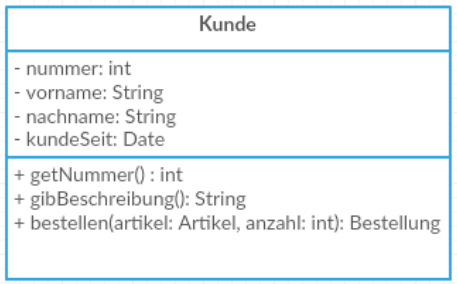
Lernziele

[LZ 1.1] Das Objektorientierte Programmiermodell erklären können.
[LZ 1.2] Einfache UML-Klassenmodelle verstehen können
[LZ 1.3] Java-Klassen definieren und nutzen können
[LZ 1.4] Möglichkeiten der Objekterzeugung kennen und anwenden können
[LZ 1.5] Wissen, was Methodenüberladung bedeutet und wann Sie sinnvoll ist
[LZ 1.6] Wissen, was Referenzen sind und wie sie verwendet werden
[LZ 1.7] Die this-Referenz kennen und ihre Einsatzmöglichkeiten beherrschen
[LZ 1.8] Den Java-Parameterübergabemechanismus kennen und einsetzen können
[LZ 1.9] Die Funktionsweise des Garbage Collectors in Java erläutern können
[LZ 1.10] Die Sichtbarkeitsstufen kennen und deren Anwendung beherrschen
[LZ 1.11] Die Array-Syntax kennen und Java-Arrays einsetzen können
[LZ 1.12] Die Java-String API kennen und diese in Java-Programmen einsetzen können
[LZ 1.13] Wissen, wie Java-Objekte inhaltlich verglichen werden können
[LZ 1.14] Motivation und Syntax statischer Elemente kennen und anwenden
[LZ 1.15] Die Konvention für Getter und Setter kennen und diese anwenden können
[LZ 1.16] Das Java package-Konzept kennen und es anwenden können

[LZ 1.1] Erläutern Sie das Objektorientierte Programmiermodell. Erklären Sie die Begriffe Objekt, Klasse, Attribut und Methode. Wie startet man ein OO-Programm und was geschieht nach dem Start typischerweise?

[LZ 1.2a] Erläutern sie folgendes Klassenmodell:



[LZ 1.2b] Entwerfen Sie ein UML-Klassenmodell zu folgenden Aussagen:

- Obst hat eine Bezeichnung (z.B. „Apfel“), eine Farbe (z.B. „rot“) und ein Gewicht in Gramm (z.B. 100)
- die Eigenschaften (s.o.) können über Methoden der Art get<Eigenschaft> gelesen werden
- Jedes Obst-Objekt kann eine Beschreibung als String liefern

[LZ 1.2c] Stellen Sie folgende Obst-Objekte grafisch dar

- 150gr. schwerer grüner Apfel
- hellgrüne Birne, Gewicht: 120 gr.
- gelbe Banane, 200 gr.
- rote Kirsche, 2 gr.

[LZ 1.3] Java-Klassen definieren und nutzen können

Definieren Sie eine Klasse Person

- eine Person hat ein Alter, z.B. 29
- eine Person hat einen Namen, z.B. „Amelie Müller“
- eine Person wird bei Erzeugung initialisiert über die Parameter für Name und Alter
- eine Person kann sich auf den Bildschirm ausgeben (gleichnamige Methode!), z.B. „Name: Amelie Müller, Alter: 29“

Ergänzen Sie folgendes Hauptprogramm zum Testen

```
static public void main(String[] args) {
    Person peter = new Person("Peter", 20);
    Person amelie = new Person("Amelie", 29);

    peter.ausgeben();
    amelie.ausgeben();
}
```

Definieren Sie eine Java-Klasse zur Beschreibung eines Tintenfüllers.

- Ein Füller hat ein Attribut „füllstand“, das initial den Wert 1.0 erhält (= voll).
- Ein Füller bietet folgende Operationen an
  - schreiben: Gibt auf die Console „schreibe etwas…“ aus und reduziert den Füllstand um 0.1, falls dies der aktuelle Füllstand zulässt; ist der Füller leer, wird nichts ausgegeben
  - auffuellen: füllt den Füller auf, falls er leer ist
  - istLeer: liefert true, falls der Füller leer ist

Definieren Sie eine **main-Methode**, in der sie ihre Füller-Klasse testen, d.h.

- ein Füller-Objekt erzeugen
- die Füllerdaten ausgeben (Hinweis: toString()-Methode nutzen!)
- mit dem Füller schreiben, bis er leer ist und ihn dann wieder auffüllen und abschließend die Füllerdaten ausgeben

- Erläutern Sie die Objekterzeugung und Konstruktoren (auch Konstruktorenverkettung) in Java anhand eines Beispiels
- Gegeben sei folgende Klasse:

```
class Y {
      int x;
      boolean y;
}
```

  Wie können Y-Objekte erzeugt werden und welcher Konstruktor wird verwendet?
- Geben Sie für die Klasse Y einen Konstruktor an, der beide Attribute über geeignete Parameter initialisiert. Definieren Sie nun einen weiteren, parameterlosen Konstruktor, der den ersten Konstruktor nutzt, um die Attribute mit Default-Werten ihrer Wahl zu belegen.

[LZ 1.5] Wissen, was Methodenüberladung bedeutet und wann Sie sinnvoll ist
- Was bedeutet Methodenüberladung?
- Wann setzt man Methodenüberladung ein?

[LZ 1.6] Wissen, was Referenzen sind und wie sie verwendet werden
- Wozu benötigt man Referenzen in Java?
- Wie weist man Referenzen zu?
- Wie vergleicht man Referenzen?
- Können Referenzen verändert werden wie etwa in „C“?

[LZ 1.7] Die this-Referenz kennen und ihre Einsatzmöglichkeiten beherrschen
- Was ist die „this“-Referenz?
- Wozu wird Sie verwendet?

[LZ 1.8] Den Java-Parameterübergabemechanismus kennen und einsetzen können

- Wie werden Parameter in Java übergeben? Gibt es Unterschiede in der Übergabe zwischen Simple Types, Objektreferenzen und Arrays?
- [*] Geben Sie den Methodenprototypen zu folgendem Aufruf von erzeugeKampagne an:

```
String[] details = new String[] { "detail 1", "detail 2"};
Auftrag a1 = new Auftrag("Businessplan-Erstellung");
Firma f1 = new Firma("Startup One");
String ergebnis = erzeugeKampagne(a1, f1, 2017, details);
```

- Sie möchten analog zu „C“ einen int-Wert als Ergebnisparameter zurückgeben (nicht über das Methodenergebnis!). Ein Beispiel wäre die „C“-Methode

```
void add(int x, int y, int *summe) {
    *summe = x + y;
}
```

Der Aufruf add(3, 5, &erg); bewirkt, dass die Variable erg den Wert 8 erhält.
Geben Sie eine Java-Lösung für obige add-Methode an.

[LZ 1.9] Die Funktionsweise des Garbage Collectors in Java erläutern können

- Welche Aufgabe hat der Garbage Collector in Java?
- Wie erfüllt der Garbage Collector seine Aufgabe?
- Wann kann welches Rechteck-Objekt aufgelöst werden?

```
{
    Rechteck r1 = new Rechteck(2, 4), r2, r3;
    r3 = new Rechteck(1, 3);
    r2 = r3;
    r3 = null;
    r1 = null;
    r2 = null;
}
```

[LZ 1.10] Die Sichtbarkeitsstufen kennen und deren Anwendung beherrschen

- Welche Sichtbarkeitsstufen gibt es in Java und wozu gibt es sie?
- Gegeben sei folgende Klasse – ergänzen Sie die Sichtbarkeiten!

```
class Film
    String titel;
    String nameRegisseur;
    Schauspieler[] mitwirkende;

    static void main(String[] args) {
        Film starWars7 = new Film();
    }

    String toString() {
        return "Titel: " + titel +
            ", Regie: " + nameRegisseur;
    }
}
```

[LZ 1.11] Die Array-Syntax kennen und Java-Arrays einsetzen können

- Kreuzen Sie die korrekten Anweisungen an!

  [ ] int intFeld[]; [ ] int[] intFeld2; [ ] char f = new char[];

  [ ] Quadrat[] quadrate; [ ] Quadrat quadrate[4]; [ ] char[] f2 = new char[3];

- [*] Definieren Sie ein char-Feld und belegen Sie es mit den Großbuchtaben von ‚A' bis ‚Z'. Berechnen Sie anschließend die Summe der ASCII-Werte der Buchstaben und geben Sie die Buchstaben und die Summe aus.
- Definieren und belegen Sie eine 2x2 int-Matrix und geben Sie deren Werte sowie die (berechnete!) Summe der Werte aus.

[LZ 1.12] Die Java-String API kennen und diese in Java-Programmen einsetzen können

- Nennen Sie wesentliche String-Methoden und beschreiben Sie deren Funktionsweise anhand der Parameter.
- Welche Bedeutung hat die Methode „toString()“? Bietet jedes Objekt diese Methode?
- [*] Definieren Sie ein Feld von Strings und belegen Sie es mit den Werten „Java Strings“, „ist sind“, „cool toll“. Geben Sie anschließend nur die erste Hälfte jedes Strings aus, danach jeweils die 2. Hälfte jedes Strings und abschließend nur die Strings aus, die das Wort „cool“ enthalten.

[LZ 1.13] Wissen, wie Java-Objekte inhaltlich verglichen werden können

- Wie werden Objekte bezüglich ihrer Referenzen verglichen? Wie werden Objekte bezüglich ihres Inhalts verglichen? Nutzen Sie das folgende Beispiel, um beide Vergleichsarten zu zeigen.

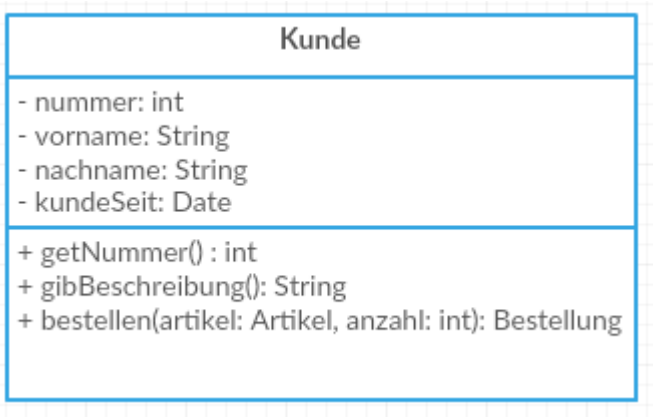
```
public void vergleicher(Rechteck r1, Rechteck r2) {
    if (                ) {
        System.out.println("Die Referenzen sind gleich!");
    }
    if (                ) {
        System.out.println("Die Rechtecke sind gleich!");
    }
}
```

- [*] Definieren Sie eine Rechteck-Klasse, die einen Vergleich von Rechteck-Objekten anhand ihrer Fläche erlaubt.

- Warum gibt es statische Methoden bzw. Atttribute? Wie werden diese genutzt (am Beispiel)?
- [*] Definieren Sie ein statisches Attribut „zaehler“ zu einer Klasse „X“ und speichern Sie darin die Anzahl der Objekte der Klasse X. Über einen Getter (getZaehler()) soll der Zähler auslesbar sein (z.B. für eine Ausgabe)

[LZ 1.15] Die Konvention für Getter und Setter kennen und diese anwenden können

- Erläutern Sie die Konvention für Getter und Setter anhand eines Beispiels.
- Definieren Sie die fehlenden Getter und Setter für folgende Klasse:



1. Klassen und Objekte

[LZ 1.16] Das Java package-Konzept kennen und es anwenden können

- Was ist ein „package“ in Java und warum verwendet man packages?
- Die Klasse „Date“ ist im package „java.util“ definiert. Wie sieht eine Anweisung zur Erzeugung eines Date-Objekts (Parameterloser Konstruktor!) aus?
- Geben Sie eine passende Import-Anweisung für obigen Fall an. Wie ändert sich der Konstruktor-Aufruf?

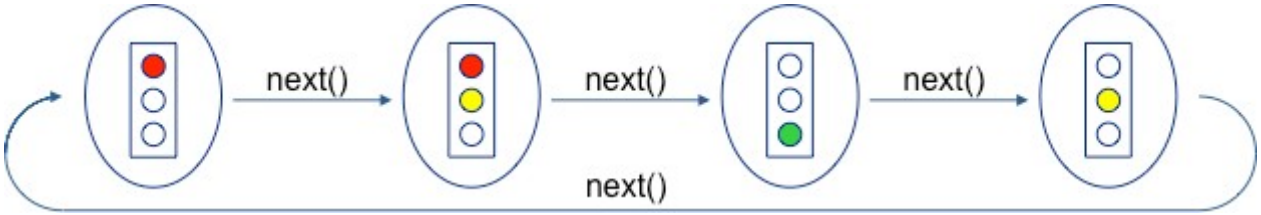
Rechner für Grundrechenarten

- Implementieren Sie eine Klasse „Rechner“ für die 4 Grundrechenarten basierend auf ganzen Zahlen.
- Die Grundrechenarten sollen über entsprechende Methoden von außen nutzbar sein.
- Testen Sie die Klasse in Rechner.main(), etwa so:

```
static public void main(String[] args) {
    Rechner r = new Rechner();
    System.out.printf("%d + %d = %d", 3, 4, r.addiere(3, 4));
    System.out.printf("%d - %d = %d", 3, 4, r.subtrahiere(3, 4));
    System.out.printf("%d * %d = %d", 3, 4, r.multipliziere(3, 4));
    System.out.printf("%d / %d = %d", 30, 4, r.dividiere(30, 4));
}
```

Übungsaufgabe Strassenampel

- Eine Verkehrsampel soll in einer Klasse Ampel definiert werden. Eine Ampel hat zunächst zwei Attribute:
  - Eine Ampel hat eine bestimmte Phase; als Phasen sind möglich: "rot", "rot/gelb", "grün" und "gelb„
  - Eine Ampel hat eine Bezeichnung, nämlich den Straßennamen, an dem sie aufgestellt ist
- Der Wechsel von einer Phase zur nächsten soll durch Aufruf einer Methode next vollzogen werden:



(1) Schreiben Sie eine Klasse Ampel, deren Konstruktor ein Straßenname übergeben wird. Der Konstruktor soll die Ampel mit der Rot-Phase initialisieren. Schreiben Sie eine Methode gibAmpelstatus, die den Straßennamen und die Ampelphase als String zurückgibt. Testen Sie Ihre Klasse mit folgender main-Methode:

```
public static void main(String[] args) {
    Ampel ampel1 = new Ampel("Goethestraße");
    System.out.println(ampel1.gibAmpelstatus());
}
```

Dieser Code sollte folgende Ausgabe erzeugen: **Goethestraße: rot**

(2) Fügen Sie der Klasse Ampel eine Methode next() hinzu, welche die Ampelphase weiterschaltet. Testen Sie Ihre Klasse mit folgender main- Methode:

```
public static void main(String[] args) {
    Ampel ampel1 = new Ampel("Goethestraße");

    for (int i = 0; i < 4; i++) {
        System.out.println(ampel1.gibAmpelstatus());
        ampel1.next();
    }
}
```

Dieser Code sollte folgende Ausgabe erzeugen:

```
Goethestraße: rot
Goethestraße: rot/gelb
Goethestraße: grün
Goethestraße: gelb
```

Übungsaufgabe Strassenampel

(3) Fügen Sie der Klasse Ampel einen weiteren Konstruktor hinzu, dem als Parameter neben dem Straßennamen eine zweite Ampel übergeben wird. Die Ampelphase soll so initialisiert werden, dass sie gerade das Gegenteil der als Parameter übergebenen Ampelphase darstellt. Ist z. B. die übergebene Ampel rot, dann soll die neue Ampel grün sein. Testen Sie Ihre Klasse mit folgender main-Methode:

```
public static void main(String[] args) {
      Ampel ampel1 = new Ampel("Goethestraße"),
      Ampel ampel2 = new Ampel("Schillerstraße", ampel1);

      for (int i = 0; i < 4; i++) {
         System.out.println(ampel1.gibAmpelstatus() + "\t\t" +
            ampel2.gibAmpelstatus());
         ampel1.next();
         ampel2.next();
      }
}
```

Dieser Code sollte folgende Ausgabe erzeugen:

```
Goethestraße: rot          Schillerstraße: grün
Goethestraße: rot/gelb     Schillerstraße: gelb
Goethestraße: grün         Schillerstraße: rot
Goethestraße: gelb         Schillerstraße: rot/gelb
```

Rationale Zahlen (**s. http://de.wikipedia.org/wiki/Rationale_Zahl**) sind Zahlen, die sich als Bruch p / q einer ganzen Zahl p und einer natürlichen Zahl q darstellen lassen (mit q != 0).

- Entwerfen und implementieren Sie eine Klasse RationaleZahl, deren Objekte rationale Zahlen repräsentieren, die sich addieren und multiplizieren lassen.
- Zähler und Nenner der vollständig *gekürzten* rationalen Zahl sollen zurückgegeben werden können.
- Ebenso soll der double-Wert der Zahl zurückgegeben werden können.
- Einen Bruch können Sie vollständig kürzen, in dem Sie Zähler und Nenner durch den größten gemeinsamen Teiler des Zählers und Nenners dividieren. Hinweis: *x,y ganze Zahlen* $\wedge$ $x > y > 0 \Rightarrow ggT(x,y) = ggT(y, mod(x, y))$
- Achten Sie darauf, dass die Null eine eindeutige Darstellung hat. Ebenso darf nicht durch Null geteilt werden.
- Die Klasse sollte einen Konstruktor enthalten, der Zähler und Nenner als int-Werte übergeben bekommt.

Weitere Aufgaben finden Sie hier:
http://www.home.hs-karlsruhe.de/~pach0003/informatik_1/aufgaben/objekt_orientierung.html

Übungsaufgabe Polynome

- Entwerfen Sie eine Klasse Polynom (s. http://de.wikipedia.org/wiki/Polynom), die ein reelwertiges (double) Polynom repräsentiert.
- Die Koeffzienten des Polynoms sollten als Feld im Konstruktur übergeben werden.
- Zwei Polynome sollen addiert werden können.
- Der Grad eines Polynoms soll sich bestimmt lassen.
- Ebenso soll die erste Ableitung eines Polynoms berechnet werden können mit der Formel

$$a_1 + 2a_2x + 3a_3x^2 + \ldots + na_nx^{n-1} = \sum_{i=1}^{n} ia_ix^{i-1}.$$

- Achten Sie bei den Tests darauf auch Randbedingungen wie Null-Polynom oder Polynome mit 0-Koeffzienten beim höchsten Exponenten zu berücksichtigen.
- Zu guter Letzt soll der Funktionswert an einer Stelle des Polynoms berechnet und zurückgegeben werden können.