Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Programmieren 2

Prof. Dr. Larissa Putzar &
Thorben Ortmann

Nebenläufige Beobachter - Aufgaben

Vorwort

- Die Observer-Implementierungen aus der Vorlesung finden Sie hier: https://artemis.mt.haw-hamburg.de/playground_tor/sose2022programmieren2/exercises/2-concurrent-observers

Aufgabe 1

Die in der Demo gezeigten Klassen *ConcreteObservee* und *ConcreteObserver* sind die konkreten Implementierungen von *Observee bzw. Observer*. Allerdings haben diese keinen Bezug zur echten Welt.

1. Überlegen Sie sich ein Beispiel, in dem ein Observer-Pattern sinnvoll ist.
2. Erstellen Sie für ihr Bespiel konkrete Klassen, analog zu *ConcreteObservee* und *ConcreteObserver,* die von der abstrakten *Observee*-Klasse erben bzw. das *Observer*-Interface implementieren.
3. Simulieren und prüfen sie das Verhalten ihres Beispiels anhand von Testfällen.

Diese Aufgabe basiert auf der sequentiellen Implementierung des Observer-Pattern.

Aufgabe 2

Implementieren Sie die naive nebenläufige Ausführung von Updates.

Sie können dafür zunächst mit der in der Demo gezeigten Implementierung der sequentiellen Ausführung starten und dann die beschriebenen Anpassungen vornehmen.

Demonstrieren Sie das Problem der Lost Updates anhand eines Testfalls (der Testfall soll aufgrund eines verpassten Updates fehlschlagen).

Aufgabe 3

Implementieren Sie das Observer-Pattern mit Hilfe von kritischen Abschnitten unter Nutzung des *synchronized* Keywords.

Sie können Ihre Implementierung auf Basis des Lazy Waiting-Ansatzes beginnen.

Versuchen Sie, Ihre *synchronized*-Blöcke nur so groß zu machen, wie es notwendig ist.

Nutzen Sie ihren Testfall zur Demonstration von Lost Updates aus Aufgabe 2 und zeigen Sie, dass diese nun nicht mehr auftreten.

Aufgabe 4

- Schreiben Sie eine Klasse *TimeThread*, die jede zweite Sekunde die aktuelle Zeit ausliest und in der Konsole ausgibt.

- Starten Sie eine Instanz Ihrer Thread-Klasse.

- Ermöglichen Sie das Stoppen Ihrer *TimeThread*-Instanz durch die Eingabe von „q“ (und einer NewLine danach bzw. Drücken der Enter-Taste) in die Konsole. Dazu können Sie einen *Scanner* benutzen:

```
Scanner scanner = new Scanner(System.in);
String consoleInput = scanner.next();
```

Aufgabe 5

- Schreiben Sie eine Klasse *CounterThread*, die alle 10 ms eine zufällige Zahl zwischen 0 und 100 in eine *ArrayList* schreibt.
- Erstellen und starten Sie zwei Instanzen von *CounterThread*. Beide Threads sollen in dieselbe *ArrayList* schreiben.
- Stoppen Sie die Threads nach einer Sekunde und geben Sie die Anzahl der Elemente in der *ArrayList* aus.

- Machen Sie den Zugriff auf die *ArrayList* nun threadsafe.
- Stoppen Sie die Threads wieder nach einer Sekunde und geben Sie die Anzahl der Elemente in der *ArrayList* aus.

- Was fällt Ihnen auf und wie können Sie sich Ihre Beobachtung erklären?