Übungsblatt 8

Ausgabe: 24.06.2008
Abgabe: 02.07.2008, 16 Uhr

Aufgabe 1 (6 P)

Implementieren Sie nun verallgemeinerte P()- und V()-Operationen, mit denen es möglich ist, mehrere Semaphoren auf einmal zu manipulieren. Diese neuen P() und V() Operationen sollen atomar sein und so Mehrobjektsperren ermöglichen. Die Ausführung von *V(semas,n)* erhöht den internen Zähler aller Elemente von *semas* um *n*. Die Ausführung von *P(semas,n)* erniedrigt den internen Zähler von allen Elementen von *semas* um den Wert *n*. Falls bei einem der Elemente der interne Zähler kleiner als *n* ist, blockiert die Ausführung von *P()* solange, bis alle internen Zähler groß genug sind. Beim Blockieren von P() ist darauf zu achten, dass alle Elemente von *semas* unverändert bleiben, bis alle Zähler groß genug sind.

Benutzen Sie dabei Java Generics (mehr Informationen dazu im Überblick bei http://java.sun.com/j2se/1.5.0/docs/guide/language/generics.html und im Detail bei http://java.sun.com/j2se/1.5/pdf/generics-tutorial.pdf) und erweitern Sie den folgenden Klassenrumpf, ohne dabei neue öffentliche Methoden einzuführen. Benutzen Sie Iteratoren, um auf die Elemente von Mengen zuzugreifen.

```
public class MultiSemaphore {
  ...
  // erzeugt ein neues Semaphor, dessen interner Zähler den Wert von
  // "init" hat
  public MultiSemaphore(int init) { ... }

  static public void P(Set<MultiSemaphore> semas, int n) {...}
  static public void V(Set<MultiSemaphore> semas, int n) {...}
}
```

Aufgabe 2 (10 P)

Beim Beispiel Betriebsmittel-Pool `Resources` aus der Vorlesung (3.2.2.3) können Prozesse jeweils eine bestimmte Anzahl von Betriebsmitteln anfordern und später wieder freigeben. Die Zahlen dürfen sich dabei durchaus unterscheiden, es ist z.B. denkbar, daß ein Prozess zuerst in mehreren Schritten einzelne Betriebsmittel anfordert und dann alle auf einmal wieder freigibt (oder auch umgekehrt).

Im vorgestellten Beispiel wurde nur die Synchronisationsstruktur gezeigt. Für eine Realisierung ist es jedoch aus Fairness-Gründen sinnvoll, die wartenden Prozesse und ihre Anforderungen in einer geeigneten Datenstruktur zu verwalten. Damit wird es möglich, Betriebsmittel-Zuteilungen auf der Basis einer bestimmten Politik durchzuführen. Mögliche Alternativen wären z.B.:

- Wer am längsten wartet, wird als nächstes bedient (FCFS).
- Die kleinsten Anforderungen werden zuerst bedient.
- Die größten Anforderungen werden zuerst bedient.
- ...

Um gezielt einzelne wartende Prozesse bedienen zu können ist es sinnvoll, für jeden Prozess ein eigenes Ereignisobjekt zu verwenden. In Kapitel 3.2.2.1 der Vorlesung wurden `Event`-Objekte mit `wait/signal`-Methoden vorgestellt, die sich hierfür gut verwenden lassen. Beachten Sie die dort beschriebenen Fairnessregeln.

Implementieren Sie in Pseudojava, mit Hilfe der in der Vorlesung eingeführten Systemklasse `Event`, den Betriebsmittel-Pool `Resources` für die Zuweisungs-Politik FCFS (*First Come First Serve*) auf der Basis von `Event`-Objekten. Geben Sie je eine Lösung für die drei möglichen Signal-Varianten an:

1. *signal-and-wait*
2. *signal-and-continue*
3. *signal-and-return*

Benutzen Sie für jede ihrer Implementierungen folgenden java-ähnlichen Klassenrumpf:

```
public MONITOR Resources {
    class ResourceRequest {
       public final int claim;
       public final Event event = new Event();
       public ResourceRequest(int claim) {this.claim = claim; }
    }

    private final Vector requests = new Vector();

    private final int max = 100;      //arbitrary resource limit
    private int available = max;


    //insert methods request & release (siehe VL Skript)

}
```

Hinweise zur Abgabe:

Die Abgabe der Lösungen erfolgt auf zwei Wegen:

- Abgabe auf Papier:

  Bitte schreiben Sie zur jeder Aufgabe eine Dokumentation, in der Sie anhand von *relevanten* Codefragmenten Ihre Implementierung erläutern. Die Codefragmente sollten möglichst kurz gehalten werden und nur der Orientierung für den Leser dienen; entscheidend sind Ihre Erläuterungen, was diese Fragmente machen und wieso Sie sich für diese Art der Implementierung entschieden haben. Die Dokumentation ist bis Mittwoch 16 Uhr in den Tutorenfächern abzugeben.

- Abgabe per E-Mail:

  Schicken Sie ein ZIP-Archiv an Ihren Tutor, welches den vollständigen Quellcode sowie ausführbare JAR-Dateien für die jeweiligen Aufgabenteile enthält. Der Betreff der E-Mail sollte wie folgt aussehen: "[ALP4] Blatt X - Namen Y".

Zum Bestehen des Übungsblattes müssen mindestens 50% der Punkte erreicht werden.