Schiffe versenken – Wiederholung Grundlagen

21. November 2024

Plan für die Woche

- Syntax und Grundelemente
- Kontrollstrukturen
- Klassen und Objekte

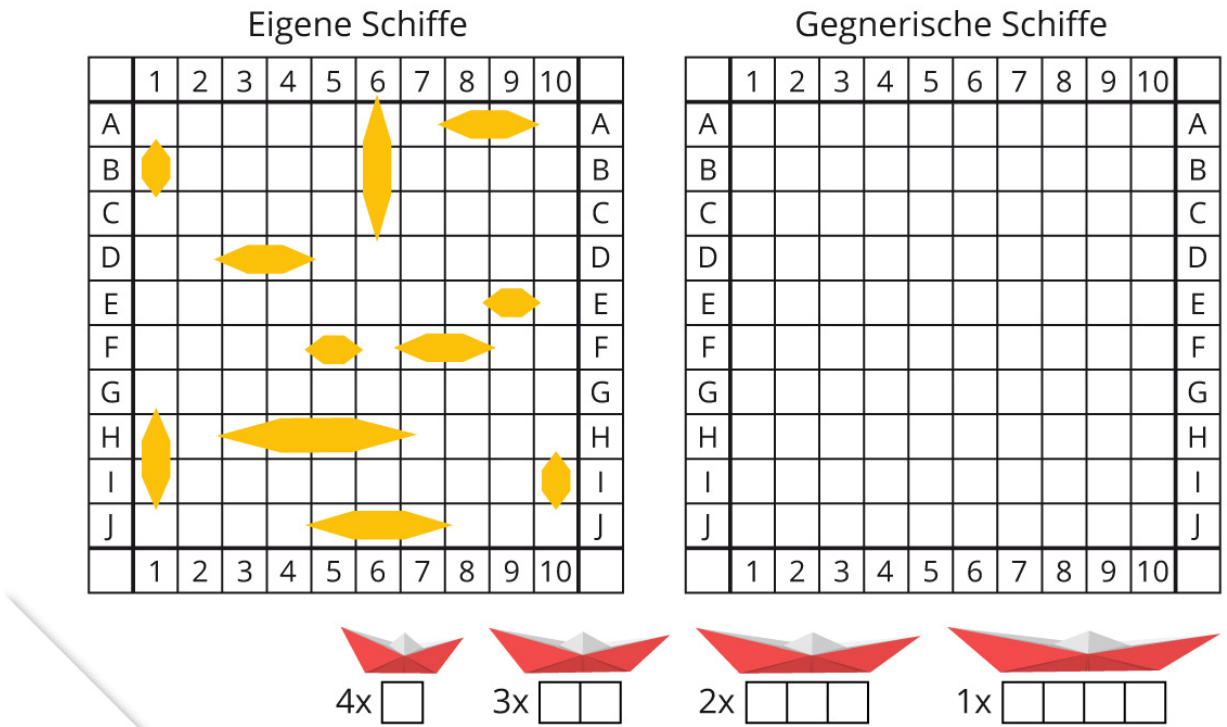
- Git Workflow
- Umgang mit großen Aufgabenstellungen
- Arrays

- Schiffe versenken

- Methoden
- OOP

- Einführung in UML

Projekt: Schiffe versenken



https://www.talu.de/schiffe-versenken/

Plan für heute

- Methoden
- Erste Grundlagen OOP

Methoden

Methoden (method)

- Bestimmen das Verhalten von Objekten
- Arbeiten mit Daten (Variablen) von Objekten

```
1  Modifier Datentyp Methodenname (Parameterliste)
2  {
3      Anweisungen;
4  }
```

Aufruf von Methoden

```
1  meinObjekt.methodeA(); // ohne Parameterliste
2
3  meinObjekt.methodeB(param1, param2); // mit Parameterliste
```

this

- Referenzvariable, die auf das aktuelle Objekt verweist
- Die this-Referenzvariable wird beim Anlegen eines Objekts automatisch erzeugt
- Zugriff auf eigene Methoden und Instanzvaraiblen des Objekts
- Explizit verwenden, wenn auf die Instanzvarible zugegriffen werden soll

```
1  public float berechneVerkaufspreis1()
2  {
3      int quadratmeterpreis = 15;
4
5      return (wollMenge * quadratmeterpreis);
6  }
7
8  public float berechneVerkaufspreis2()
9  {
10     int quadratmeterpreis = 15;
11
12     return (this.wollMenge * quadratmeterpreis);
13 }
```

Parameterliste

– Übergebene Daten, mit denen die Methode arbeiten kann

```
1  public double methodeA(); // Methode mit leerer Parameterliste
2
3  public double methodeB(int a); // Parameterliste mit einem primitiven Parameter
4
5  public double methodeC(String a); // Parameterliste mit einem Objekttyp-Parameter
6
7  public double methodeD(int a, int b);  // Parameterliste mit zwei Parametern
```

Rückgabewert - return

- Zurückgeben eines Wertes aus der Methode heraus: **return** rückgabewert;
- Festgelegter Datentyp
- Es kann ein primitiver Datentyp oder ein Objekttyp sein

```
1  public int methodeA(); // Datentyp des Rückgabewerts ist int
2
3  public String methodeB(); // Datentyp des Rückgabewerts ist String
4
5  public void methodeC(); // Kein Rückgabewert da void (leer)
```

Aufgabe

- Lege eine neue Klasse "*MatheAufgabe*" an.
- Füge eine public static void main(String[] arguments) Methode hinzu.
- Deklariere innerhalb der main-Methode eine double-Variable.
- Belege diese Variable dem Wert 5.0/3.0 und gib den Inhalt der Variable auf dem Bildschirm aus.
- Schreibe nun eine Methode public static double add(double x, double y), die die Summe der beiden übergebenen Zahlen x und y zurückgibt.

https://wiki.freitagsrunde.org/Javakurs/%C3%9Cbungsaufgaben/MathematischeMethoden

Grundlagen OOP

Was ist OOP?

- Objektorientierte Programmierung
- Konzept der Verwendung von Objekten

Quiz

Which of the following statements are true about objects in OOP?

a) Objects should only be used to represent real-world physical objects.

b) Objects use fields to store state

c) Objects provide methods to operate on ist internal state

d) Objects optimize performace of your code

Es gibt **zwei** richtige Antworten.

Pursley, Question 5

Quiz

In OOP, what is the blueprint, or prototype from which objects are created?

a) Class

b) Instance

c) Struct

d) Type

e) Varaible

Es gibt **eine** richtige Antwort.

Pursley, Question 6

Grundprinzipien

- Abstraktion (abstraction)
- Kapselung (encapsulation)
- Vererbung (inheritance)
- Polymorphismus (polymorphism)

Abstraktion (Abstraction)

- Verstecken der komplexen Details einer Implementierung
- Zeigen der wichtigen Dinge
- Aus einem definierten Konzept können mehrere Instanzen erstellt werden

- **Beispiel**: Es gibt ein Objekt *Auto*. Dafür muss nicht gewusst werden, wie das Auto fährt, sondern es reicht aus, die Methode *fahren()* aufzurufen.

Kapselung (Encapsulation)

- Variablen (Daten) und Methoden (Funktionen) werden in Klassen gespeichert
- Zugriffsmodifizierer (Access Modifiers):
  - **private**: Zugriff nur innerhalb der Klasse oder über getter() und setter()
  - **public**: Sind über das gesamte Programm zugänglich
  - **protected**: Nur Klassen des gleichen Pakets zugänglich

Wenn du **keinen Zugriffsmodifizierer** angibst, wird der Zugriff als **Paketzugriff** (auch **default access** genannt) bezeichnet.

Vererbung (Inheritance)

- Eine Klasse kann Eigenschaften und Methoden einer anderen Klasse erben („übernehmen“)

- **Beispiel**: Ich habe eine Klasse *Auto*. Wenn ich einen *BWM* erstellen möchte, kann ich von der Klasse *Auto* erben und die Eigenschaften aus der Klasse *Auto* übernehmen.

Polymorphismus (Polymorphism)

- Eine Methode kann unterschiedliche Implementierungen haben, je nach Objekt
- Überschriebene Methoden mit @Override kenntlich machen

- **Beispiel:** Mein *BMW*, der von *Auto* geerbt hat, kann anders *fahren()* als das allgemeine *Auto*.

Quiz

Consider the following code snippit: class MountainBike extends Bicycle {}

Which OOP concept is this an example of?

a) Composition

b) Encapsulation

c) Idempotency

d) Inheritance

Es gibt **eine** richtige Antwort.

Pursley, Question 7

Aufgabe in Intellij

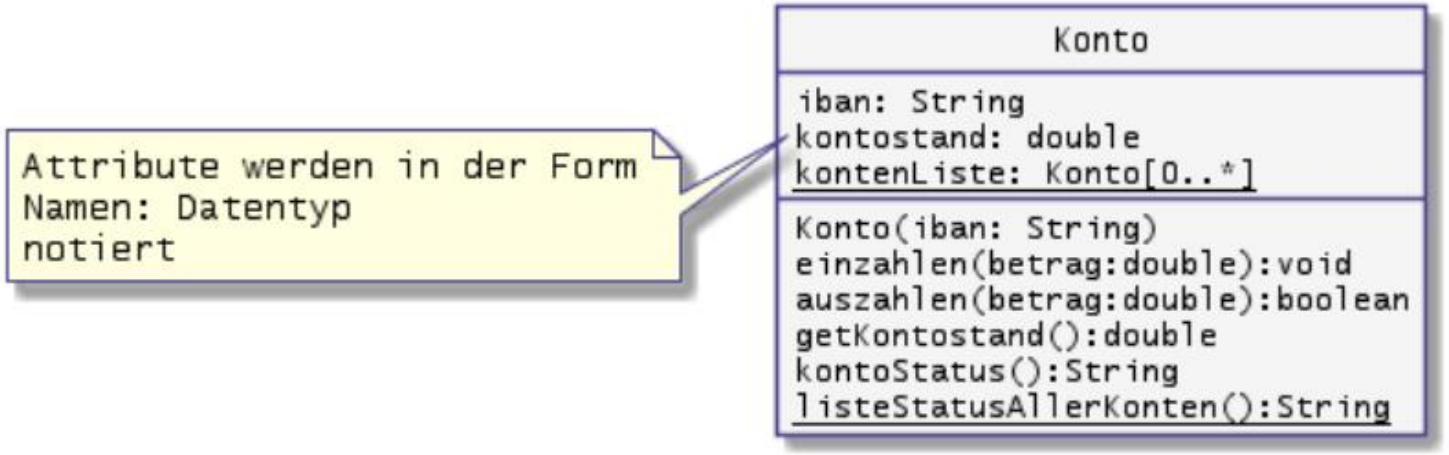
Einführung in UML

UML

- Unified Modeling Language
- Strukturierte Darstellung von Abläufen und Strukturen
- Verschiedene Diagrammtypen u.a.:
  - **Klassendiagramme**
  - Sequenzdiagramme
  - Ablaufdiagramme
  - Anwendungsfall-Diagramme

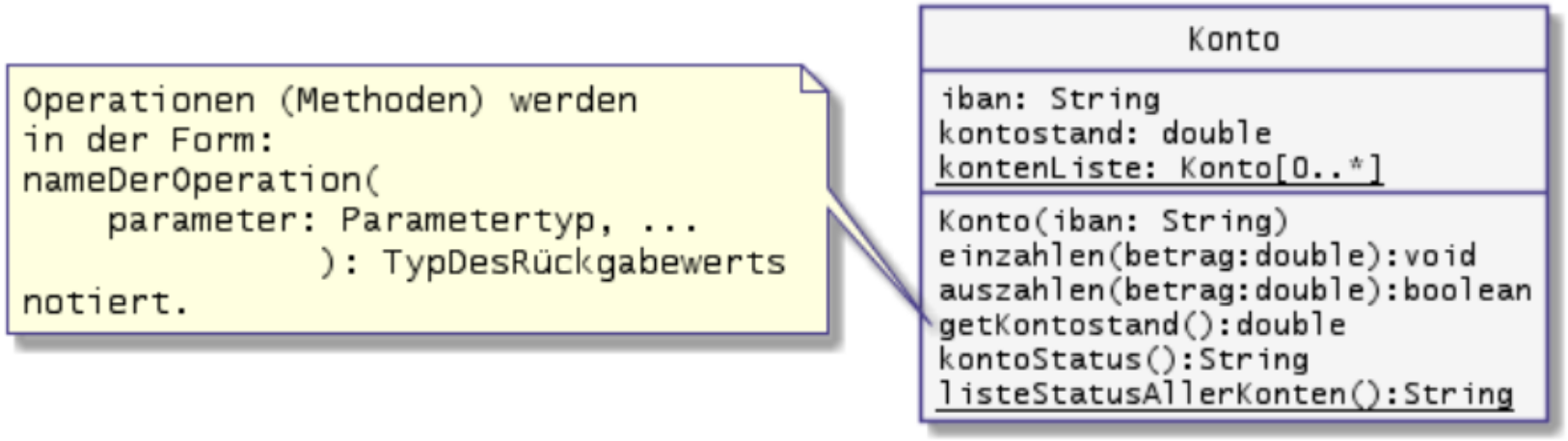
UML-Klassendiagramme

Klasse



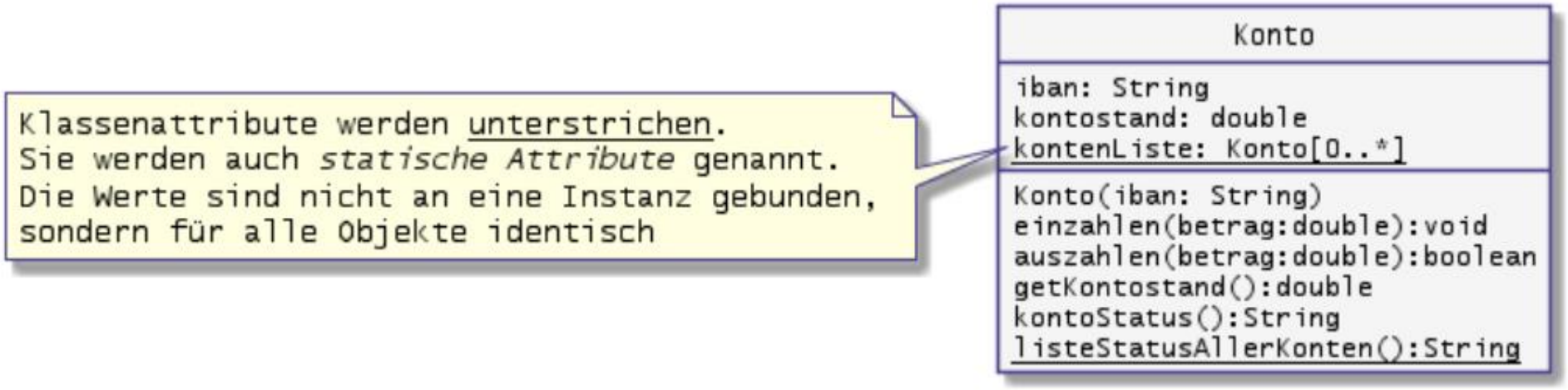
UML-Klassendiagramme

Methoden in Klassen:



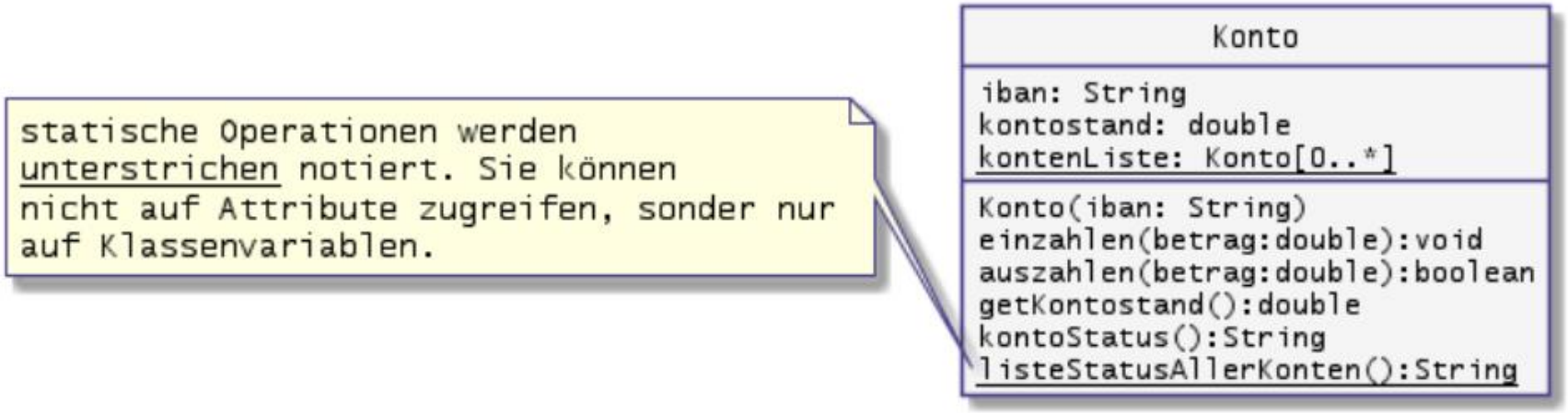
UML-Klassendiagramme

Statische Attribute:



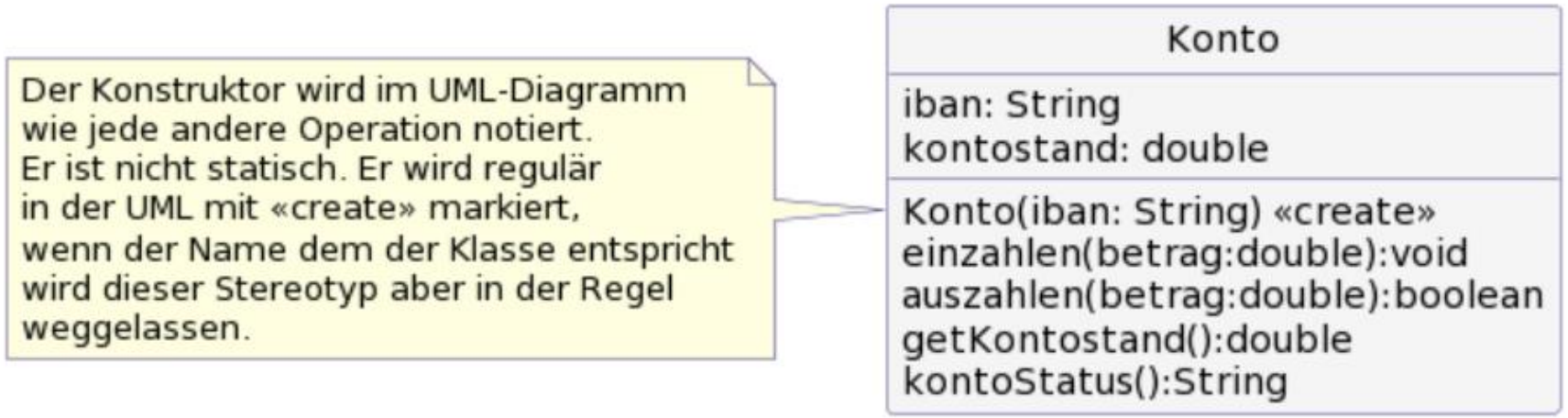
UML-Klassendiagramme

Statische Methoden:



UML-Klassendiagramme

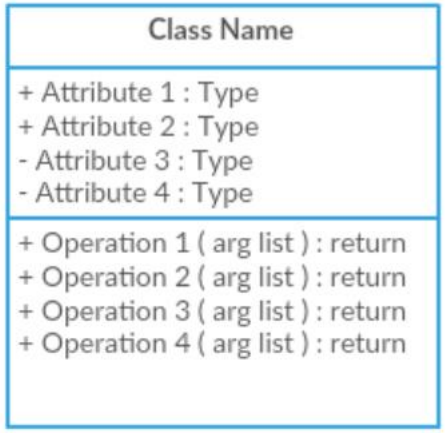
Konstruktor:



UML-Klassendiagramme

Sichtbarkeiten

- public: +
- private: -
- protected: #

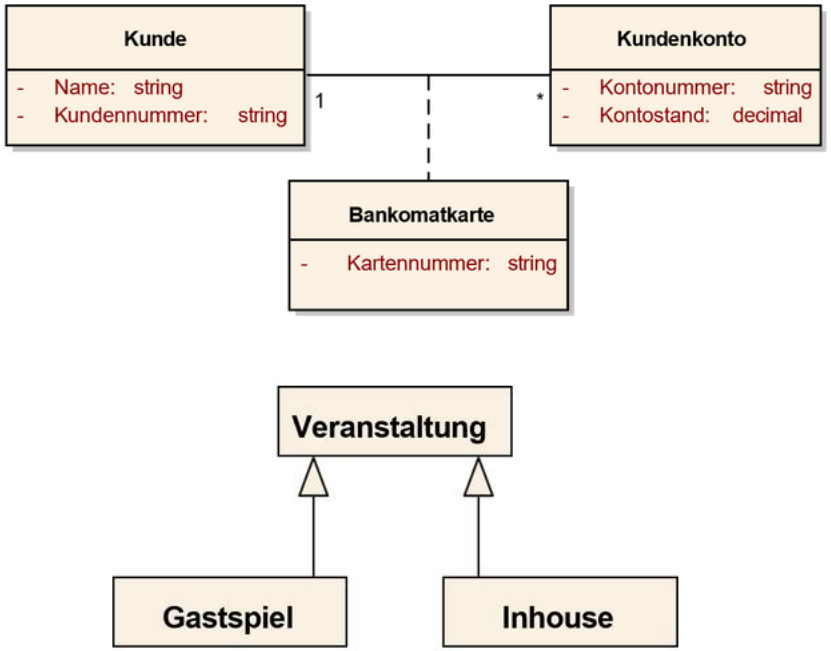


UML-Klassendiagramme

Verbindungen zwischen Klassen herstellen

u.a.

- **Assoziation: Beziehung** zwischen zwei oder mehr Klassen dar, bei der Objekte der einen Klasse mit Objekten der anderen Klasse interagieren können. Diese Beziehung kann in beide Richtungen gehen und ist normalerweise in Form einer Linie zwischen den Klassen im Diagramm dargestellt.
- **Vererbung:** Beschreibt eine **"ist-ein"**-Beziehung, bei der eine Klasse (die **Subklasse**) die Eigenschaften und Methoden einer anderen Klasse (der **Superklasse**) übernimmt.



Tools zur Modellierung

- Umlet (https://www.umlet.com/)
- draw.io (https://app.diagrams.net/)
- …

Quellen

https://www.programmierenlernenhq.de/methoden-in-java-was-sind-methoden-und-wie-werden-sie-verwendet/