Aufgabe K14: Semaphor mit Zusatzmethoden

Ausgangspunkt ist ein als Kernobjekt implementiertes Semaphor mit den Operationen `P` und `V`, wie im Begleitbuch beschrieben. Die Methoden heißen hier `Psyn` und `Vasyn`.

Attribute:

- **Zähler**, nicht negativ. Initialwert beim Erzeugen anzugeben.
- **Wartemenge** für Prozesse. Initial leer.

Dateninvariante:

- Wenn **Zähler** > 0, dann **Wartemenge** leer.
  (d.h. ungültiger Fall: **Zähler** > 0 und **Wartemenge** nicht leer)

Methoden:

`Psyn`: Runterzählen um 1, synchron, wartet bei Zählerstand 0

- **Zähler** > 0 ?

ja: **Zähler** vermindern um 1

nein: Aufrufer blockieren und in **Wartemenge** stecken

`Vasyn`: Hochzählen um 1, asynchron

- **Wartemenge** leer?

ja: **Zähler** erhöhen um 1

nein: Prozess aus **Wartemenge** nehmen und deblockieren

Die Implementierung soll um die unten aufgeführten Methoden ergänzt werden. Welche davon lassen sich einfach, d.h. ohne Änderungen an den Attributen oder der Dateninvariante, implementieren? Beschreiben Sie für diese jeweils den Ablauf. Begründen Sie bei den anderen, warum keine einfache Implementierung möglich ist.

a) `Ptry`: Versuchendes Runterzählen um 1. Erfolg falls Zähler > 0.

- **Zähler** > 0 ? +1

ja: **Zähler** vermindern um 1; Erfolg melden +1.5

nein: Misserfolg melden +0.5

b) `Vtry`: Versuchendes Hochzählen um 1. Erfolg falls ein Prozess wartet.

Die Beschreibung ist etwas widersinnig, weil diese Methode niemals hochzählt. Sie hat nur Erfolg, wenn ein Prozess wartet, der das Hochzählen ausgleicht.

- **Wartemenge** leer? +1

ja: Misserfolg melden +0.5

nein: Prozess aus **Wartemenge** nehmen und deblockieren; Erfolg melden +1.5

c) `PNsyn`: Runterzählen um mehr als 1, synchron. Argument N, positive Zahl.

Das geht nicht ohne tiefgreifende Änderungen. Die Dateninvariante gilt nicht mehr, weil trotz Zähler > 0 ein Prozess in der Wartemenge liegen kann, der um mehr als den Zählerstand runterzählt. In der Wartemenge können sich mehrere Prozesse befinden, die unterschiedlich weit runterzählen. Das erfordert Änderungen an allen Methoden, die Prozesse aus der Wartemenge deblockieren.

Aufgabe K15 beschäftigt sich mit dieser Problemstellung.

d) `VNasyn`: Hochzählen um mehr als 1, asynchron. Argument N, positive Zahl.

- Delta := N
- solange Delta > 0 und **Wartemenge** nicht leer: +1.5
  - Prozess aus **Wartemenge** nehmen und deblockieren +1
  - Delta vermindern um 1 +0.5
- **Zähler** um Delta erhöhen +1

e) `PNtry`: Versuchendes Runterzählen um mehr als 1. Erfolg falls Zähler groß genug.

Im Gegensatz zu `PNsyn` ist diese Methode einfach zu implementieren, weil der Aufruf nicht gespeichert und später zu Ende gebracht werden muss.

- **Zähler** $\geq N$? +1

ja: **Zähler** vermindern um N; Erfolg melden +1.5

nein: Misserfolg melden +0.5

f) `VNtry`: Versuchendes Hochzählen um mehr als 1. Erfolg falls mindestens so viele Prozesse warten, dass sie um N runterzählen.

Da alle Prozesse in der Wartemenge um 1 runterzählen, muss die Methode die Anzahl der Elemente in der Wartemenge abfragen können. Falls das möglich ist:

- **Wartemenge** enthält weniger als N Prozesse? +1

ja: Misserfolg melden +0.5

nein: N Prozesse aus **Wartemenge** nehmen und deblockieren; Erfolg melden +1.5

Die Semantik dieser Methode ergibt meines Erachtens keinen Sinn. Ich habe sie nur wegen der Symmetrie mit in die Aufgabe genommen.

g) `P0asyn`: Runterzählen bis 0.

- **Zähler** auf 0 setzen +1

h) `V0asyn`: Hochzählen bis Wartemenge leer.

Die Beschreibung ist etwas widersinnig, weil diese Methode niemals hochzählt.

- alle Prozesse aus **Wartemenge** nehmen und deblockieren +1