Übungen 6

Patrick Bucher

28.03.2017

Inhaltsverzeichnis

1 Wait-Pool-Demo

a)

Was passiert bei der Ausführung von `DemoWaitPool`?

Es kommt zu zwei Exceptions. Die eine wird vom `wait()`-Aufruf innerhalb der `run()`-Methode der Klasse `MyTask` geworfen, die andere vom `notify()`-Aufruf innerhalb der `main()`-Methode der Klasse `DemoWaitPool`.

Wie erklären Sie sich das Verhalten der Klassen?

Der Aufruf von `wait()` versucht `this` zu sperren: ein Objekt, das nicht gesperrt ist. Das hat eine `IllegalMonitorStateException` zur Folge. Da dies eine unchecked-Exception ist, wird die `run()`-Methode verlassen, ohne dass der Lock freigegeben wird.

Mit dem `notify()`-Aufruf auf das Lock-Objekt soll dann der Thread wieder geweckt werden. Doch `lock` wurde zuvor nicht gesperrt, was eine weitere `IllegalMonitorStateException`zur Folge hat.

Welche *minimalen* Korrekturen sind nötig?

In `MyTask` müsste `wait()` auf das Objekt `lock` ausgeführt werden:

```
synchronized (lock) {
    try {
        lock.wait();
    } catch (InterruptedException ex) {
        return;
    }
}
```

In `DemoWaitPool` müsste `lock` in einem `synchronized`-Block stehen:

```
synchronized (lock) {
    lock.notify();
}
```

Gibt es noch andere Korrektur-Varianten?

Statt auf das `lock`-Objekt könnte man direkt auf die `MyTask`-Instanz sperren. Es ginge hier auch ohne spezielles `lock`-Objekt.

b)

Eclipse bietet diese Funktionalität leider nicht, weswegen ich diese Aufgabe vorerst nicht mache.

c)

Was passiert bei der Ausführung von `DemoWaitPool`?

`MyTask` wartet ewig.

Wie erklären Sie sich das Verhalten?

`MyTask` kann keine Sperre auf `lock` erstellen, da `lock` bereits von `DemoWaitPool` gesperrt wurde, und wartet deswegen ewig.

Reflektion

Frage: Was ist bei der Benachrichtigung mit Hilfe der `notify`/`notifyAll`-Methoden zu beachten?

Antwort: Es muss vorher mindestens ein Thread wartend sein. Der `wait()`-Aufruf muss *vor* dem `notify()`/`notifyAll()`-Aufruf erfolgen.

Frage: Warum wird für die Benachrichtigung `notifyAll` statt `notify` empfohlen?

Antwort: `notify` weckt nur einen schlafenden Thread auf, `notifyAll` ermöglicht es allen schlafenden Threads aufzuwachen. Wird eine Aktion abgeschlossen, nach der es mehreren Threads wieder möglich wird weiterzuarbeiten, sollte `notifyAll` ausgeführt werden, damit soviele Threads wie möglich wieder arbeiten können.

Frage: Wenn ein Thread einen anderen Thread steuern will, ist dies offensichtlich keine gute Lösung. Wie sieht eine bessere Lösung aus?

Antwort: Eine übergeordnete Logik soll sich um die Steuerung der beiden Threads kümmern.

2 Pferderennen

3 Bounded Buffer

4 Signalgeber

5 Optional: Scatter/Gather-Verarbeitung