Übungsblatt 6: Vertiefung zu Klassen

Deadline: 20.11.2025 12:00 Uhr
Besprechung: 24.11.–28.11.2025

Hinweise für die Bearbeitung der Aufgaben

- Bitte lesen Sie die Aufgabenstellungen genau durch.
- Testen Sie Ihre Lösungen vor der Abgabe selbst!
- Halten Sie sich an die Java Coding Konventionen (siehe letzte Seite).
- Damit die automatischen Tests korrekte Ergebnisse liefern können, achten Sie bitte auf die folgenden Punkte:
  - Halten Sie sich bitte genau an die Vorgaben/Vorlagen aus der Aufgabenstellung.
  - Wenn Sie Umlaute verwenden, stellen Sie die Kodierung Ihrer Dateien unbedingt auf UTF-8.
  - Achten Sie darauf, dass Ihr Projekt mit Java Version 21 kompiliert und ausgeführt werden kann.
  - Ihre Java-Klassen müssen sich im default package befinden (es darf kein package angegeben werden).

Aufgabe 1 Abgabe über GATE mit Tests
Pythagoras

Sie kennen vermutlich alle den berühmten Satz des Pythagoras: $a^2 + b^2 = c^2$.

Implementieren Sie eine Klasse `Pythagoras` mit einer statischen Methode

`double calculate(double a, double b),`

welche zu gegebenen `a` und `b` den Wert von `c` berechnet.
Tipp: Sie können in Java mithilfe von `Math.sqrt` eine Quadratwurzel berechnen.

Aufgabe 2 Abgabe über GATE mit Peer-Review
Punkte

In dieser Aufgabe modellieren Sie etwas grundlegende Geometrie. Diese spielt sich im zweidimensionalen kartesischen Koordinatensystem ab, deren Grundbausteine Punkte sind.

a) Schreiben Sie eine Klasse `Point` mit zwei Koordinatenwerten `x` und `y`.

   Geben Sie der Klasse einen Konstruktor `Point(double x, double y)`, der einen Punkt mit x-Koordinate `x` und y-Koordinate `y` erzeugt.

   Geben Sie der Klasse außerdem Methoden `double getX()`, `double getY()`, `void setX(double x)` und `void setY(double y)`, mit denen die Koordinaten abgefragt und geändert werden können.

b) Implementieren Sie zwei weitere Methoden in der Klasse `Point`:

   - `double distanceTo(Point other)`, die den euklidischen Abstand zwischen zwei Punkten berechnet. Hierbei können Sie die Klasse `Pythagoras` aus Aufgabe 1 verwenden.
   - `void shift(double xShift, double yShift)`, die einen Punkt um `xShift` nach rechts und um `yShift` nach oben verschiebt.

Aufgabe 3 Abgabe über GATE mit Tests
Dreiecke

In dieser Aufgabe nutzen Sie die Punkte aus Aufgabe 2, um Dreiecke zu implementieren.

a) Erstellen Sie eine Klasse `Triangle`, die Dreiecke modelliert und deren Datenelemente die drei Eckpunkte des Dreiecks sind.

Geben Sie der Klasse einen Konstruktor:

```
Triangle(Point A, Point B, Point C)
```

Stellen Sie außerdem folgende öffentliche Methoden bereit:

- **`boolean`** `isEquilateral()`, die prüft, ob das Dreieck gleichseitig ist,
- **`boolean`** `isIsosceles()`, die prüft, ob das Dreieck gleichschenklig ist,
- **`boolean`** `isAcute()`, die prüft, ob das Dreieck spitzwinklig ist (alle Winkel kleiner als ein rechter),
- **`boolean`** `isRight()`, die prüft, ob das Dreieck rechtwinklig ist, und
- **`boolean`** `isObtuse()`, die prüft, ob das Dreieck stumpfwinklig ist (einen Winkel hat, der größer ist als ein rechter).

Hinweise:

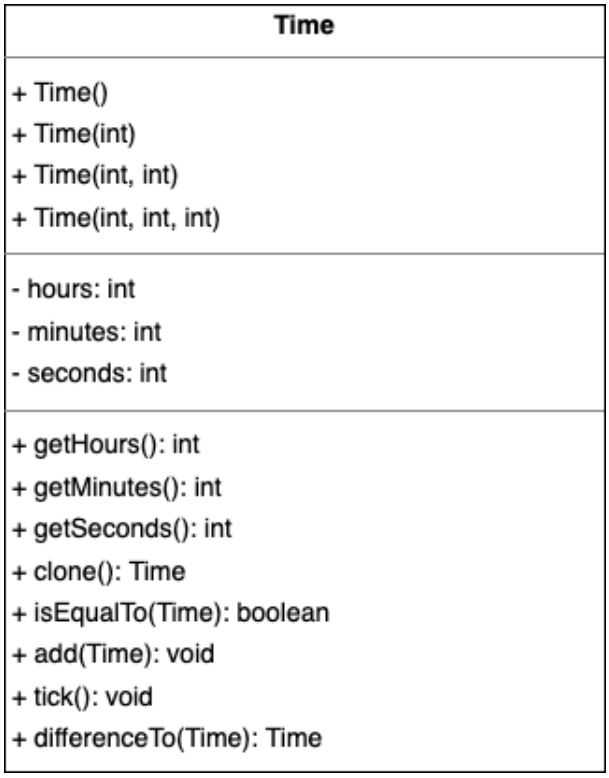
- Auch für diesen Aufgabenteil kann Ihnen die statische Methode der Klasse `Pythagoras` gute Dienste leisten.
- Ein Winkel eines Dreiecks ist:
  - kleiner als ein rechter, wenn: $a^2 + b^2 > c^2$
  - ein rechter, wenn: $a^2 + b^2 = c^2$
  - größer als ein rechter, wenn: $a^2 + b^2 < c^2$
  - wobei a und b die anliegenden Seiten und c die gegenüberliegende Seite des Winkels sind.
- Außerdem kann es ratsam sein, weitere private Methoden zu implementieren, die innerhalb der öffentlichen Methoden aufgerufen werden.

b) Haben Sie in Ihrer Implementierung der Methoden `isEquilateral()`, `isIsosceles()` und `isRight()` **`double`**s auf Gleichheit geprüft? Warum kann das problematisch sein? Wie kann man diese Probleme umgehen?

   Tipp: Falls Ihre GATE Abgabe nicht alle Tests besteht, könnte dies der Grund ein.

Aufgabe 4 Abgabe in GATE mit Test
Uhrzeiten

In dieser Aufgabe stellen Sie mit der Klasse `Time` eine Uhrzeit zur Verfügung:



Sie hat als Attribute:

- Stunden zwischen 0 und 23
- Minuten zwischen 0 und 59
- und Sekunden zwischen 0 und 59

mit entsprechenden Gettern.

Die Klasse hat vier Konstruktoren mit 0-3 Parametern. Die Parameter geben Stunden, Minuten und Sekunden vor, wobei fehlende Parameter mit 0 angenommen werden. So soll etwa `Time(12)` exakt die Mittagszeit darstellen. Werte außerhalb der sinnvollen Spanne der Uhrzeit werden auf 0 initialisiert.

Die Uhrzeit hat folgende Methoden:

- eine Kopier-Methode `Time clone()`, die eine neue Uhrzeit mit den selben Werten erzeugt

- eine Methode, die prüft, ob die Uhrzeit dieselbe ist wie eine andere
- eine Methode, die die Uhrzeit um die andere gegebene Uhrzeit vor stellt
- `void tick()`, die die Uhrzeit um eine Sekunde vor stellt
- (**Schwer**) eine Methode, die den Abstand zwischen zwei Uhrzeiten angibt. (Oder in anderen Worten: Wie viel Zeit muss vergehen um von der ersten Uhrzeit zur zweiten zu kommen?) Der Abstand ist immer positiv, und kann auch über Mitternacht hinweg gemessen werden. So beträgt etwa der Abstand zwischen 23:59:55 und 00:00:05 zehn Sekunden (00:00:10).

  Tipp: Es kann hilfreich sein die privaten Hilfsmethoden
  `void subtractHours(int hours)`,
  `void subtractMinutes(int minutes)` und
  `void subtractSeconds(int seconds)`,
  welche die angegebene Zeitspanne abziehen, zu implementieren und zu benutzen. Alternativ können Sie auch alle Uhrzeiten erst in Sekunden umrechnen, den Abstand berechnen und aus den Sekunden wieder eine Uhrzeit erzeugen.

Bei den Implementierungen können und sollen andere, bereits implementierte Methoden verwendet werden. Zum Testen kann eine `toString`-Methode hilfreich sein:

Time to String

```
@Override
public String toString() {
    return String.format(
            "Time(hours: %d, minutes: %d, seconds: %d)",
            hours,
            minutes,
            seconds
    );
}
```

Abgabe

Die Abgabe erfolgt bis zum 20.11.2025 12:00 Uhr auf GATE, https://gate.ifi.lmu.de.

Zuletzt aktualisiert: 24. Oktober 2025

Java Coding Konventionen

Zur Erstellung eines lesbaren und wartbaren Codes sind Coding Conventions sehr hilfreich. Im Folgenden sind einige der wichtigsten Konventionen für Java aufgeführt:

- Achten Sie auf gute Lesbarkeit Ihrer Programme und rücken Sie Anweisungen immer entsprechend ein (üblich sind 2 oder 4 Leerzeichen bzw. ein Tabulator pro Ebene).
- Kommentare sollten verwendet werden, um den Code zu erklären an den Stellen, an denen Ihre Algorithmus-Idee nicht offensichtlich ist und nicht um ihn zu wiederholen.
- Leerzeichen sollten verwendet werden, um den Code lesbarer zu machen (z. B. nach Kommas, um Operatoren herum). Nutzen Sie diese auch vor und nach Klammern (z. B. `if (condition) { ... }`).
- Java ist eine Case-sensitive Sprache. `Variable`, `variable` und `VARIABLE` sind drei verschiedene Bezeichner.
- Klassen beginnen mit einem Großbuchstaben, z. B. `HelloWorld`.
- Methoden und Variablen beginnen mit einem Kleinbuchstaben, z. B. `main`, `firstName`.
- Mehrteilige Bezeichner werden im camelCase-Stil geschrieben, z. B. `firstName`, `calculateSum`.
- Konstanten werden in Großbuchstaben mit Unterstrichen geschrieben, z. B. `MAX_VALUE`.
- Verwenden Sie verständliche bzw. sprechende Bezeichner für Variablen und Methoden.
- Verwenden Sie geschweifte Klammern auch für einzeilige Blöcke.
- Leerzeilen verbessern die Lesbarkeit, indem sie logisch zusammenhängende Abschnitte des Codes visuell trennen, z. B. zwischen Methoden, Definition von Klassen und Interface, lokalen Variablen in einer Methode und der ersten Anweisung.

Weitere Informationen finden Sie in den offiziellen Java Coding Conventions: https://www.oracle.com/java/technologies/javase/codeconventions-contents.html