Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. C. Böhm, G. Zhang

Übungen zu Einführung in die Informatik

Freiwillige Abgabe bis zum 26. 11. 2007, 12:00h

Aufgabe 5-1 Quadrat

In der Vorlesung vom 17. 10. wurde Ihnen die Klasse `Square` (Siehe Skript, Seite 23) vorgestellt. Dabei wurde die Methode `square` als `public` deklariert. Deklarieren Sie die Methode nun als `private` und kompilieren Sie die Klasse `Programm` (Seite 24. Laden Sie bitte die *neue* Folie vom 26. 10. herunter). Welche Fehlermeldung erhalten Sie? Was bedeutet sie?

Aufgabe 5-2 Punkte

Laden Sie die Klasse `Point` von unserer Web-Seite herunter.

a) Vergleichen Sie die Klasse mit der Klasse `Point`, die Ihnen aus der Vorlesung bekannt ist (Siehe Skript, Seite 5). Welche syntaktischen Unterschiede können Sie erkennen?

b) Sind die zwei Klassen auch *semantisch* unterschiedlich? Das heißt, können die zwei Klassendefinitionen in einem Programm gegeneinander ausgetauscht werden? Warum?

c) Erweitern Sie die Klasse `Point` um eine Methode `public double distance(Point p)`, welche die Distanz zwischen dem impliziten Parameter und dem Punkt `p` zurückgibt. Für zwei Punkte $p_1 = (x_1, y_1)$ und $p_2 = (x_2, y_2)$ gilt $distance(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.
Hinweis: Der *implizite Parameter* einer Methode ist das mit `this` gekennzeichnete Objekt. Die (statische) Methode `Math.sqrt(double a)` gibt die Quardratwurzel von a zurück.

d) Testen Sie die Methode.

Aufgabe 5-3 Bruchrechnen

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

a) Definieren Sie eine Klasse `Fraction`, die (gewöhnliche) Brüche mit Zähler und Nenner vom Typ `int` repräsentiert. Deklarieren Sie dazu Instanzvariablen `numerator` und `denominator` vom Type `int`, die Zähler bzw. Nenner des Bruchs repräsentieren.
Hinweis: Attribute werden auch *Instanzvariablen* genannt.

b) Erweitern Sie die Klasse `Fraction` um eine Methode `public String toString()`, die ein `String` zurückgibt, das den impliziten Parameter textuell darstellt.

c) Erweitern Sie die Klasse `Fraction` um eine Methode `int abs(int n)`, die den Absolutbetrag von `n` berechnent. Dabei gilt:

$$abs(n) = \begin{cases} n, & falls\ n \geq 0 \\ -n, & sonst \end{cases}$$

d) Erweitern Sie die Klasse `Fraction` um einen Konstruktor `Fraction(int m, int n)`, der (für n≠0) den Bruch `m/n` erzeugt. Für den erzeugten Bruch soll gelten:

- `denominator` > 0 und
- `Divisor.gcd(numerator, denominator) = 1`.

Hinweis: Die Methode `Divisor.gcd` zum Berechnen des größten gemeinsamen Teilers zweier ganzer Zahlen kennen Sie aus Übungsblatt 3. Auch die Methode `abs` leistet Ihnen gute Dienste.

e) Erweitern Sie die Klasse `Fraction` und einen Konstruktor `Fraction(int n)`, der eine Repräsentation der ganzen Zahl `n` als Bruch erzeugt.

f) Erweitern Sie die Klasse `Fraction` um einen Konstruktor `Fraction()`, der einen Bruch erzeugt, dessen Wert eine Zufallszahl zwischen 0 und 1 ist.
Hinweis: Mit `Math.random()` kann eine `double`-Zahl d generiert werden, wobei $0 \leq d < 1$ gilt.

g) Erweitern Sie die Klasse `Fraction` um eine Methode `Fraction negate()`, die einen neuen Bruch erzeugt, so dass gilt: Ist p ein Bruch mit Wert m/n, so ist `p.negate()`ein Bruch mit Wert $-m/n$.

h) Erweitern Sie die Klasse `Fraction` um die Methoden `Fraction add(Fraction q)`, `Fraction sub(Fraction q)`, `Fraction mult(Fraction q)`, und `Fraction div(Fraction q)`, die Addition, Subtraktion, Multiplikation und Division von Brüchen beschreiben. Jede dieser Methoden soll einen neuen Bruch zurückgeben, der den Wert der Berechnung repräsentiert. Es gilt:

$$\frac{m_1}{n_1} + \frac{m_2}{n_2} = \frac{m_1 n_2 + m_2 n_1}{n_1 n_2}, \frac{m_1}{n_1} - \frac{m_2}{n_2} = \frac{m_1 n_2 - m_2 n_1}{n_1 n_2}, \frac{m_1}{n_1} \times \frac{m_2}{n_2} = \frac{m_1 m_2}{n_1 n_2}, \frac{m_1}{n_1} / \frac{m_2}{n_2} = \frac{m_1 n_2}{n_1 m_2}$$

i) Erweitern Sie die Klasse `Fraction` um eine Methode `boolean lessThan(Fraction rhs)`, die folgende Werte zurückgibt: `true`, wenn `rhs` größer ist als der implizite Parameter, `false` sonst.

j) Erweitern Sie die Klasse `Fraction` um eine Methode `double asDouble()`, die den impliziten Parameter in eine Gleitkommazahl umwandelt.

k) Modellieren Sie die Klasse `Fraction` in einem UML-Klassendiagramm.