
Programmieren in Java
<http://proglang.informatik.uni-freiburg.de/teaching/java/2013/>

Java-Übung Blatt 3 (Einfache Klassen)

2013-04-30

Hinweise

- Schreiben Sie Identifier *genau so*, wie sie auf dem Blatt stehen (inklusive Groß- und Kleinschreibung), nicht nur ungefähr.
- Identifier 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.
- Das Übungssystem kann überprüfen, ob Sie Ihr Quelltext den Anforderungen genügt und ob Sie alle Klassen erstellt haben, etc. Nutzen Sie dies!
- Sollte das Übungssystem Ihre Lösungen ablehnen, dann werden sie *nicht* korrigiert! Akzeptanzkriterien:
 - Compiliert erfolgreich
 - Checkstyle bringt keine Fehler
 - Alle Packages, Klassen, Interfaces, Methoden, Typen, Argumente sind exakt wie auf dem Übungsblatt gefordert.
- Ihr korrigierender Tutor wird die Korrektur Ihrer Abgabe in Ihr svn-Repository unter dem Namen `Feedback-<login>-ex<XX>.txt` comitten. (<login> ist dabei Ihr myAccount name und <XX> die Kennziffern des Übungsblatts)
- Zusätzlich können Sie Ihre Gesamtpunktzahl im Übungsportal einsehen.

Abgabe: Freitag, 10.05.2013, um 23.59 Uhr.

Testen heißt in diesem und in folgenden Aufgabenblättern Testen mittels JUnit-Tests.

Aufgabe 1 (Papier, 14 Punkte)Projekt: `ex03_1`. Package: `paper`.

Wir betrachten rechteckige Papierformate mit einer Länge und einer Breite, jeweils als `double` in Millimetern. Zusätzlich haben Papierformate einen Namen (z.B. für den Seite-Einrichten-Dialog), der mit der Methode `getName` abgefragt werden kann. Man kann ein Papierformat nach Länge und Breite in Millimetern fragen, indem man die Methoden `getLength` und `getWidth` aufruft. Man kann ein Papierformat auch fragen, was herauskommt, wenn man die längere Seite halbiert (Methode `bisect`) – die Antwort ist wieder ein Papierformat, und zwar mit Länge \geq Breite, also hochkant.

- (a) Definieren Sie nach dieser Beschreibung das Interface `PaperSize` für Papierformate.
- (b) Implementieren Sie eine Klasse für allgemeine Papiere (`GeneralPaperSize`) die das Interface (`PaperSize`) implementiert. Als Name soll $\ell \times w$ zurückgegeben werden, wobei ℓ und w die Länge bzw Breite in Millimetern sind, etwa “305x210” für ein Format mit Länge 305mm und Breite 210mm.
Testen Sie Namensberechnung und Halbieren (`GeneralPaperSizeTest`). Beim Halbieren gibt es wegen der Hochkant-Regel viele interessante Fälle. Passen Sie beim `assert` auf wegen Rundungsfehlern!
- (c) Die DIN-A-Reihe der Papiere sei so definiert:
 - Die Länge ist das $\sqrt{2}$ -Fache der Breite.
 - A0 hat die Fläche $1m^2$.
 - $A(i+1)$ ist ein halbiertes A_i .

Implementieren Sie `PaperSize` für A-Papiere als Klasse `ASeriesPaperSize`. Die Klasse darf als einziges Feld die Nummer innerhalb der A-Reihe enthalten. Als Name soll “ A_n ” herausgegeben werden, wobei n die Nummer innerhalb der A-Reihe ist.

Testen Sie Größen- und Namensberechnung und Halbieren.

Aufgabe 2 (Speisekarte, 14 Punkte)

Projekt: `ex03_2`. Package: `menu`.

Sie möchten eine Speisekarte modellieren. Extrahieren Sie aus folgendem Text Klassen und Interfaces. (Die geklammerten Namen sind hier lediglich Vorschläge... das Design ist in dieser Aufgabe Ihnen überlassen.)

Eine Speisekarte (`Menu`) ist eine Liste¹. Ein Eintrag in der Speisekarte hat einen Namen (`name`), eine Beschreibung (`description`) und einen Preis (`price`, in Cent). Denken Sie daran, dass man Preise ganzzahlig abspeichern sollte! Die Einträge können Getränke (`Drink`) oder Speisen (`Dish`) sein. Für Speisen kann bestimmt werden, ob sie vegetarisch sind. Außerdem gibt es separat dazu sogenannte Menüs (`SetMenu`), die aus Vorspeise, Hauptgang, Nachspeise und einem Getränk bestehen.

Implementieren Sie zusätzlich folgende Funktionalität:

- Es soll feststellbar sein, ob ein Menü vegetarisch ist
- Der Gesamtpreis eines Menüs soll feststellbar sein. Für ein Menü gibt es folgende Rabatte bezüglich der Einzelpreise seiner Bestandteile:
 - Es gibt 5% Rabatt wenn der Gesamtpreis der Bestandteile < 10 EUR ist.
 - Es gibt 10% Rabatt wenn der Gesamtpreis der Bestandteile ≥ 10 EUR und < 20 EUR ist.
 - Es gibt 15% Rabatt wenn der Gesamtpreis der Bestandteile ≥ 20 EUR ist.

Testen Sie auch die Korrektheit Ihrer Implementierung.

Aufgabe 3 (Singletons, 5 Punkte)

Projekt: `ex03_3`. Package: `monopoly`.

In der Vorlesung haben wurde eine Implementierung einer Strassenklasse für Monopoly gezeigt. Der Konstruktor wurde privat gemacht und die einzelnen Strassen über statische Methoden erstellbar gemacht. Eine Anmerkung in der Vorlesung war, dass aber trotzdem mehrere unterschiedliche Straßenobjekte erstellt werden könnten.

Der Code aus der Vorlesung ist schon im `.zip`-Paket von diesem Blatt enthalten. Modifizieren Sie diesen so, dass immer nur *eine* Instanz einer Straße erzeugt werden kann. Das Interface (d.h. die Art und Weise wie man eine Instanz von außerhalb der Klasse erzeugt) darf nicht verändert werden. Implementieren Sie zusätzlich zur vorhandenen „Baltic Avenue“ die Straßen „New York Avenue“ (`makeNewYork`) und „St. Charles Place“ (`makeStCharles`).

Hinweise:

- Die Aufgabe lässt sich mit dem sogenannten „Singleton Pattern“ lösen, aber es gibt auch einfachere Lösungen.
- Unter *einer Instanz* verstehen wir ein im Speicher abgelegtes Objekt. Es können verschiedene *Referenzen* (d.h. lokale Variables und Felder) bestehen, die auf eine einzige Instanz zeigen. Ob zwei Referenzen auf dieselbe Instanz zeigen, lässt sich mit dem Operator „`==`“ feststellen. Insbesondere soll hier für diese Aufgabe gelten:

```
1 Street inst1 = Street.getBaltic(...);
2 Street inst2 = Street.getBaltic(...);
3 assertTrue(inst1 == inst2);
```

¹In der Vorlesung wurden Listen (oder ähnliche Datenstrukturen) schon benutzt. Unter dem Stichwort *Java Collections Framework* finden Sie auch im Netz ausreichend Informationen.