**Programmieren in Java**

http://proglang.informatik.uni-freiburg.de/teaching/java/2015/

# Java-Übung Blatt 4 (Zustandsbehaftete Klassen)

2015-05-18

## Hinweise

- Ändern Sie nicht die Schreibweise von Bezeichnen, die auf dem Übungsblatt vorgegeben sind. Dies betrifft sämtliche global sichtbaren Namen von Eclipse-Projekten, Paketen, Klassen, etc.
- Bezeichner und Kommentare bitte auf *Englisch*!
- Schreiben Sie *sinnvolle* Kommentare.
- Laden Sie Ihre Lösungen mit Subversion (SVN) ins Übungssystem hoch. Den entsprechenden Pfad finden Sie online.
- Sollte das Übungssystem Ihre Einreichung nicht übersetzen können, dann wird sie nicht korrigiert und Sie erhalten keine Punkte.
- Die Korrektur Ihrer Abgabe wird unter dem Namen `Feedback-<login>-ex<NN>_<X>.txt` in das jeweilige Projektverzeichnis eingecheckt. Dabei ist `<login>` Ihr *myAccount*-Name und `<NN>_<X>` der Projektname.
- Sie können Ihre Gesamtpunktzahl im Übungsportal einsehen.
- Dokumentieren Sie Ihren Zeitaufwand, den gefühlten Schwierigkeitsgrad, sowie Probleme beim Lösen der Aufgabe oder auch Anregungen und Vorschläge zur Verbesserung in der Datei `<Projektverzeichnis>/erfahrungen.txt`

**Hinweis :** Dieses PDF stammt aus dem Archiv `ex04.zip`. Dieses Archiv enthält weiterhin noch Skelett-Projekte, die für die Bearbeitung der Aufgaben hilfreich sind.

### Exercise 1 (Zugbrücke Teil I, 5 Punkte)

Projektverzeichnis: `ex04_1`. Package: `drawbridge`.

Mittelalterlichen Zugbrücken sollen durch ein Java-Programm simuliert werden. Eine Zugbrücke kann hochgezogen (rise) und heruntergelassen (lower) werden. Um eine Brücke zu bewegen muss Arbeit investiert werden. Die Menge an Arbeit für hoch- und herunterziehen kann sich von Brücke zu Brücke unterscheiden, beträgt jedoch immer mindestens eine Einheit. Um die Arbeit zu verrichten können jeder Brücke bis zu vier Arbeiter (worker) zugeordnet werden, die jeweils eine Arbeitseinheit pro Zeiteinheit leisten können. Einer Brücke können auch Arbeiter abgezogen werden, allerdings muss immer mindestens ein Arbeiter zugewiesen sein, während sich die Brücke bewegt. Es ist auch nicht möglich den Bewegungsprozess einer Brücke zu unterbrechen.

Im Skelett der Aufgabe finden Sie eine Klasse BridgeSimulation deren main-Methode die gewünschte Funktionsweise weiter demonstriert (siehe unten für die Ausgabe).

1. Erstellen Sie ein Klassendiagramm für Zugbrücken.
2. Implementieren Sie ihr Design. Brücken sollen auch das im Skelett spezifizierte Interface SimulationObject implementieren. Die Methode aus dem Skelett soll zusammen mit Ihrer Implementierung lauffähig sein. Achten Sie darauf, dass die Vorgaben der Spezifikation eingehalten werden. Signalisieren Sie ungültige Zustände und Operationen wo nötig durch entsprechende Exceptions mit sinnvollen Fehlermeldungen.
   (Siehe `java.lang.IllegalArgumentExceptions` bzw. `java.lang.IllegalStateException.`
3. Testen Sie die Methoden der Zugbrücken-Klasse mittels JUnit. Erstellen Sie die Testfälle derart, dass alle Zustände erreicht und wieder verlassen werden. Testen Sie auch das erwartete Auftreten von Fehlern. (Siehe `http://stackoverflow.com/questions/156503/how-do-you-assert-that-a-certain-exception-is-thrown-in-junit-4-tests` für Hinweise wie in JUnit auf Exceptions geprüft werden kann.)

**Ausgabe von BridgeSimulation:** (Ihre Ausgabe kann abweichen, sollte aber vergleichbare Informationen liefern.)

```
---Initial states
Bridge Light Bridge (2 workers): DOWN
Bridge Heavy Bridge (3 workers): DOWN
---Timestep 0---
Bridge Light Bridge (2 workers): DOWN
Bridge Heavy Bridge (3 workers): DOWN
---Timestep 1---
Bridge Light Bridge (2 workers): RISING
Bridge Heavy Bridge (3 workers): DOWN
---Timestep 2---
Bridge Light Bridge (2 workers): UP
```

```
Bridge Heavy Bridge (3 workers): DOWN
---Timestep 3---
Bridge Light Bridge (1 workers): UP
Bridge Heavy Bridge (3 workers): RISING
---Timestep 4---
Bridge Light Bridge (1 workers): UP
Bridge Heavy Bridge (3 workers): RISING
---Timestep 5---
Bridge Light Bridge (1 workers): LOWERING
Bridge Heavy Bridge (3 workers): RISING
---Timestep 6---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): UP
---Timestep 7---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): UP
---Timestep 8---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): UP
---Timestep 9---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): UP
```

**Exercise 2** (Zugbrücke Teil 2, 5 Punkte)
Projektverzeichnis: `ex04_2`. Package: `drawbridge`.
Im zweiten Teil der Aufgabe soll die Simulation aus Teil I folgendermaßen erweitert werden: Die Brücken wollen von Reisenden (Travellers) überquert werden. Ein Reisender hat einen Reiseplan, der die Brücken auflistet, die er nacheinander überqueren muss. Brücken können nur dann überquert werden, wenn sie unten sind; Ein Reisender muss also vor gehobenen oder sich in Bewegung befindenen Brücken warten. Ein Reisender hat außerdem nur eine begrenzte Zeit Geduld für Wartezeiten auf einer Reise. Überschreitet die gesamte Wartezeit einer Reise an Brücken seine Geduld, bricht er die Reise ab.[1] Kann er alle Brücken überqueren, ist seine Reise beendet. Reisende können ihre Reise zum Ausruhen (`rest`) unterbrechen und wieder fortsetzen (`continueTrip`). Nehmen Sie der Einfachheit halber an, dass die Reisezeit zwischen Brücken vernachlässigbar ist.

1. Erweitern Sie das Klassendiagramm aus Teil I für Reisende.
2. Implementieren Sie ihr Design. Das Programm TravellerSimulation aus dem Skelett soll zusammen mit Ihrer Implementierung lauffähig sein. Wie zuvor: Achten Sie darauf, dass die Vorgaben der Spezifikation eingehalten werden. Signalisieren Sie ungültige Zustände und Operationen wo nötig durch entsprechende Exceptions mit sinnvollen Fehlermeldungen. (siehe java.lang.IllegalArgumentExceptions bzw. java.lang.IllegalStateException).
3. Wie zuvor: Testen Sie die Methoden der Traveller-Klasse mittels JUnit. Erstellen Sie die Testfälle derart, dass alle Zustände erreicht und wieder verlassen werden. Testen Sie auch das erwartete Auftreten von Fehlern.

**Ausgabe von TravellerSimulation** (Ihre Ausgabe kann abweichen, sollte aber vergleichbare Informationen liefern.)

```
Initial states:
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): DOWN
Bridge Some Bridge (3 workers): DOWN
Traveller Traveller1 (patience 4): RESTING (plan: [Light Bridge, Heavy Bridge])
Traveller Traveller2 (patience 15): RESTING (plan: [Light Bridge, Heavy Bridge, Some Bridge])
Traveller Traveller3 (patience 5): RESTING (plan: [Heavy Bridge, Light Bridge])
---Timestep 0---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): RISING
Bridge Some Bridge (3 workers): DOWN
Traveller Traveller1 (patience 3): WAITING_TO_CROSS (plan: [Heavy Bridge])
```

[1] Stellen Sie sich vor, dass Reisende, denen die Geduld ausgeht sich schmollend auf den Boden setzten und gar nichts mehr tun.

```
Traveller Traveller2 (patience 14): WAITING_TO_CROSS (plan: [Heavy Bridge, Some Bridge])
Traveller Traveller3 (patience 4): WAITING_TO_CROSS (plan: [Heavy Bridge, Light Bridge])
---Timestep 1---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): UP
Bridge Some Bridge (3 workers): DOWN
Traveller Traveller1 (patience 2): WAITING_TO_CROSS (plan: [Heavy Bridge])
Traveller Traveller2 (patience 13): WAITING_TO_CROSS (plan: [Heavy Bridge, Some Bridge])
Traveller Traveller3 (patience 4): RESTING (plan: [Heavy Bridge, Light Bridge])
---Timestep 2---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): UP
Bridge Some Bridge (3 workers): DOWN
Traveller Traveller1 (patience 1): WAITING_TO_CROSS (plan: [Heavy Bridge])
Traveller Traveller2 (patience 12): WAITING_TO_CROSS (plan: [Heavy Bridge, Some Bridge])
Traveller Traveller3 (patience 4): RESTING (plan: [Heavy Bridge, Light Bridge])
---Timestep 3---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): LOWERING
Bridge Some Bridge (3 workers): RISING
Traveller Traveller1 (patience 0): ABORTED (plan: [Heavy Bridge])
Traveller Traveller2 (patience 11): WAITING_TO_CROSS (plan: [Heavy Bridge, Some Bridge])
Traveller Traveller3 (patience 4): RESTING (plan: [Heavy Bridge, Light Bridge])
---Timestep 4---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): DOWN
Bridge Some Bridge (3 workers): UP
Traveller Traveller1 (patience 0): ABORTED (plan: [Heavy Bridge])
Traveller Traveller2 (patience 10): WAITING_TO_CROSS (plan: [Some Bridge])
Traveller Traveller3 (patience 4): RESTING (plan: [Heavy Bridge, Light Bridge])
---Timestep 5---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): DOWN
Bridge Some Bridge (3 workers): LOWERING
Traveller Traveller1 (patience 0): ABORTED (plan: [Heavy Bridge])
Traveller Traveller2 (patience 9): WAITING_TO_CROSS (plan: [Some Bridge])
Traveller Traveller3 (patience 4): RESTING (plan: [Heavy Bridge, Light Bridge])
---Timestep 6---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): DOWN
Bridge Some Bridge (3 workers): DOWN
Traveller Traveller1 (patience 0): ABORTED (plan: [Heavy Bridge])
Traveller Traveller2 (patience 8): DONE
Traveller Traveller3 (patience 3): WAITING_TO_CROSS (plan: [Light Bridge])
---Timestep 7---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): DOWN
Bridge Some Bridge (3 workers): DOWN
Traveller Traveller1 (patience 0): ABORTED (plan: [Heavy Bridge])
Traveller Traveller2 (patience 8): DONE
Traveller Traveller3 (patience 2): DONE
---Timestep 8---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): DOWN
Bridge Some Bridge (3 workers): DOWN
Traveller Traveller1 (patience 0): ABORTED (plan: [Heavy Bridge])
Traveller Traveller2 (patience 8): DONE
Traveller Traveller3 (patience 2): DONE
---Timestep 9---
Bridge Light Bridge (1 workers): DOWN
Bridge Heavy Bridge (3 workers): DOWN
Bridge Some Bridge (3 workers): DOWN
Traveller Traveller1 (patience 0): ABORTED (plan: [Heavy Bridge])
Traveller Traveller2 (patience 8): DONE
```

```
Traveller Traveller3 (patience 2): DONE
```