Übungsblatt 1: Interne Variablen verwenden

Quadratische Funktion

Das Projekt `01 QuadratischeFunktion` im Tauschverzeichnis enthält eine Klasse, die eine quadratische Funktion modelliert.

- Überlegen Sie, welche Informationen gespeichert werden müssen, um eine quadratische Funktion vollständig zu beschreiben.
- Schreiben Sie einen Konstruktor, um eine quadratische Funktion zu erzeugen.
- Die Klasse soll die folgenden Methoden anbieten:
  - `getFunktionswert(double x)`: liefert den Funktionswert an einer Stelle x
  - `getScheitelX()`: liefert den x-Wert des Scheitels
  - `getScheitelY()`: liefert den y-Wert des Scheitels
  - `getAnzahlNullstellen()`: liefert die Anzahl der Nullstellen

Ersetzen Sie an den Stellen, an denen noch TODO steht, den bestehenden Code durch Ihre Implementation.
Klicken Sie links auf „Tests starten“, um automatisch 100 Testfälle ausführen zu lassen.

Notenverwaltung

Das Projekt `02 Klausuren` im Tauschverzeichnis enthält eine Klasse namens `Klausur`, die das Ergebnis einer Oberstufenklausur modelliert. Sie bietet die folgenden Methoden an:

- `getAnzahl()` – die Gesamtzahl der erfassten Noten
- `getDurchschnitt()` – die Durchschnittsnote der erfassten Noten
- `getBesteNote()` und `getSchlechtesteNote()` – die beste bzw. schlechteste erfasste Note
- `noteEintragen(int note)` – fügt eine neue Note zur Erfassung hinzu
- `reset()` – löscht alle bisher erfassten Einträge aus dem Kurs

Überlegen Sie sich, wie Sie das Ergebnis modellieren wollen. Es sollen wie üblich keine Variablen nach außen hin sichtbar sein – der Zugriff darf nur über die oben aufgezählten Methoden geschehen.
Implementieren Sie die Methoden, in denen noch TODO steht.
Klicken Sie links auf „Tests starten“, um automatisch 100 Testfälle ausführen zu lassen.

Brüche

Das Projekt `03 Brüche` im Tauschverzeichnis enthält eine Klasse namens `Bruch`, die einen Bruch repräsentiert.
Sie bietet die folgenden öffentlichen Methoden an:

- `Bruch addieren(Bruch b)`
- `Bruch subtrahieren(Bruch b)`
- `Bruch multiplizieren(Bruch b)`
- `Bruch dividieren(Bruch b)`
- `int getZaehler()`
- `int getNenner()`

Die ersten vier Methoden führen die Grundrechenarten mit dem aktuellen Bruch („`this`“) und dem übergebenen Bruch `b` aus.
Das Ergebnis ist stets ein neues Objekt, d.h. `this` ändert sich durch einen Methodenaufruf nicht. Orientieren Sie sich an der Methode `multiplizieren`, um zu sehen, wie ein neues `Bruch`-Objekt erzeugt und zurückgegeben wird.

Dazu kommen die Konstruktoren:

- `public Bruch(int zaehler, int nenner)`
- `public Bruch(int wert)` – erzeugt einen Bruch mit dem Nenner 1.

Der Nenner eines Bruchs muss immer positiv sein. Dafür sollte im Konstruktor gesorgt werden. Der Aufruf `new Bruch(1,2)` sollte also den gleichen Bruch erzeugen wie `new Bruch(-1,-2)`.
Implementieren Sie alle Stellen, an denen derzeit noch TODO steht.

Beispiel für die Verwendung:

```
Bruch a = new Bruch(1, 3); // repräsentiert die Zahl 1/3
Bruch b = new Bruch(1, 4); // repräsentiert die Zahl 1/4
Bruch c = a.addieren(b);
// c repräsentiert 1/3 + 1/4 = 7/12
// a ist weiterhin 1/3, b ist weiterhin 1/4
```

Lassen Sie die 100 Testfälle in der Testklasse `BruchTester` ausführen.

Für Fortgeschrittene:
Sorgen Sie dafür, dass Ihre Brüche immer weitestmöglich gekürzt sind. Um kürzen zu können, benötigt man den größten gemeinsamen Teiler von Zähler und Nenner. Diesen erhält man effizient mit dem Euklidischen Algorithmus (→ Google oder Wikipedia). Lassen Sie dann die Testfälle in der Testklasse `BruchTester2` ausführen.

Ganzrationale Funktionen

Das Projekt `04 Funktionen` im Tauschverzeichnis enthält einige Klassen, die das Zeichnen von Funktionen sowie der Tangente ihres Schaubilds ermöglichen.
Zunächst beschäftigen wir uns mit den ganzrationalen Funktionen, d.h. Funktionen der Gestalt$f(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_2 \cdot x^2 + a_1 \cdot x + a_0$
Die Klasse `GanzrationaleFunktion` repräsentiert eine solche Funktion. Sie bietet zwei Methoden an, die Sie implementieren müssen:

- `double getFunktionsWert(double x)` – gibt den Funktionswert zu einer gegebenen Stelle x zurück.
- `Funktion getAbleitung()` – gibt die Ableitungsfunktion der Funktion zurück. Die Ableitung einer ganzrationalen Funktion ist wieder eine ganzrationale Funktion, d.h. es wird ein neues Objekt der Klasse `GanzrationaleFunktion` zurückgegeben.

Dem Konstruktor wird ein Array aus `double`-Werten übergeben, die die Koeffizienten a_0, a_1, … repräsentieren sollen.
Die Klasse `FunktionsTester` ist zum Starten der graphischen Ausgabe da. Im Konstruktor wird ein Funktionsobjekt erzeugt (vgl. das vorhandene Beispiel). Um Ihre Implementation zu testen, erzeugen Sie ein `FunktionsTester`-Objekt und rufen Sie seine Methode `anzeigen()` auf.

Für Fortgeschrittene: Informieren Sie sich über das Horner-Schema zur effizienten Berechnung von Funktionswerten von ganzrationalen Funktionen.