nebenläufige programmierung

STEFAN ERCEG, MaRTIN KRITZL, SEBASTIAN STEINKELLNER

Roboterfabrik – A01

2014

Inhalt

[1. Aufgabenstellung 1](#_Toc399935146)

[2. Requirementanalyse 2](#_Toc399935147)

[3. Designüberlegung mittels UML-Klassendiagramm 4](#_Toc399935148)

[4. detaillierte Arbeitsaufteilung mit Aufwandabschätzung 4](#_Toc399935149)

[4.1. Aufwandabschätzung 4](#_Toc399935150)

[4.2. Arbeitsaufteilung für die Code-Erstellung 4](#_Toc399935151)

# 1. Aufgabenstellung

Es soll eine Spielzeugroboter-Fabrik simuliert werden. Die einzelnen Bestandteile des Spielzeugroboters (kurz Threadee) werden in einem Lager gesammelt. Dieses Lager wird als Verzeichnis und die einzelnen Elementtypen werden als Files im Betriebssystem abgebildet. Der Lagermitarbeiter verwaltet regelmäßig den Ein- und Ausgang des Lagers um Anfragen von Montagemitarbeiter und Kunden zu beantworten. Die Anlieferung der Teile erfolgt durch Ändern von Files im Verzeichnis, eine Lagerung fertiger Roboter ebenso.  
  
Ein Spielzeugroboter besteht aus zwei Augen, einem Rumpf, einem Kettenantrieb und zwei Armen.    
Die Lieferanten schreiben ihre Teile ins Lager-File mit zufällig (PRNG?) erstellten Zahlenfeldern. Die Art der gelieferten Teile soll nach einer bestimmten Zeit gewechselt werden.  
  
Die Montagemitarbeiter müssen nun für einen "Threadee" alle entsprechenden Teile anfordern und diese zusammenbauen. Der Vorgang des Zusammenbauens wird durch das Sortieren der einzelnen Ganzzahlenfelder simuliert. Der fertige "Threadee" wird nun mit der Mitarbeiter-ID des Monteurs versehen.  
  
Es ist zu bedenken, dass ein Roboter immer alle Teile benötigt um hergestellt werden zu können. Sollte ein Monteur nicht alle Teile bekommen, muss er die angeforderten Teile wieder zurückgeben um andere Monteure nicht zu blockieren. Fertige "Threadee"s werden zur Auslieferung in das Lager zurück gestellt.  
  
Alle Aktivitäten der Mitarbeiter muss in einem Logfile protokolliert werden. Verwenden Sie dazu Log4J [1].  
  
Die IDs der Mitarbeiter werden in der Fabrik durch das Sekretariat verwaltet. Es dürfen nur eindeutige IDs vergeben werden. Das Sekretariat vergibt auch die eindeutigen Kennungen für die erstellten "Threadee"s.  
  
Beachten Sie beim Einlesen die Möglichkeit der Fehler von Files. Diese Fehler müssen im Log protokolliert werden und entsprechend mit Exceptions abgefangen werden.

**Tipps und Tricks**

Verwenden Sie (optional) für die einzelnen Arbeiter das ExecutorService mit ThreadPools. Achten Sie, dass die Monteure nicht "verhungern". Angeforderte Ressourcen müssen auch sauber wieder freigegeben werden.  
  
**Beispiel für Teile-Files**  
-- "auge.csv"  
Auge,11,24,3,4,25,6,8,8,9,10,11,12,13,14,15,16,17,18,195,5  
Auge,91,62,3,4,54,6,7,8,9,10,11,12,13,14,15,16,17,18,119,32  
Auge,91,62,3,4,54,6,7,8,9,10,11,12,13,14,15,16,17,18,119,520  
  
-- "rumpf.csv"  
Rumpf,91,62,3,4,54,6,7,8,9,10,11,12,13,14,15,16,17,18,119,21  
  
**Beispiel für Threadee-File**  
-- "auslieferung.csv"  
Threadee-ID123,Mitarbeiter-ID231,Auge,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Auge,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Rumpf,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Kettenantrieb,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Arm,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Arm,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20  
Threadee-ID124,Mitarbeiter-ID231,Auge,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Auge,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Rumpf,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Kettenantrieb,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Arm,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,Arm,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20  
  
**Ausführung**  
  
Zu bedenken sind die im Beispiel angeführten Argumente. Diese können mit eigenem Code oder mit einer CLI-Library implementiert werden (z.B. [2]).  
Alle Argumente sind verpflichtend und die Anzahl muss positiv sein. Die obere Grenze soll sinnvoll festgelegt werden. Vergessen Sie auch nicht auf die Ausgabe der Synopsis bei einer fehlerhaften Eingabe! Sollten Sie zusätzliche Argumente benötigen sind diese erst nach einer Rücksprache implementierter.  
  
java tgm.sew.hit.roboterfabrik.Simulation --lager /verzeichnis/zum/lager --logs /verzeichnis/zum/loggen --lieferanten 12 --monteure 25 --laufzeit 10000

# 2. Requirementanalyse

Lager:

* darin werden die einzelnen Bestandteile des Roboters (= Threadee) gesammelt
* wird als Verzeichnis abgebildet

Lagermitarbeiter:

* verwaltet Ein- und Ausgang des Lagers

Spielzeugroboter:

* besteht aus zwei Augen, einem Rumpf, einem Kettenantrieb und zwei Armen
* die jeweiligen Bestandteile des Roboters werden als Files abgebildet

Lieferant:

* gibt dem Lagermitarbeiter Bestandteile die er liefert
* Art der gelieferten Teile soll nach einer bestimmten Zeit gewechselt werden

Montagemitarbeiter:

* fordert alle Bestandteile des Roboters
* falls nicht alle Teile geliefert werden, schickt der jeweilige Montagemitarbeiter sie wieder zurück, um andere Mitarbeiter nicht zu blockieren
* baut den Threadee zusammen
* stellt den fertigen Threadee wieder in das Lager zurück

Sekretariat:

* verwaltet die IDs der Mitarbeiter, damit nur eindeutige vorkommen

Loggen:

* alle Aktivitäten der Mitarbeiter werden protokolliert
* zu jeder Aktivität wird auch gespeichert, wer diese ausgeführt hat, dies wird durch die IDs realisiert

# 3. Designüberlegung mittels UML-Klassendiagramm

# 4. detaillierte Arbeitsaufteilung mit Aufwandabschätzung

## 4.1. Aufwandabschätzung

|  |  |
| --- | --- |
| **Teilaufgabe** | **Aufwandabschätzung (in Minuten)** |
| UML-Diagramm | 250 |
| Code-Erstellung inkl. Javadoc-Kommentaren | 500 |
| Debuggen | 200 |
| Testen | 250 |
| Protokoll | 200 |
| ***Gesamt*** | **1400** |

## 4.2. Arbeitsaufteilung für die Code-Erstellung

|  |  |  |  |
| --- | --- | --- | --- |
| **Klasse** | **Steinkellner** | **Kritzl** | **Erceg** |
| Simulation | x |  |  |
| Sekretariat | x |  |  |
| Mitarbeiter |  |  | x |
| Lagermitarbeiter |  |  | x |
| Montagemitarbeiter |  | x |  |
| Lieferant |  | x |  |
| Bauplan | x |  |  |
| Watchdog |  | x |  |

# 5. anschließende Endzeitaufteilung

**Erceg:**

|  |  |  |
| --- | --- | --- |
| **Arbeit** | **Datum** | **Zeit in Minuten** |
| Requirementanalyse | 11.09.2014 | 20 |
| UML | 11.09.2014 | 60 |
| UML | 16.09.2014 | 20 |
| UML | 17.09.2014 | 15 |
| UML | 18.09.2014 | 20 |
| Egit | 23.09.2014 | 30 |
| Log4J | 25.09.2014 | 90 |
| Mitarbeiter,Lagermitarbeiter | 27.09.2014 | 110 |
| RAF | 28.09.2014 | 20 |
| Testen | 29.09.2014 | 30 |
| Protokoll | 29.09.2014 | 50 |
| Änderungen am UML | 29.09.2014 | 20 |
| UML-Änderungen | 29.09.2014 | 60 |
| Debuggen | 30.09.2014 | 180 |
| Protokoll | 30.09.2014 | 120 |
| ***Gesamt*** | ***01.10.2014*** | ***845*** |

**Kritzl:**

|  |  |  |
| --- | --- | --- |
| **Arbeit** | **Datum** | **Zeit in Minuten** |
| Requirementanalyse | 11.09.2014 | 3 |
| UML | 11.09.2014 | 60 |
| UML | 16.09.2014 | 20 |
| UML | 17.09.2014 | 70 |
| UML | 18.09.2014 | 20 |
| UML | 21.09.2014 | 45 |
| Egit | 21.09.2014 | 70 |
| ExecutorService | 23.09.2014 | 20 |
| UML | 23.09.2014 | 15 |
| Arbeitseinteilung | 24.09.2014 | 30 |
| WatchDog, Lieferant, Montagemitarbeiter | 24.09.2014 | 150 |
| Lieferant, Montagemitarbeiter | 25.09.2014 | 120 |
| Kommentieren, Änderungsvorschläge | 25.09.2014 | 45 |
| log-statements, sortPart | 26.09.2014 | 60 |
| edit log-statements, improve Code | 28.09.2014 | 90 |
| change getAllParts(), tests | 28.09.2014 | 120 |
| change giveBack(ArrayList<String[], int) | 29.09.2014 | 40 |
| Debuggen | 29.09.2014 | 60 |
| Debuggen | 30.09.2014 | 120 |
| Testen, getAllParts | 30.09.2014 | 120 |
| ***Gesamt*** | ***01.10.2014*** | ***1278*** |

**Steinkellner:**