Ludwig-Maximilians-Universität München
Institut für Informatik
PD. Dr. Peer Kröger
Johannes Niedermayer, Andreas Züfle

München, 08.01.2013

Einführung in die Programmierung
WS 2012/13

Übungsblatt 10: Mehr zu Objektorientierung, Arbeiten mit Strings

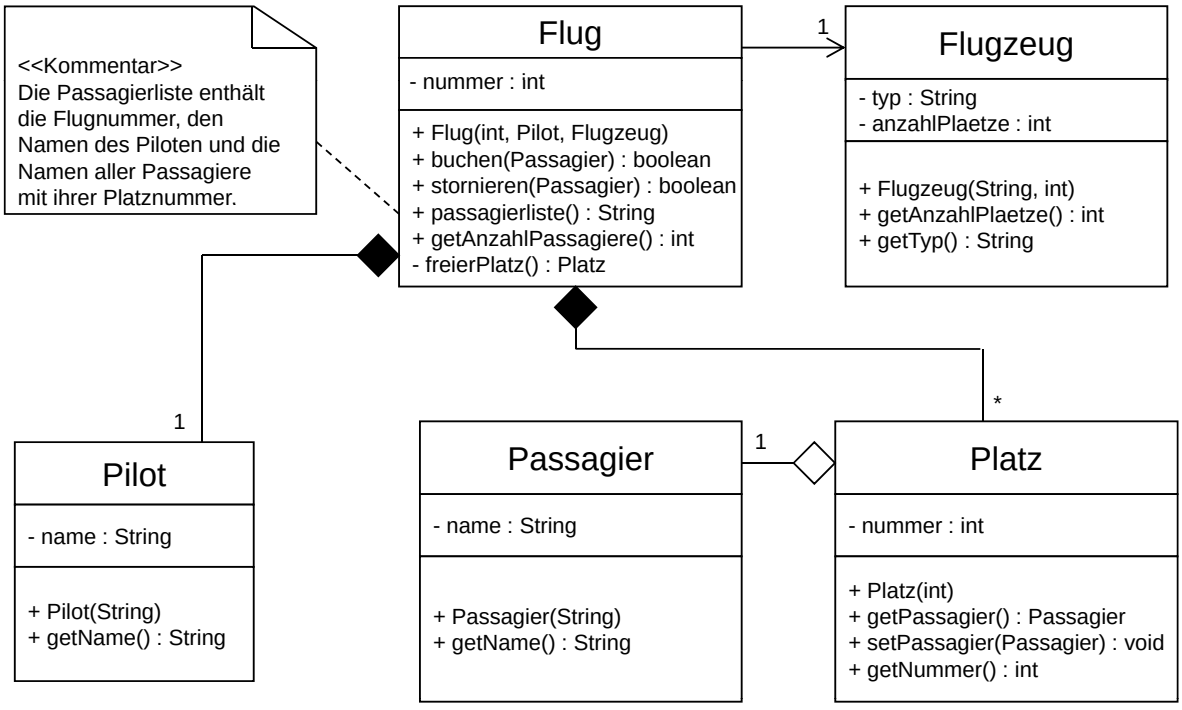
Besprechung: 16./18./21./01.2013

Ende der Abgabefrist: Dienstag, 15.01.2013 14:00 Uhr.

Hinweise zur Abgabe:
Geben Sie bitte Ihre gesammelten Lösungen zu diesem Übungsblatt in einer Datei `loesung10.zip` unter `https://uniworx.ifi.lmu.de` ab.

Aufgabe 10-1 *UML* **0 Punkte**

Gegeben ist folgendes UML-Klassendiagramm zur Modellierung von Flugbuchungen:

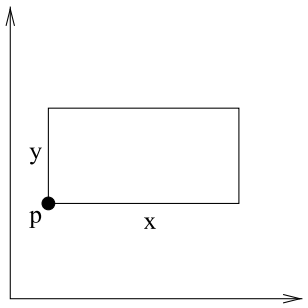


Implementieren Sie diese Klassen in Java und geben Sie sie ab.

Aufgabe 10-2 *Objektorientierung* **10 Punkte**

In dieser Aufgabe sollen Sie geeignete Klassenstrukturen zur Verwaltung von Rechtecken implementieren. Bei Rechtecken soll es sich um Objekte handeln, deren Kanten achsenparallel in einem zweidimensionalen Koordinatensystem liegen. Rechtecke, die schief zu den Koordinatenachsen liegen, brauchen Sie nicht zu beachten!

Rechtecke werden folgendermaßen repräsentiert (siehe Bild): Gespeichert werden die Koordinaten eines der vier Eckpunkte p und die Länge der beiden an p angrenzenden Kanten x (parallel zur x-Achse) und y (parallel zur y-Achse).



(a) Implementieren Sie eine Klasse `Punkt` zur Verwaltung zweidimensionaler Punkte in einem reellen Koordinatensystem. Definieren Sie neben einem geeigneten Konstruktor und geeigneten Attributen folgende Methoden:

- **`void`** `bewege(`**`double`** `betrag)`:
  bewegt den Punkt um den angegeben reellen Wert weg vom Koordinatenursprung (Skalarmultiplikation).
- **`void`** `spiegle()`:
  spiegelt den Punkt am Ursprung des Koordinatensystems
- **`void`** `verschiebeX(`**`double`** `betrag)`:
  verschiebt den Punkt um den angegebenen Wert entlang der x-Achse.
- **`void`** `verschiebeY(`**`double`** `betrag)`:
  verschiebt den Punkt um den angegebenen Wert entlang der y-Achse.

(b) Implementieren Sie eine Klasse `Rechteck`, die die Klasse `Punkt` zur Darstellung eines Rechtecks verwendet. Die Klasse soll neben einem geeigneten Konstruktor und geeigneten Attributen folgende Methoden bereitstellen:

- **`double`** `diagonale()`:
  berechnet die Länge der Diagonale des Rechtecks und gibt den Wert zurück.
- **`double`** `flaeche()`:
  berechnet die Fläche des Rechtecks und gibt den Wert zurück.
- **`void`** `bewege(`**`double`** `betrag)`:
  bewegt das Rechteck um den angegebenen reellen Wert weg vom Koordinatenursprung.
- **`void`** `spiegle()`:
  spiegelt das Rechteck am Ursprung des Koordinatensystems.
- **`void`** `verschiebeX(`**`double`** `betrag)`:
  verschiebt das Rechteck um einen bestimmten Wert entlang der x-Achse.
- **`void`** `verschiebeY(`**`double`** `betrag)`:
  verschiebt das Rechteck um einen bestimmten Wert entlang der y-Achse.

Achten Sie bei Ihrer Implementierung auf eine sinnvolle Datenkapselung und geben Sie für diese Aufgabe Ihre Dateien `Punkt.java` und `Rechteck.java` ab.

Aufgabe 10-3 *Bruchrechnen mit Java* **0 Punkte**

In dieser Aufgabe soll eine Klasse `Bruch` implementiert werden, die Brüche (rationale Zahlen) und die Grundrechenarten für Brüche implementiert.

(a) Definieren Sie eine Klasse `Bruch`, die (gewöhnliche) Brüche mit Zähler und Nenner vom Typ **`int`** repräsentiert. Deklarieren Sie dazu entsprechende Attribute.

(b) Erweitern Sie die Klasse `Bruch` um einen Konstruktor `Bruch(`**`int`**` n)`, der eine ganze Zahl `n` als Bruch erzeugt.

(c) Erweitern Sie die Klasse `Bruch` um eine Methode `String toString()`, die ein `String`-Objekt zurückgibt, das einen entsprechenden Bruch geeignet textuell repräsentiert.

(d) Erweitern Sie die Klasse `Bruch` um eine Methode `Bruch negiere()`, die einen neuen Bruch erzeugt, so dass gilt: Ist b ein Bruch mit Wert z/n, so ist `b.negiere()` ein Bruch mit Wert $-z/n$.

(e) Erweitern Sie die Klasse `Bruch` um folgende Methoden:

- `Bruch addiere(Bruch b)`
- `Bruch subtrahiere(Bruch b)`
- `Bruch multipliziere(Bruch b)`
- `Bruch dividiere(Bruch b)`

Jede dieser Methoden soll einen neuen Bruch zurückgeben, der den Wert der Berechnung repräsentiert. Es gilt:

$$\frac{z_1}{n_1} + \frac{z_2}{n_2} = \frac{z_1 n_2 + z_2 n_1}{n_1 n_2}, \frac{z_1}{n_1} - \frac{z_2}{n_2} = \frac{z_1 n_2 - z_2 n_1}{n_1 n_2}, \frac{z_1}{n_1} \times \frac{z_2}{n_2} = \frac{z_1 z_2}{n_1 n_2}, \frac{z_1}{n_1} / \frac{z_2}{n_2} = \frac{z_1 n_2}{n_1 z_2}$$

Testen Sie diese Methoden für geeignete Eingaben.

(f) Erweitern Sie die Klasse `Bruch` um eine Methode **`boolean`** `kleinerAls(Bruch b)`, die folgende Werte zurückgibt: **`true`**, wenn `b` grösser ist als der Wert des aufrufenden Objekts vom Typ `Bruch`, **`false`** sonst.

(g) Erweitern Sie die Klasse `Bruch` um eine Methode **`double`** `alsDouble()`, die das aufrufende Objekt vom Typ `Bruch` in eine Gleitkommazahl umwandelt.

Aufgabe 10-4 *Generalisierungshierarchie* **0 Punkte**

Beschreiben Sie eine Generalisierungshierarchie, die die *is a*-Beziehungen für folgende Begriffe angibt:

```
Adler, Affe, Ahorn, Amsel, Andreas Züfle, Baum, Computer,
Delfin, Flipper, Hund, Kastanie, Kiefer, Küchengerät, Lassie,
Laubbaum, Lebewesen, Maschine, Mensch, Mikrowelle,
Nadelbaum, Peer Kröger, Pflanze, Säugetier, Stuhl,
Tanne, Taube, Tier, Toaster, Vogel
```

Die Hierarchie soll dabei einfach durch Einrückung dargestellt werden. Achten Sie darauf, zur Einrückung Leerzeichen zu verwenden, nicht Tabulatoren, um dauerhafte Lesbarkeit für die Korrektoren zu gewährleisten.

Beispiel:

```
Transportmittel
  Fährschiff
    Autofähre
    Personenfähre
  Straßenfahrzeug
    Auto
    Lastwagen
```

Aufgabe 10-5 *Vererbung* **6+0+0 Punkte**

Wir betrachten folgenden Ausschnitt aus der Realität: Es gibt Schiffe und als Spezialfälle davon Segelschiffe und Motorschiffe. Jedes Schiff hat eine Tonnage (in Bruttoregistertonnen, aber das ist für die Aufgabe unwichtig), jedes Segelschiff hat zusätzlich zur Tonnage eine Segelfläche (in Quadratmetern, ebenfalls für die Aufgabe unwichtig) und jedes Motorschiff hat zusätzlich zur Tonnage eine Motorleistung (in Kilowatt, ebenfalls für die Aufgabe unwichtig). Segelschiffe haben keine Motorleistung und Motorschiffe haben keine Segelfläche, andere Schiffe haben weder das eine noch das andere.

(a) Definieren Sie eine Klasse `Schiff` und zwei Unterklassen `Segelschiff` und `Motorschiff` zur Repräsentation dieses Ausschnitts der Realität. Die Datei `Schifffahrt.java`, die Sie auf der Homepage finden, enthält ein Hauptprogramm, das die drei zu definierenden Klassen verwendet, das aber für diese Teilaufgabe nicht verändert werden soll.

Tonnage, Motorleistung und Segelfläche sollen durch **`private`** Attribute (Instanzvariablen) der entsprechenden Klassen repräsentiert sein, deren Werte als Parameter der Konstruktoren mitgeliefert werden. Zur Implementierung der Konstruktoren ist es in einigen Fällen sinnvoll, **`super`**`(...)` zu verwenden.

Jede der drei Klassen soll eine Methode `toString()` definieren, die eine Textdarstellung des jeweiligen Objekts liefert. Das Format der Textdarstellung entnehmen Sie den Kommentaren hinter den Ausgabeanweisungen in `Schifffahrt.java`. Auch bei der Implementierung dieser Methoden ist **`super`** sinnvoll, aber diesmal ohne Klammern.

(b) Schließen Sie die Deklarationen der Methode `toString()` in allen drei Klassen in `/*` und `*/` ein, so dass sie vorübergehend entfernt werden. Das Programm bleibt trotzdem übersetzbar und ablauffähig. Erklären Sie, warum das so ist und welcher Unterschied zu der Version mit den Methoden besteht. Schreiben Sie diese Erklärung als Kommentar hinter das Hauptprogramm in die Datei `Schifffahrt.java`.

(c) Reaktivieren Sie Ihre Deklarationen der Methode `toString()` und übersetzen Sie alle Dateien neu. Starten Sie das Programm einmal mit `java Schifffahrt` und einmal mit `java Schifffahrt xyz` und vergleichen Sie die Ausgabe für `c`.

Erklären Sie anhand dieses Beispiels, warum ein Java-Compiler im allgemeinen nicht feststellen kann (zur Compile-Zeit), welcher Java-Programmcode bei einem Methodenaufruf ausgeführt werden muss. Schreiben Sie diese Erklärung ebenfalls als Kommentar ans Ende der Datei `Schifffahrt.java`.

Geben Sie die Klassen `Schiff`, `Segelschiff` und `Motorschiff` sowie die Datei `Schifffahrt.java` mit den Kommentaren für (b) und (c) ab.