Einführung in die objektorientierte Programmierung

Objekte & Klassen

J. Kleimann, H.-W. Sehring, D. Versick, F. Zimmermann

Studiengang Wirtschaftsinformatik

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

Vorlesungsinhalt

1. Organisatorisches
2. Die Programmiersprache Java
3. Erste Konzepte der Objektorientierung
4. Objekte & Klassen in Java: Figuren-Demo

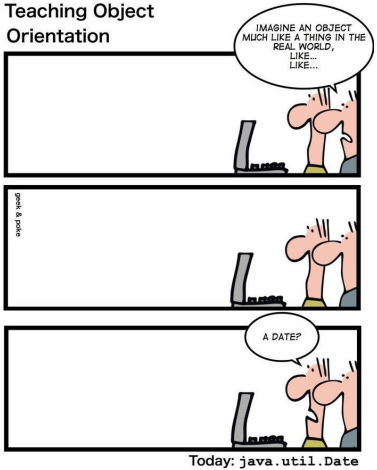


Abbildung: CC-BY-3.0 Oliver Widder

Lernziele

Wenn Sie die Veranstaltung erfolgreich absolviert haben,

- kennen Sie die Mechanismen der objektorientierten Programmierung und können diese in vorgegebenen Beispielen anwenden.
- kennen Sie grundlegende Prinzipien, die bei der Programmierung angewendet werden und können die Einhaltung dieser Prinzipien bewerten.
- können Sie ein kleines Projekt in Java selbstständig erstellen und debuggen.
- können Sie grundlegende Verfahren der Arbeit im Team anwenden.

Ablauf der Veranstaltung

- In jeder Woche haben Sie 2 Unterrichtstunden Vorlesung
- In jeder Woche haben Sie eine Übungsveranstaltung (3 Stunden)
- Die Übung findet in halber Zenturienstärke statt. Sie werden in zwei Gruppen eingeteilt werden.

Portfolioprüfung

Achtung, unbedingt lesen!

- Prüfung erfolgt vorlesungsbegleitend.
- Drei Prüfungsbestandteile müssen absolviert werden.
  - Moodle Tests (Gewicht 15% entspricht 15 Prüfungspunkten)
  - Ein Midterm Test (Gewicht 30% entspricht 30 Prüfungspunkten)
  - Eine Abschlussprüfung (Gewicht 55% entspricht 55 Prüfungspunkten)
- Die Prüfungspunkte der Teilprüfungen werden addiert.
- Zum Bestehen der Gesamtprüfung sind mindestens 50 Prüfungspunkte erforderlich.
- Im Falle eines festgestellten Täuschungsversuchs in einem der Prüfungsteile gilt die gesamte Prüfung als nicht bestanden.

Moodle Tests

- Begleitend zur Vorlesung/Übung machen Sie Tests in Moodle.
- Jeder Test hat eine Punktzahl. Ihr Ergebnis ergibt sich als Verhältnis ihrer in allen Tests zusammen erreichten Punktzahl und der maximal erreichbaren Punktzahl.
- Die Tests machen 15% der Note aus.
- Die Tests sollen die Diskussion der Studierenden über den Vorlesungsstoff fördern. Machen Sie deshalb die Tests möglichst **gemeinsam**, aber in wechselnder Zusammensetzung.
- Jeder muss seine Abgabe selbst in Moodle einreichen. Die Abgabe zählt nur für ihn allein.
- Jeder Test hat eine Bearbeitungszeit. Sie können die Tests innerhalb der Bearbeitungszeit (ca. 14 Tage) mehrfach durchführen und erhalten nach jedem Versuch ein automatisiertes Feedback (richtig/falsch). Es zählt die erreichte Punktzahl des letzten abgegebenen Versuchs.
- Nicht oder nicht rechtzeitig abgegebene Tests werden nicht gewertet.
- Mit Teilnahme an einem Moodle Test gelten Sie automatisch für die Portfolio-Prüfung angemeldet.

Midterm Test

- Midterm Test in Woche 5 macht 30% Ihrer Note aus.
- Art des Artefakts: Schriftliche Prüfung mit Multiple Choice und Programmieraufgaben.
- Die Testaufgaben orientieren sich zum Teil an den Moodle Tests.
- Die Testdauer beträgt 30 min und findet in der 5. Vorlesungswoche statt. Es ist eine Einzelprüfung. Der genaue Termin wird zeitnah bekannt gegeben.
- Wenn Sie am Prüfungstermin krank sind, können Sie in Woche 6 einen Ausweichtermin wahrnehmen.

Abschlussprüfung

- Die Abschlussprüfung ist eine schriftliche Prüfung von 60 Minuten Dauer und macht 55% Ihrer Note aus.
- Prüfungsart: schriftliche Prüfung.
- Es ist eine Einzelprüfung. Dabei gelten die während „normaler“ Klausuren üblichen Regeln.
- Wenn Sie am Prüfungstermin krank sind, können Sie in Woche 1 der Vorlesungszeit des Folgesemesters einen Ausweichtermin wahrnehmen. Hier gelten die Regelungen der Prüfungsordnung.
- Im Falle des Nicht-Bestehens der Portfolioprüfung, wird eine Wiederholungsprüfung in der regulären Nachprüfungsphase des nächsten Semesters angeboten. Auch diese ist wiederum eine Portfolioprüfung.

Pair Programming

- Kommunikation als Erfolgsfaktor (Wirtschafts-)Informatik
- Werte agiler Softwareentwicklung
  - Fachexperten und Entwickler müssen während des Projektes täglich zusammenarbeiten.
  - Die effizienteste und effektivste Methode, Informationen an und innerhalb eines Entwicklungsteams zu übermitteln, ist im Gespräch von Angesicht zu Angesicht.
  - In regelmäßigen Abständen reflektiert das Team, wie es effektiver werden kann und passt sein Verhalten entsprechend an.
- Know-How Verteilung auf mehrere Köpfe (Truck Factor!)
- Die praktischen Übungen werden als wechselnde Paare durchgeführt. Die Gruppeneinteilung während der Online-Veranstaltungen erfolgt zufällig durch Teams- oder Zoom-Breakout-Rooms.

Vorlesungsinhalte

- Objekt- und Klassenbegriff
- Elemente einer Klassendefinition
- Objektinteraktion
- Ordnung schaffen in der Objektwelt
- Softwarebibliotheken
- Grundlagen des Klassenentwurfs
- *Testen, Fehleranalyse, Debugging*
- Schnittstellenvererbung, Polymorphie
- Vererbung
- Abstrakte Klassen und späte Bindung
- *Fehlerbehandlung*
- Entwurf von Anwendungen

Je nach Vorlesungsfortschritt und Vorkenntnissen innerhalb der Gruppen werden evtl. Themen weggelassen oder hinzugefügt.

Lernunterlagen

- Moodle Kurs: I166 -Einführung in die objektorientierte Programmierung
- Foliensätze der Dozenten
- Java lernen mit BlueJ
  Objects first - Eine Einführung in Java
  David J. Barnes - Michael Kölling
  6., aktualisierte Auflage
  Pearson ISBN 978-3-86894-911-7

- Anschaffung des Buches wird für Anfänger empfohlen, da die Vorlesung sich an diesem Buch orientiert.
- Ältere (und damit billigere) Auflagen sind nutzbar. Inhalte stimmen zu 95% überein, sind jedoch teilweise anders gegliedert.
- Exemplare dieser und älterer Auflagen sind in der Bibliothek vorhanden.
- Weitere E-Books zu dem Thema sind kostenlos in SpringerLink zu finden (s.a. Moodle Seite des Kurses)

BlueJ

Der Vorteil von BlueJ für Programmieranfänger ist, dass

- man sehr intuitiv Objekte erzeugen kann.
- man sehr intuitiv Objekte manipulieren kann.
- man ein natürliches Verständnis von Objekten aufbauen kann.
- man sich zunächst auf die Business Logik einer Anwendung konzentrieren kann, ohne durch Interaktionselemente abgelenkt zu werden.
- die Umgebung von BlueJ einfach gehalten ist, sodass Anfänger nicht verwirrt werden.

Es wird empfohlen, BlueJ auf dem eigenen Rechner zu installieren.
BlueJ ist keine Entwicklungsumgebung für professionelle Programmierer.
Folgeveranstaltungen werden mit anderen Entwicklungsumgebungen arbeiten (Eclipse oder IntelliJ).

Objekte und Klassen

Diese beiden Konzepte sind zentral für alles Folgende.

- **Klasse**
  Kategorie oder Schablone von Dingen.
  Als Kategorie zur Abstraktion von gleichartigen Objekten.
  Als Schablone ähnlich einer Kopier- oder Mustervorlage.
- **Objekt**
  Ein Ding unserer Anschauung oder unseres Denkens das zu einer Klasse gehört. Man sagt, das Objekt sei ein *Exemplar* oder (falsch übersetzter Anglizismus) eine Instanz der Klasse.
  Ein Objekt
  - hat eigene Eigenschaften (Zustand) und
  - unterscheidet sich von anderen Objekten (Identität) und
  - kann manipuliert werden (Verhalten).
  - **Eine Klasse, viele Exemplare**

Vom Programmieren lernen

Programmieren können bedeutet

- Programmiersprache „sprechen können“
  - → Vokabular kennen
  - → Grammatik beherrschen
- Herausforderungen analysieren und mittels Algorithmen & Datenstrukturen lösen können
- Werkzeuge beherrschen
- Bibliotheken kennen und gekonnt einsetzen
- Architekturelle Konzepte in der Sprache abbilden können

Unsere Aufgabe:

- Vorstellen der Vokabeln, Grammatik und der Konzepte
- Erste Übungsaufgaben bearbeiten

Ihre Aufgabe:

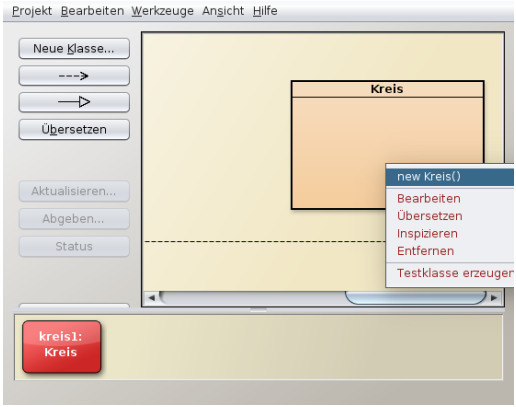
- Vokabeln lernen
- Grammatik lernen
- Sprechen üben

Der Lernerfolg hängt maßgeblich von Ihnen ab. Machen Sie zu wenig davon, werden Sie nicht genug gelernt haben!

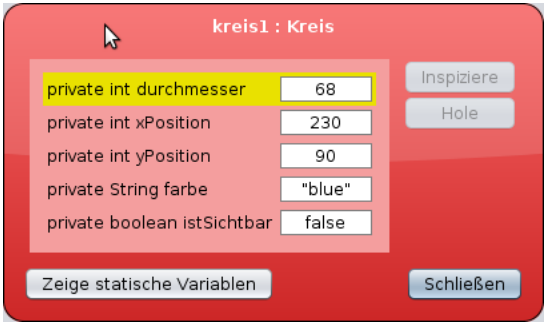
Ein zu geringer Lernerfolg wird zu großen Problemen in Folgelehrveranstaltungen führen!

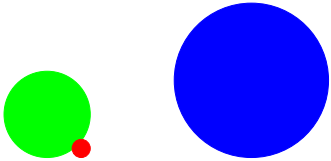
Erzeugen von Objekten

- Das Erzeugen von Objekten erledigt ein **Konstruktor**.
- *BlueJ stellt für den Aufruf von Konstruktoren eine grafische Oberfläche zur Verfügung. Jedes Objekt, das so erzeugt wurde, wird durch ein rotes Rechteck repräsentiert. Es wird mittels eines Bezeichners identifiziert, um unterschiedliche Objekte derselben Klasse unterscheiden zu können.*
- Erzeugung im Programm: `new Kreis();`



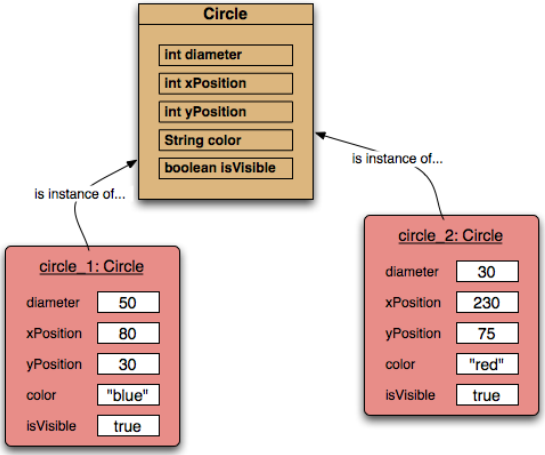
Objekte haben einen Zustand

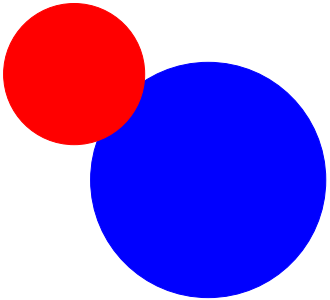




- Alles Kreise
- Jeder Kreis hat einen individuellen Zustand

Objekte haben eine Identität

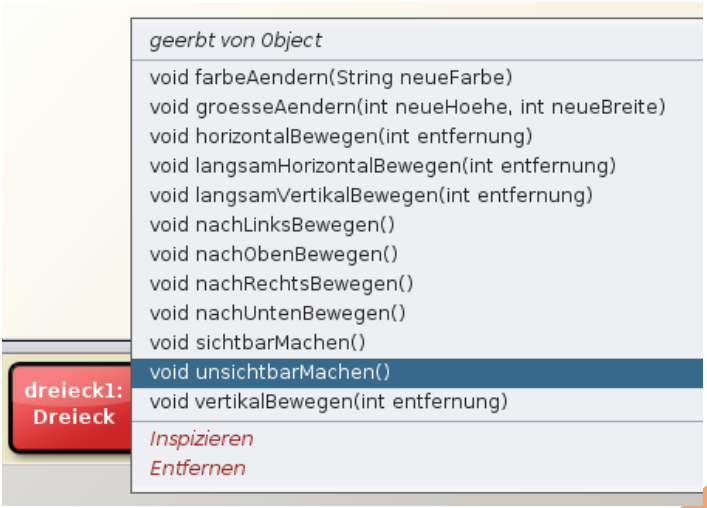




- Alles Kreise
- Alles unterscheidbare Objekte
- Jeder Kreis hat eine Identität

Objekte haben ein Verhalten

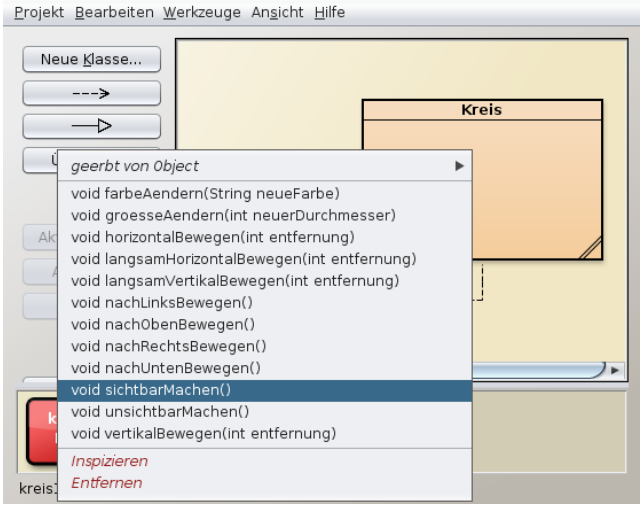
Das **Verhalten** eines Objekts wird durch die aufrufbaren Methoden beschrieben.



Aufrufen von Methoden

- Eine **Methode** kann auf einem Objekt ausgeführt werden.
- *BlueJ stellt auch dafür eine grafische Oberfläche zur Verfügung. Klickt man auf ein Objekt, dann kann eine auszuführende Methode auswählen.*

- **Methodenaufruf** im Programm:
  `kreis1.sichtbarMachen();`



Klassen

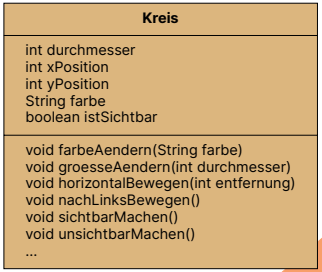
Klassen sind Schablonen, weil sie

- den **Zustandsraum**, also alle erlaubten Zustände, und
- das **Verhalten** (die Manipulationsmöglichkeiten)

ihrer Exemplare in ihrem Quellcode definieren.

- Klassen werden benutzt um Exemplare entsprechend der hinterlegten Schablone zu konstruieren.
- Dazu ruft man ihren Konstruktor auf, z.B. `new Kreis();`

zum Beispiel



Quellcode (Quelltext)

- Eine Klasse definiert den Zustandsraum und das Verhalten ihrer Exemplare in Form von
  - **Exemplarvariablen** (Attributen, Eigenschaften, Felder oder kurz Variablen)
  - und **Exemplarmethoden** (oder kurz Methoden)
- Exemplarvariablen und -methoden werden im **Quellcode** beschrieben.
  - Der Quellcode ist nach strikten Regeln aufgebaut.
  - Durch diese Regeln werden viele Programmierfehler schon vor der Laufzeit durch den **Compiler** erkennbar gemacht.
  - Diese Regeln machen die Programmiersprache Java aus. Die Regeln und ihre Interpretation sind in der **Java Spezifikation** festgelegt, die plattform- und herstellerunabhängig angibt, was Java ist und tut.

Compile+Run

- Java Programme sind einfache Texte. Damit Java-Programme ausgeführt werden können, müssen sie vom Java Compiler in ausführbare Form übersetzt werden.
- Dies erledigt das (Kommandozeilen-) Programm `javac`, *dessen Aufruf in BlueJ integriert ist*.
- Bei der Übersetzung überprüft der Compiler die Java Syntax und meldet Fehler, falls diese nicht eingehalten wird.
- der Compiler überprüft unter anderem auch Querverweise zwischen Klassen und Sicherheitsaspekte. Damit verhindert er den Missbrauch der Programmiersprache.
- Das Ergebnis des Compilers ist sog. Bytecode, der von der Java Virtual Maschine (JVM) ausgeführt werden kann. Der Start der JVM erfolgt über das (Kommandozeilen-) Programm `java`. *Auch die JVM ist in BlueJ integriert.*

Quellcode 1/4
Exemplarmethoden

Exemplarmethode

```
public int getXPosition() {
  return xPosition;
}
```

- Methoden haben eine **Signatur** (**`public int`** `getXPosition()`) und einen Block ({...}), der das Verhalten der Objekte der Klasse festlegt.

- Die Sichtbarkeit **`public`** sagt, dass diese Methode aus anderen Klassen heraus benutzt werden kann.
- Methoden können ein Ergebnis liefern, das einen Rückgabedatentyp (hier **`int`**) hat.
- Wenn Methoden ein **Ergebnis** liefern sollen, dann muss dieses Ergebnis in einem **`return`**-Statement im Block der Methode festgelegt werden.
- Methoden können **Parameter** haben. Die möglichen Parameter werden in einer Parameterliste angegeben. In dem Beispiel ist die Parameterliste leer.

Quellcode 2/4
Exemplarmethoden

Exemplarmethode

```
public void horizontalBewegen(int entfernung) {
  loeschen();
  xPosition += entfernung;
  zeichnen();
}
```

- Der Rückgabedatentyp **`void`** sagt, dass keine Ergebnisse von dieser Methode geliefert werden. Es ist dann kein **`return`** Statement erforderlich.
- Dieser Methode wird ein Parameter mit dem Namen `entfernung` vom **Typ** **`int`** mitgegeben. Er legt die Entfernung fest, die das Objekt bewegt werden soll.

Quellcode 3/4
Klassen

Der Quellcode der Klasse Kreis ist in der Datei **Kreis.java** enthalten. Es ist eine Textdatei, die mit jedem Texteditor bearbeitet werden kann. Es ist kein BlueJ oder Eclipse oder IntelliJ erforderlich.

Klassendefinition in Kreis.java

```
public class Kreis {
  //hier gehören Exemplarvariablen
       und -methoden hin.
}
```

- Das Schlüsselwort **`class`** leitet eine Klassendefinition ein.
- Die geschweiften Klammern **{...}** legen den Anfang und das Ende einer Klassendefinition fest. Sie sind der Anfang und das Ende eines **Blocks**.
- Das Schlüsselwort **`public`** legt fest, dass diese Klasse von anderen Klassen aus benutzt werden kann. Hierzu später mehr.
- `//` leitet einen Kommentar bis zum Ende der Zeile ein.

Quellcode 4/4

Exemplarvariablen

Exemplarvariablen

```
private int durchmesser;
private int xPosition;
private int yPosition;
private String farbe;
private boolean istSichtbar;
```

- Exemplarvariablen beschreiben den Zustandsraum, den die Objekte einer Klasse einnehmen können. Sie haben einen Namen, der Ihre Bedeutung beschreibt: `durchmesser`, `xPosition`, …
- Exemplarvariablen nehmen für unterschiedliche Objekte in der Regel unterschiedliche Werte an, müssen das aber nicht.
- Welche Werte angenommen werden können, wird durch den Datentyp der Variablen beschrieben, hier `int`, `String` oder `boolean`.
- Das Schlüsselwort **`private`** gibt an, dass diese Exemplarvariable nur innerhalb der Klasse benutzt oder verändert werden kann.

Übung

Bearbeiten Sie den **Test zu Foliensatz 1** aus dem Moodle Kurs. Geplante Dauer: ca. 20 min.

Fragen?

Für weitere Fragen im Nachgang können Sie mich gerne über Moodle oder via E-Mail kontaktieren!

Übung

Machen Sie sich dabei mit den Funktionen von BlueJ zur Verwaltung von Objekten mit Hilfe des Beispielprojektes Figuren vertraut:

- Erzeugen Sie einen Kreis.
- Erzeugen Sie dann ein Quadrat.
- Erzeugen Sie einen weiteren Kreis.

Fragen:

- Was unterscheidet die Objekte voneinander, was haben sie gemeinsam?
- Was müssen Sie tun, damit die Objekte angezeigt werden?
- Durch welche Eigenschaften kann hier ein Kreis bzw. ein Quadrat beschrieben werden?

Übung

Machen Sie sich mit dem Aufruf von Methoden und deren Auswirkung auf Objekte (inspizieren!) vertraut:

- Was geschieht, wenn Sie `nachUntenBewegen()` zweimal aufrufen? Oder dreimal?
- Was passiert, wenn Sie `unsichtbarMachen()` zweimal aufrufen?

Sorgen Sie dafür, dass Sie die unterschiedlichen Objekte auf der Leinwand unterscheiden können.

Übung

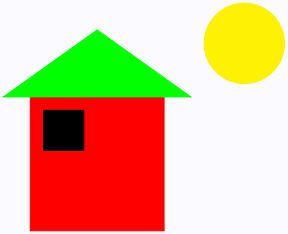
Beschreiben Sie die Struktur der Klasse Quadrat. Inspizieren Sie dazu eines der bereits erzeugten Objekte:

- Welche möglichen Zustandsausprägungen gibt es?
- Welches Verhalten ist definiert und welche Auswirkungen scheint es auf den Zustand zu haben?

Bitte lösen sie diese Aufgabe **ohne** in den Quelltext zu schauen!

Übung

Zeichnen Sie mittels der Ihnen bekannten Methodik ein Haus, in etwa so:



- Nachdem Sie wissen, wie es geht, notieren Sie die notwendigen Schritte, und setzen Sie die virtuelle Maschine zurück.
- Öffnen Sie die Konsole und lassen sie die Methodenaufrufe protokollieren.
- Wiederholen Sie die Übung und erklären Sie die Ausgabe.

Übung

Öffnen Sie das Projekt Haus und inspizieren Sie den Quellcode der Klasse Bild. Dieser sollte Ihnen bekannt vorkommen.

- Lassen Sie die Sonne in einer Animation untergehen.
- Ergänzen Sie einen Mond (blauer Kreis). Diskutieren Sie anschließend, ob es sinnvoll ist, ein neues Objekt zu erzeugen, oder die Sonne zu recyclen bzw. umzufärben.
- Ergänzen Sie eine Person in dem Bild - dabei ist es hilfreich, sich bzgl. der Ergänzungen von den anderen Elementen inspirieren zu lassen. Was hat sich in der Übersicht geändert? Was könnte das bedeuten?

Analysieren sie anschließend die Methode `horizontalBewegen(...)` aus der Klasse Kreis.

Konzepte: Zusammenfassung I

- **Klasse** Sind Baupläne/Schablonen für Objekte. Eine Klasse beschreibt den Zustandsraum und das Verhalten für gleichartige Objekte.
  Objekte nennt man auch Exemplare (Instanzen) einer Klasse.
- **Objekt** Objekte repräsentieren zusammengehörige Aspekte eines Anwendungsbereichs.
  Sie haben einen Zustand, ein Verhalten und eine Identität.
- **Eine Klasse, viele Exemplare** Von einer Klasse können viele gleichartige Exemplare erzeugt werden.
  Eine Klasse kennt ihre Objekte nicht, jedes Objekt aber seine Klasse.
  Ein Objekt kann seine Klasse niemals verändern (in der Programmiersprache Java).
- **Konstruktor** Konstruktoren initialisieren Objekte, und bringen sie in einen konsistenten Anfangszustand.
  Konstruktoren stellen kein Verhalten des Objekts zur Verfügung.
  Eine Klasse kann mehrere Konstruktoren für Ihre Objekte anbieten.
- **Zustand** Objekte haben einen Zustand. Dieser Zustand wird durch Werte repräsentiert, die in Exemplarvariablen (Feldern, ...) gehalten werden.

Konzepte: Zusammenfassung II

- **Verhalten** Das Verhalten eines Objekts wird durch die Methoden, die auf diesem Objekt aufgerufen werden können, beschrieben.
  Das Zusammenführen von Zustand und Verhalten in Objekten ist ein wichtiges Konzept der Objektorientierung.
- **Methode** Wir können mit Objekten kommunizieren, indem wir ihre Methoden aufrufen. Ein Objekt tut üblicherweise etwas, wenn eine seiner Methoden aufgerufen wird. Methoden definieren das Verhalten eines Objekts.
- **Methodenaufruf** Objekte können miteinander kommunizieren, indem sie gegenseitig ihre Methoden aufrufen.
- **Quellcode (Quelltext)** Der Quelltext einer Klasse legt die Struktur und das Verhalten (die Datenfelder und Methoden) aller Instanzen dieser Klasse fest. Durch die textuelle Repräsentation werden Anweisungen (z.B. Methodenaufrufe) der Reihe nach ausgeführt.
- **Signatur** Der Name und die Parametertypen einer Methode werden als deren Signatur bezeichnet. Sie benennt die benötigten Informationen für einen Aufruf der Methode.
- **Ergebnis** Methoden können Informationen über ein Objekt durch einen Ergebniswert zurückliefern.
- **Parameter** Methoden können Parameter haben, mit denen zusätzliche Informationen für eine Aufgabe angegeben werden.

Konzepte: Zusammenfassung III

- **Typ** Ein Typ definiert, welche Arten von Werten oder Objekten eine Variable (Parameter, lokale Variable oder Exemplarvariable) annehmen kann.

Klasse

Sind Baupläne/Schablonen für Objekte. Eine Klasse beschreibt den Zustandsraum und das Verhalten für gleichartige Objekte. Objekte nennt man auch Exemplare (Instanzen) einer Klasse.

Objekt

Objekte repräsentieren zusammengehörige Aspekte eines Anwendungsbereichs.
Sie haben einen Zustand, ein Verhalten und eine Identität.

Eine Klasse, viele Exemplare

Von einer Klasse können viele gleichartige Exemplare erzeugt werden.
Eine Klasse kennt ihre Objekte nicht, jedes Objekt aber seine Klasse.
Ein Objekt kann seine Klasse niemals verändern (in der Programmiersprache Java).

Konstruktor

Konstruktoren initialisieren Objekte, und bringen sie in einen konsistenten Anfangszustand.
Konstruktoren stellen kein Verhalten des Objekts zur Verfügung.
Eine Klasse kann mehrere Konstruktoren für Ihre Objekte anbieten.

Zustand

Objekte haben einen Zustand. Dieser Zustand wird durch Werte repräsentiert, die in Exemplarvariablen (Feldern, ...) gehalten werden.

Verhalten

Das Verhalten eines Objekts wird durch die Methoden, die auf diesem Objekt aufgerufen werden können, beschrieben. Das Zusammenführen von Zustand und Verhalten in Objekten ist ein wichtiges Konzept der Objektorientierung.

Methode

Wir können mit Objekten kommunizieren, indem wir ihre Methoden aufrufen. Ein Objekt tut üblicherweise etwas, wenn eine seiner Methoden aufgerufen wird. Methoden definieren das Verhalten eines Objekts.

Methodenaufruf

Objekte können miteinander kommunizieren, indem sie gegenseitig ihre Methoden aufrufen.

Quellcode (Quelltext)

Der Quelltext einer Klasse legt die Struktur und das Verhalten (die Datenfelder und Methoden) aller Instanzen dieser Klasse fest. Durch die textuelle Repräsentation werden Anweisungen (z.B. Methodenaufrufe) der Reihe nach ausgeführt.

Signatur

Der Name und die Parametertypen einer Methode werden als deren Signatur bezeichnet. Sie benennt die benötigten Informationen für einen Aufruf der Methode.

Ergebnis

Methoden können Informationen über ein Objekt durch einen Ergebniswert zurückliefern.

Parameter

Methoden können Parameter haben, mit denen zusätzliche Informationen für eine Aufgabe angegeben werden.

Typ

Ein Typ definiert, welche Arten von Werten oder Objekten eine Variable (Parameter, lokale Variable oder Exemplarvariable) annehmen kann.