Dr. Johannes Mayer
Ralph Guderlei
Institut für Angewandte Informationsverarbeitung

10. Mai 2007
Blatt 3

Softwaretest (SS 2007)

Abgabetermin: **23**. Mai 2007

Einstieg in JUnit

Wurzeln

Unter den Beispielen zu den Übungen finden Sie ein jar-File `blatt3.jar`. Darin sind zwei Klassen (`blatt3.IntegerSqrt1` und `blatt3.IntegerSqrt2` enthalten, die eine statische Methode zum Berechnen der ganzahligen Wurzel einer nicht-negativen Integer-Zahl n bereitstellen. Die ganzzahlige Wurzel ist die größte Integer-Zahl r, für die gilt: $r^2 \leq n$. Die Methode hat den Namen `isqrt` und erwartet einen Integer als Parameter, der Rückgabewert ist dann die ganzzahlige Wurzel des Parameter-Werts. Falls eine negative Zahl übergeben wird, wird eine `IllegalArgumentException` geworfen.

Beide Klassen enthalten Fehler, versuchen Sie, alle Fehler in den Klassen zu finden. Implementieren Sie die Tests mit JUnit.

Blatt 2 reloaded

In Blatt 2 haben Sie Testfälle basierend auf der Spezifikation eines Programms zur Berechnung von Objekt-Bewegungen entwickelt. Die Klasse `blatt3.TurtleGraphics` verwendet nun diese Spezifikation. Sie ist angelehnt an LOGO, eine funktionale Programmiersprache, die hauptsächlich für eine einfache Methoden für Computer-Grafik bekannt wurde.

Die Idee ist, dass eine Schildkröte mit einem Stift auf dem Rücken über den Bildschirm läuft und durch einfache Befehle zu steuern ist. Folgende Befehle sind möglich:

`left()` Dreht die Schildkröte um 90 Grad nach links ohne sie zu bewegen

`right()` Dreht die Schildkröte um 90 Grad nach rechts ohne sie zu bewegen

`move()` bewegt die Schildkröte einen Schritt nach vorne

Zusätzliich existieren noch die Methoden `move(double steps)` und `setAngle(double newAngle)` um die Schildkröte zu bewegen bzw. die Laufrichtung der Schildkröte direkt zu setzen.

Die aktuellen Werte für die Postition und die Richtung des Tiers lassen sich über die Methoden `double getX()`, `double getY()` und `double getAngle()` abfragen.

Die ebenfalls in `blatt3.jar` zur Verfügung gestellte Klasse enthält Fehler, versuchen Sie, alle Fehler in der Klasse zu finden.

Entkopplung

Warenkorb

Gegeben sei folgende einfache Warenkorb-Komponente. Die Klasse `Cart` dient als Container für diverse Waren. Jede Klasse, die eine solche Ware darstellt, muss das Interface `Item` implementieren. Die Klasse `Cart` besitzt zwei Methoden, eine zum Hinzufügen von Waren zum Einkaufswagen in einer bestimmten Menge (`addItem(Item anItem, int aQuantity)`) und eine zum Berechnen des Gesamtpreises des Einkaufs (`double getTotalPrice()`). Dieser berechnet sich aus dem Einzelpreis der Ware und deren Anzahl im Warenkorb. Jede Ware hat (definiert im Interface) eine Methode, um ihren Einzelpreis zu berechnen. Die Anzahl der (unterschiedlichen) Waren im Warenkorb kann über die Methode `int getNumberOfItems()` abgefragt werden.

Bevor jedoch eine Ware in den Warenkorb gelegt werden kann, fragt der Warenkorb erst im Lagersystem an, ob die gewünschte Ware in der gewünschten Menge verfügbar ist. Dazu greift der Warenkorb auf ein Objekt vom Typ `Store` zurück. Eine Referenz auf dieses Objekt kann mit der Methode `setStore()` übergeben werden.

Zum Testen steht ihnen natürlich nicht das "echte" Lagersystem zur Verfügung, implementieren Sie deshalb zuerst ein passendes Dummy-Objekt.

Um den Einkaufswagen unabhänging von evtl. fehlerbehafteten Waren-Implementierungen testen zu können, implementieren Sie Waren-Dummys und benutzen dann diese zum Testen des Einkaufswagens.

Die notwendingen Interfaces finden Sie in den Beispielen, die class-files befinden sich ebenfalls in `blatt3.jar` .

Test-Suiten

Fassen Sie alle JUnit-Tests des Übungsblatts zu einer Test-Suite zusammen. **Viel Erfolg!**