiert zu werden.

e) Die Implementierung ist in der Datei `StackExample.java` verfügbar.

Lösung zu Aufgabe 14

- Problem 1: Die Abfrage freeSeats(1) und die Belegung reserveSeats(3) können unkontrolliert verzahnt werden. Folge: Inkonsistenz. Abhilfe: Bedingungssynchronisation (auch wenn die Aufgabenstellung davon abrät)
- Problem 2: Inkonsistente Parameter der Aufrufe: freeSeats(1) und reserveSeats(3)
- Problem 3: freeSeatsLeft wird nie verkleinert; wenn einmal seats > freeSeatsLeft gilt, gilt es immer.
- Problem 4: stopIt hält die Prozesse nicht endgültig an - Missbrauch des Schemas.

- Problem 5: Prozesse in der synchronized Methode freeSeats anzuhalten ist nicht sinnvoll, denn es ist sowieso gerade ein Prozess im Monitor.
- Problem 6: Das Programm terminiert durch Schließen des Ausgabefensters. Dabei werden die Prozesse gewaltsam beendet.
- Problem 7: Unübersichtliche Struktur. besser: Eine Monitor-Klasse verwaltet freeSeatsLeft; Prozess-Klasse nicht einbetten.
- Problem 8: sleep ist eine Klassenmethode; t.sleep(1000) ist irreführend; es wirkt auf den Prozess, der es ausführt, unabhängig vom Wert von t.
- Problem 9: freeSeatsLeft = (new Integer(totalSeats)).intValue(); ist Unsinn; besser: freeSeatsLeft = totalSeats

Hier finden Sie die Folien aus der Zentralübung `zue6.pdf`.