Modellierung und Programmierung 1

Übung 9

Stefan Preußner

18./ 19. Januar 2021

Interfaces

(Wiederholung)

Schnittstellen (Interfaces)

- In Java gibt es keine Mehrfachvererbung, eine Klasse kann also nicht von mehreren Klassen erben
- Soll eine Klasse mehrere Typen haben, so kann sie **Schnittstellen** - **Interfaces** - implementieren
- Eine Schnittstelle legt fest, welche Methoden eine Klasse besitzen muss, stellt aber selbst i.d.R. keine Implementierung zur Verfügung
  - In Interfaces können aber, u.a. aus Gründen der Rückwärtskompatibilität, `default`-Methoden implementiert werden

Interfaces erstellen

- Ein Interface kann mit der Syntax

```
<Sichtbarkeit> interface <Name>
```

  deklariert werden
- Beispiel:

```
public interface Connection
```

- Die Sichtbarkeit ist auf `public` und *package* (also das Fehlen eines Modifizierers) beschränkt
- Ein Interface kann mit dem Schlüsselwort `extends` andere Interfaces erweitern:

```
public interface Connection extends AutoCloseable, Wrapper
```

Interfaces implementieren

- Ein Interface kann mit dem Schlüsselwort `implements` durch eine Klasse implementiert werden:

```
public class NetworkConnection implements Connection
```

- Eine Klasse kann beliebig viele Interfaces implementieren:

```
public class NetworkConnection implements
    Connection, Resettable
```

Interfaces implementieren

- Eine Klasse **muss** alle Methoden eines Interfaces implementieren
  - Dies gilt nicht für Klassen, welche `abstract` sind (von abstrakten Klassen können keine Instanzen erzeugt werden, daher müssen sie keine Methoden implementieren)
  - Dies gilt nicht für Methoden, für welche es eine `default`-Implementierung im Interface gibt
- Implementierte Methoden **müssen** `public` sein

Interfaces - Konstruktoren und Methoden

- Von einer Schnittstelle können keine Objekte erzeugt werden, nur von den sie implementierenden Klassen
- Eine Schnittstelle darf deshalb keinen Konstruktor haben
- Die Methoden eines Interfaces sind automatisch `public` und `abstract`
  - `abstract` Methoden werden nur deklariert, aber nicht implementiert
  - Die Deklaration einer Schnittstellenmethode enthält nur Modifizierer, Rückgabetyp und Signatur

Variablen in Schnittstellen

- Instanzvariablen sind immer Teil einer Implementierung; da Schnittstellen keine Implementierung enthalten, besitzen sie auch keine Instanzvariablen
- In Schnittstellen können Konstanten festgelegt werden
  - Diese sind automatisch `public`, `static` und `final`
  - Erweitert eine Schnittstelle eine andere Schnittstelle, so kann sie deren Konstanten mit eigenen Werten überschreiben

Schnittstellen und instanceof

`instanceof` funktioniert bei Schnittstellen wie bei Klassen. In dem Beispiel

```
public class NetworkConnection implements
        Connection, Resettable
```

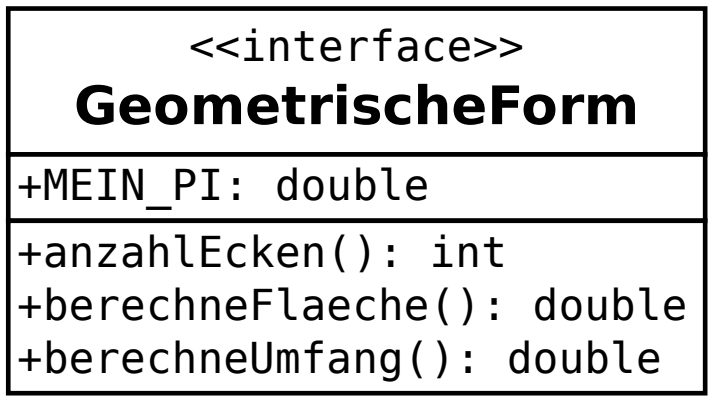
geben die Tests

```
NetworkConnection verbindung;
verbindung instanceof NetworkConnection;
verbindung instanceof Connection;
verbindung instanceof Resettable;
```

alle `true` zurück.

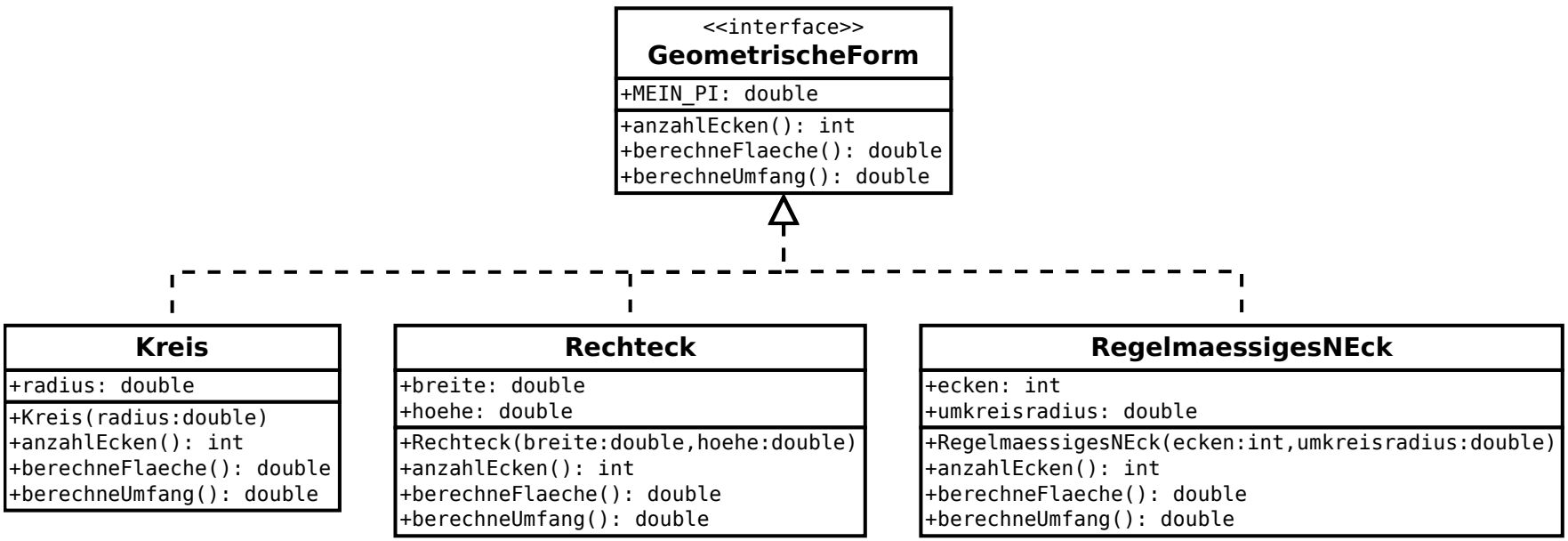
Programmierübung

Erstellen Sie die Schnittstelle `GeometrischeForm` entsprechend des folgenden UML-Diagramms:



Programmierübung

Erstellen Sie weiterhin die Klassen `Kreis`, `Rechteck` und `RegelmaessigesNEck` (für ein regelmäßiges N-Eck), welche alle `GeometrischeForm` implementieren sollen:



Pseudocode

Pseudocode

- Pseudocode dient dazu, einen Algorithmus oder ein Programm darzustellen und (für Menschen) verständlich zu machen, ohne dafür auf Bestandteile einer bestimmten Programmiersprache zurückgreifen zu müssen
- Pseudocode ist
  - weniger formal und i.d.R. verständlicher als Programmcode
  - formaler und kompakter als eine Beschreibung *im Fließtext*
- Für Pseudocode gibt es **keinen Standard**, d.h. grundsätzlich kann Pseudocode frei gestaltet werden

Umwandlung von Java-Code in Pseudocode

Java	mögliche Entsprechung im Pseudocode
`if`	FALLS
`else`	SONST
`while`	SOLANGE
`for`	FÜR
for-each-loop	FÜR ALLE
`return`	ENDE / GEBE ... ZURÜCK
`System.out.println`	GEBE ... AUS
Wertzuweisung	INITIALISIERE / SETZE
Anweisungsblock	Einrückung