Praktikum Programmiermethodik 2 (Technische Informatik)
WS 2016/2017, Hochschule für Angewandte Wissenschaften (HAW), Hamburg
Prof. Dr. Philipp Jenke, Kasperczyk-Borgmann

Für dieses Aufgabenblatt gelten die folgenden Regeln:

- Der Java Code Style (siehe EMIL) ist einzuhalten. Es werden keine Abgaben abgenommen, die diese Anforderungen nicht erfüllen.
- Pro Team muss ein gemeinsames Git-Repository für das Projekt vorhanden sein (z.B. auf GitHub, Bitbucket oder dem HAW-Informatik-Home-Verzeichnis).
- Der Code muss vollständig getestet sein.

Aufgabenblatt 3: Synchronisation von Threads, Visualisierung mit JavaFX, Entwurfsmuster

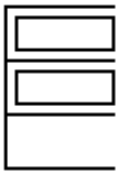

Abbildung 1: Eine mögliche Visualisierung für einen Rangierbahnhof. Die Gleise 0 und 1 sind durch je einen Zug belegt, Gleis 2 ist leer. Gültige Operationen sind beispielsweise das Einfahren eines Zuges in Gleis 2 oder das Ausfahren des Zuges in Gleis 0. Die Operation Einfahren eines Zuges in Gleis 1 beispielsweise müsste "geparkt" werden.

Aufgabe 3.1: Rangierbahnhof

Lernziele: Synchronisation und das Monitor-Konzept anwenden.
Aufgabe: Implementieren Sie eine Klasse, die einen Rangierbahnhof repräsentiert. Ein Rangierbahnhof hat eine feste Anzahl von Gleisen. Auf jedem Gleis kann genau ein Zug stehen. (Umsetzung: der Bahnhof hat ein Array vom Typ Zug). Legen Sie für Zug einen eigenen Typ an, der aber keine weiteren Eigenschaften hat. Der Bahnhof muss unter anderem je eine Methode zum Einfahren eines Zuges auf ein bestimmtes Gleis und zum Ausfahren eines Zuges aus einem bestimmen Gleis bieten. Die Gleise werden durch Ihre Nummer angegeben (0 ... n-1). Zum Rangierbahnhof gibt es nur ein Zufahrtsgleis. Daher kann immer nur genau ein Thread auf diese Methoden gleichzeitig zugreifen. Offensichtlich ist, dass aus einem Gleis immer nur genau dann ein Zug ausfahren kann, wenn einer da ist und in ein Gleis immer nur genau dann ein Zug einfahren kann, wenn das Gleis leer ist. Setzen Sie dies durch den Monitor-Mechanismus in Java um.

Aufgabe 3.2: Lokführer

Lernziele: Mit einem Thread auf ein Monitor-überwachtes Objekt zugreifen.
Aufgabe: Lokführer sind als Thread implementiert. Ein Lokführer hat immer genau eine von zwei möglichen Aufgaben: Entweder einen neuen Zug auf ein bestimmtes Gleis einfahren oder einen Zug (der dort steht) aus einem bestimmten Gleis ausfahren. Hat ein Lokführer eine der beiden Aktionen erfolgreich durchgeführt, dann muss dieses Ereignis auf der Konsole ausgegeben werden.

Aufgabe 3.3: Simulation

Lernziele: Threads starten, Monitor-Konzept verwenden.
Aufgabe: Schreiben Sie eine Simulation für das Ein- und Ausfahren von Zügen aus einem Rangierbahnhof. Erstellen Sie zunächst einen Rangierbahnhof. Erzeugen Sie dazu kontinuierlich (z.B. alle 500 ms) neue Lokführer mit je einer Aufgabe. Der Lokführer fährt dann je nach Aufgabe entweder einen neuen Zug in den Bahnhof ein oder aus dem Bahnhof aus. Ist die Aktion aktuell nicht möglich, so muss die Aktion durch den Monitor "zwischengeparkt" werden (nicht durch den Lokführer!). Setzen Sie die Simulation als ein Runnable um (dies wird in der kommenden Aufgabe benötigt). Die eigentliche Simulation darf eine Endlosschleife sein, die durch Beenden des Programms abgebrochen wird.

Aufgabe 3.4: Visualisierung + Beobachter-Muster

Lernziele: JavaFX - Fenster, Zeichnen, Thread-Sicherheit, Entwurfsmuster anwenden
Aufgabe: Schreiben Sie eine Visualisierung für die Simulation. Die Form der Darstellung ist freigestellt (als Inspiration kann Abbildung 1 dienen). Es müssen die Gleise erkennbar sein und es muss erkennbar sein, ob

sich auf dem Gleis aktuell ein Zug befindet oder nicht. Die Darstellung muss immer dann aktualisiert werden, wenn die Simulation einen weiteren Schritt gemacht hat. Starten Sie dazu die Simulation als eigenen Thread (sie ist ja schon als Runnable vorbereitet) und verwenden Sie für die Kommunikation das Beobachter-Muster. Die Simulation ist dann das beobachtete Objekt während die Visualisierung der Beobachter ist. Wird im Beobachter die update()-Methode aufgerufen, muss der Bahnhof neu gezeichnet werden. Achten Sie darauf, dass das Zeichnen durch den JavaFX-Thread durchgeführt werden muss (siehe Vorlesungsfolien).

[Optionale Erweiterung, freiwillig:] Visualisieren Sie die Warteschlangen für jedes Gleis (also die Züge, die gerne einfahren würden, das aber nicht können, weil das Gleis belegt ist).