

Übungen zu Methoden des Software-Engineering
(Florian Hacklinger, Dr. N. Koch, Prof. Dr. M. Wirsing)

Aufgabe 1

Petrinetze

Ein Bahnhof hat zwei Bahnsteige, zwei eingehende Gleise und zwei ausgehende Gleise. Von jedem eingehenden Gleis kann jeder Bahnsteig angefahren werden; von jedem Bahnsteig kann jedes ausgehende Gleis angesteuert werden. Zur Vereinfachung wird der Bahnhof nur in einer Richtung benutzt: Züge kommen von einem eingehenden Gleis und verlassen den Bahnhof auf einem ausgehenden Gleis.

Es wartet eine beliebige, nicht festgelegte Anzahl von Zügen darauf, eines der beiden eingehenden Gleise zu befahren. Von einem eingehenden Gleis befährt ein Zug anschließend eines der beiden Gleise im Bahnhof und verläßt den Bahnhof wahlfrei auf einem der beiden ausgehenden Gleise.

- a) Modellieren Sie den Bahnhof mit Hilfe eines Petrinetzes und dem Tool PNK. Achten Sie darauf, dass jeder Gleisabschnitt von maximal einem Zug besetzt sein darf und dass jederzeit neue Züge auf die Einfahrt auf eines der beiden eingehenden Gleise warten.
- b) Analysieren Sie Ihr Modell mit Hilfe von INA. Stellen Sie dabei folgende Eigenschaften sicher: Das Netz ist lebendig und es existieren keine toten Transitionen zu Anfang. Stellen Sie mit einem Erreichbarkeitsgraph sicher, dass zu jedem Zeitpunkt ein Gleisabschnitt von maximal einem Zug besetzt ist.
- c) Implementieren Sie den Bahnhof in Java als Applikation für den Petrinetz-Kernel. Dabei soll als Modell der Applikation das Petrinetz aus Teilaufgabe a) verwendet und der PNK zur Ablaufsteuerung benutzt werden. Ferner soll die Anwendung eine graphische Oberfläche haben, die eingehenden, die ausgehenden und die Gleise des Bahnhofs darstellt und anzeigt auf welchen Gleisabschnitten ein Zug ist. Die Anwendung soll einen Knopf haben, der das Schalten einer Transition auslöst. Nach jedem Schaltvorgang soll die Oberfläche das geänderte Modell darstellen.

Für diese Aufgabe sollen

- die pnml-Datei die das Modell enthält,
- der vollständige Java-Quelltext, die Bytecode-Dateien, und verwendete Bibliotheken,
- eine Anleitung zum Starten des Programms

abgegeben werden.

Aufgabe 2

Äquivalenzklassentest

Die Funktion `countMinutes(int hours, int minutes)` berechnet die Anzahl der verstrichenen Minuten seit Tagesbeginn (0 Uhr), wobei die Parameter `hours` und `minutes` als Uhrzeit interpretiert werden. Wird für mindestens einen der Parameter ein negativer Wert übergeben, so soll eine Ausnahme (`IllegalArgumentException`) „keine gültige Uhrzeit: negative Zahl“ auftreten, wird für die Stunden mehr als 23 und/oder für die Minuten mehr als 59 übergeben, so soll eine Ausnahme (`IllegalArgumentException`) „keine gültige Uhrzeit: Wert zu groß“ geworfen werden.

Bestimmen Sie für die Funktion `countMinutes` Äquivalenzklassen für die Parameter `hours` und `minutes`. Geben Sie für jede mögliche Kombination der Äquivalenzklassen ein Testdatum an.

Implementieren Sie die Funktion in Java und Testen Sie sie mit JUnit. Verwenden Sie dazu die gefundenen Testdaten.

Geben Sie

- eine Beschreibung der gefundenen Äquivalenzklassen und die entsprechenden Testdaten,
- den Quelltext des Programms und
- die JUnit-Tests

ab.

Hinweise zur Abgabe: Das Übungsblatt ist bis zum **24.01.2007, 14.15 Uhr** im Abgabesystem UniWorX abzugeben. Abgaben nach dem genannten Termin können in **keinem** Fall bewertet werden.