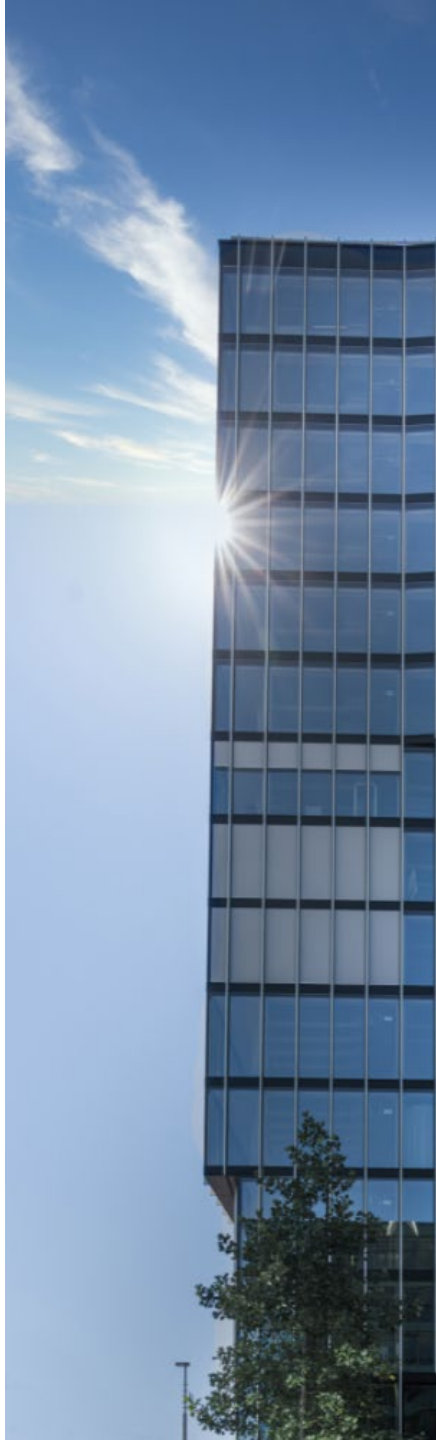
HSLU Hochschule Luzern

Objektorientierte Programmierung

Objektorientierung

Roland Gisler

FH Zentralschweiz



Inhalt

- Was ist eigentlich Objektorientierung?
- Objekte und Klassen
- Klassen: Zustand und Verhalten
- Quellcode in Java
- Was ist ein Programm?
- Kurze Live-Demo
- Übung
- Zusammenfassung

Lernziele

- Erstes, einfaches Verständnis von «Objektorientierung».
- Sie wissen was Klassen und Objekte sind.
- Sie können Objekte identifizieren und erste einfache Beispiele von Klassen entwerfen.
- Sie können zwischen Zustand und Verhalten unterscheiden.
- Sie verstehen, dass es unterschiedliche Abstraktionen gibt.
- Sie wissen was Quellcode (z.B. in Java) ist.
- Sie wissen was ein Programm ist.

Objektorientierung

Objektorientierung (OO)

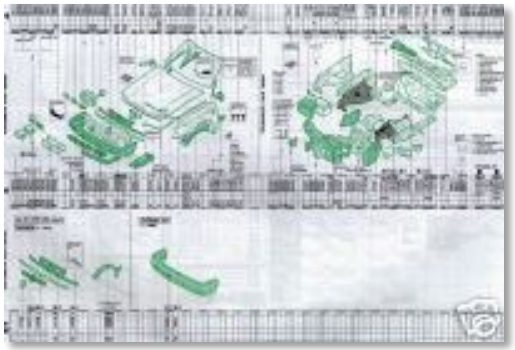
- Die Objektorientierung ist bereits Ende der 1960er entstanden.
- Im Unterschied zur prozeduralen Programmierung, welche Daten und Funktionen noch weitgehend trennt, wird dies in der Objektorientierung in Objekten zusammengefasst.
- Eine sehr frühe Sprache welche OO umsetzte war Smalltalk.
- Richtig populär wurde die OO ab ca. 1985 mit der Sprache C++.
- Java ist eine objektorientierte Sprache und existiert seit 1996.
- Zentrale Elemente in der Objektorientierung sind Objekte und häufig (aber nicht zwingend) Klassen.
- Mehr zu den verschiedenen Programmier-Paradigmen:
  Modul PCP (Programming Concepts & Paradigms)

Objekte und Klassen

Objekte und Klassen – in der Realität

- Objekte
  - Repräsentieren "Dinge" aus der realen Welt oder aus einem abstrakten Problembereich.
  - Beispiel 1 - Können konkret sein:
    "*Der rote Wagen da im Parkhaus.*"
  - Beispiel 2 - Können abstrakt sein:
    "*Das Telefongespräch mit Frau Müller am 24. August 2023.*"

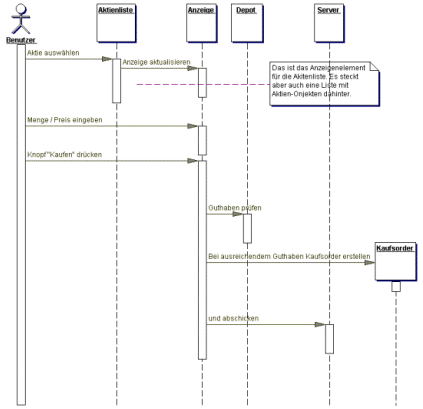
- Klassen
  - Sind quasi die Baupläne für Objekte.
    Beispiel 1: "Bauplan für einen Wagen."
    Beispiel 2: "Kontaktprotokoll(-formular)"



Objekte und Klassen - Interaktion

- In der realen Welt:
  Die realen Objekte interagieren untereinander bzw. miteinander.
  - Kommunikation (Schall, Funk, Licht etc.).
  - Die Objekte verändern ihren Zustand.
  - Die Objekte beeinflussen sich gegenseitig.

- In einem objektorientierten Programm:
  Die (Software-)Objekte interagieren!
  - Kommunikation mittels Methodenaufrufen.
  - Die Objekte verändern ihren Zustand.
  - Die Objekte beeinflussen sich gegenseitig.



UNIFIED MODELING LANGUAGE

Objekte und Klassen - Modellierung

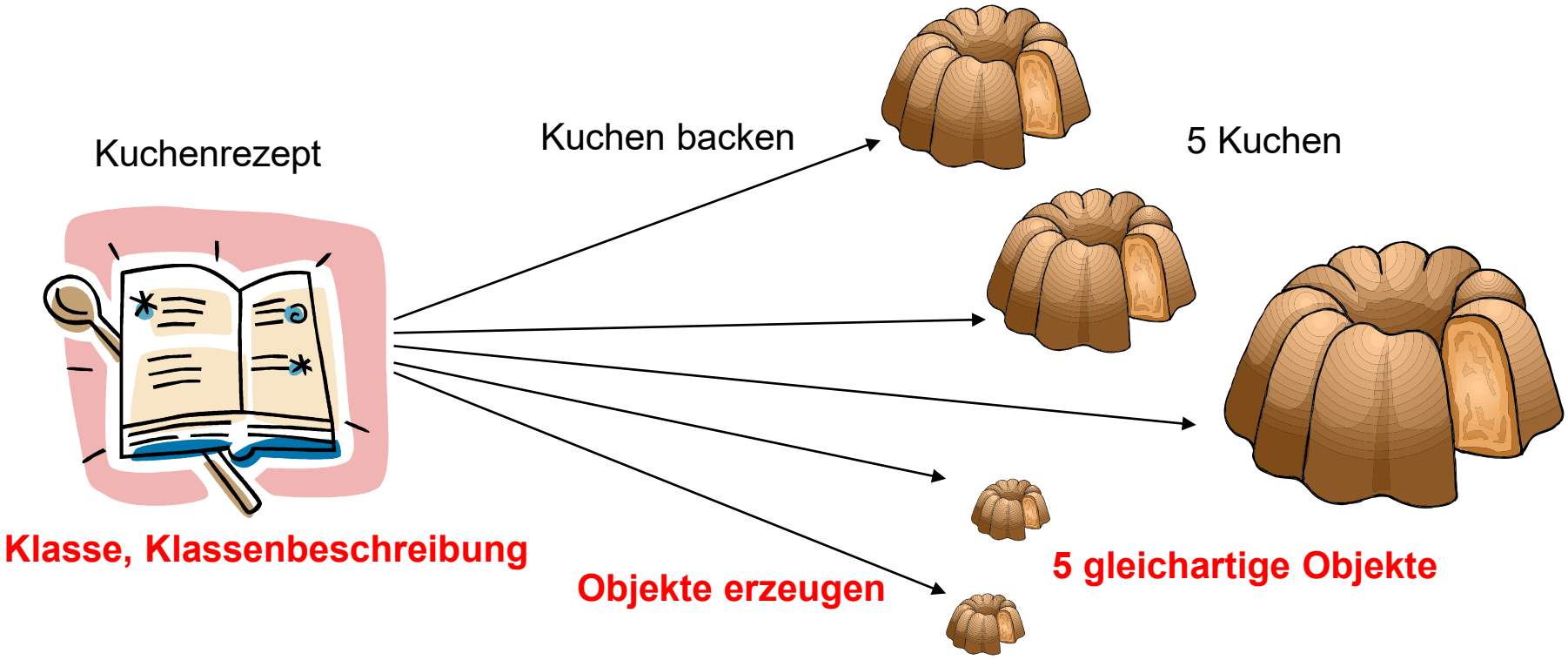
- Sowohl in der realen Welt als auch in der Welt der OOP hat man es mit Objekten zu tun.
- Reale Objekte werden im jeweiligen Kontext als Software-Objekte **modelliert** (→ vereinfacht, abstrahiert) und implementiert.
- Vorteile:
  - Man bewegt sich immer in einer Objekt-Welt.
  - Es existiert eine adäquate und durchgängige Betrachtungsweise.
- Nachteil:
  - Objekte sind immer eine (starke) Vereinfachung. Sie können manchmal (sehr) abstrakt werden.

Objekte und Klassen - Programmierung

- Software-Entwickler*innen programmieren **nicht** einzelne Software-Objekte.
- Stattdessen beschreibt/programmiert man gleichartige Objekte mit Hilfe von **Klassen**.
- Ausgehend von einer bestimmten Klasse lassen sich dann beliebig viele entsprechende **Objekte** erzeugen.

Objekte und Klassen - Beispiel

- Analogie: "Kuchen backen"
- Ausgehend von einem bestimmten Kuchenrezept (vgl. Klasse) lassen sich beliebig viele gleichartige Kuchen (vgl. Objekte) backen (vgl. erzeugen).



Objekte und Klassen - Zusammengefasst

- Fazit: Wir programmieren Klassen!
- Objekte erzeugt man ausgehend von Klassen,
  und zwar «erst» zur Laufzeit des Programmes.
- Zur Laufzeit hat man es also mit Objekten zu tun, die miteinander interagieren und so gemeinsam eine bestimmte Aufgabe lösen.

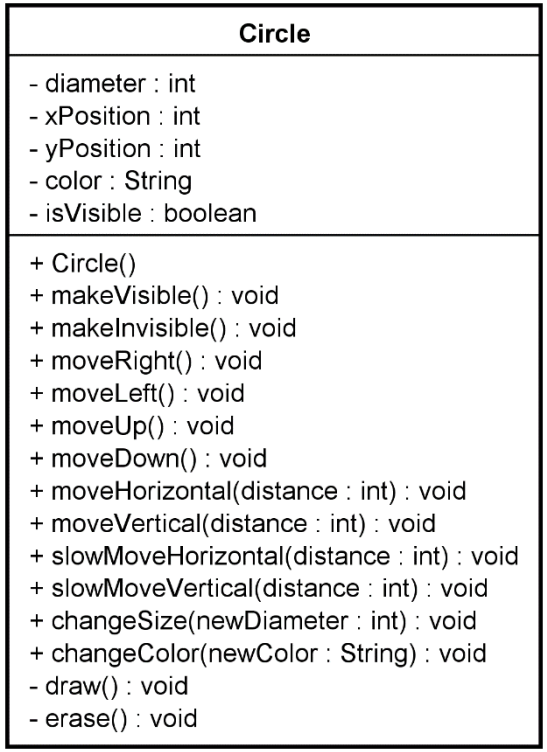
Klassen

Klassen

- Von einer Klasse können mehrere Instanzen, d.h. Objekte erzeugt werden.
- Ein Klasse definiert **Attribute**: Werte, die in Variablen gespeichert werden und den **aktuellen Zustand** des Objektes wiederspiegeln.
  - Die Klasse definiert, welche Attribute ein Objekt besitzt. Jedes Objekt hat seine eigene Wertemenge (Zustand des Objektes).
- Eine Klasse definiert **Methoden**: Definieren **das Verhalten** eines Objektes und werden (häufig) dessen Zustand verändern.
  - Die Klasse definiert, welche Methoden zur Verfügung stehen.
- Herausforderung der OO: Wohlüberlegte Zusammenlegung von Attributen und Methoden in einer Klasse (➔ Kapselung).

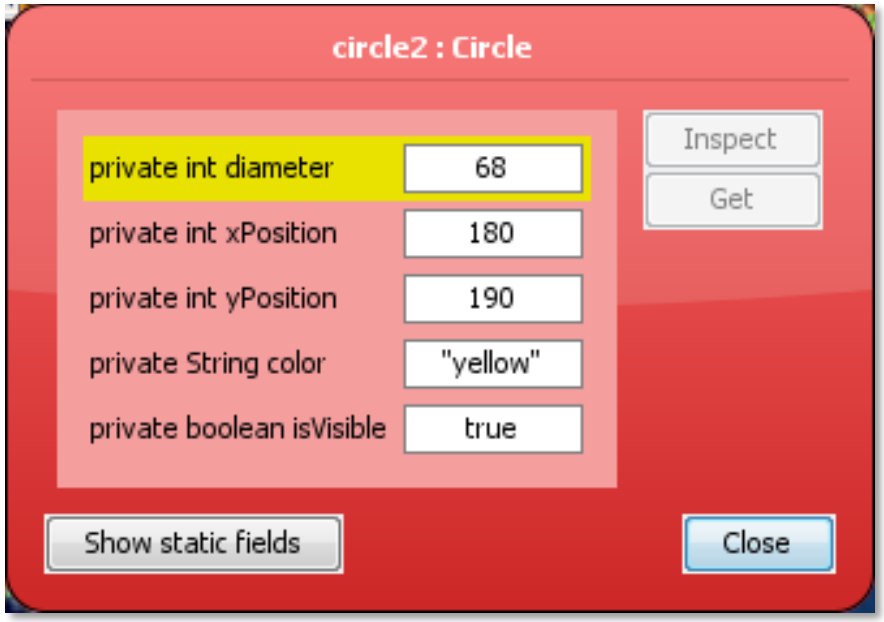
Attribute, Methoden und Parameter

- Objekte halten ihren aktuellen Zustand in Attributen fest.
  - Attribute haben einen ➔Datentyp
- Objekte haben Methoden (Operationen), die aufgerufen werden können, und die das Verhalten definieren.
  - Methoden können Parameter haben, die zusätzlich notwendige Information übertragen.
  - Methoden können ein Resultat zurückliefern (Returnwert).



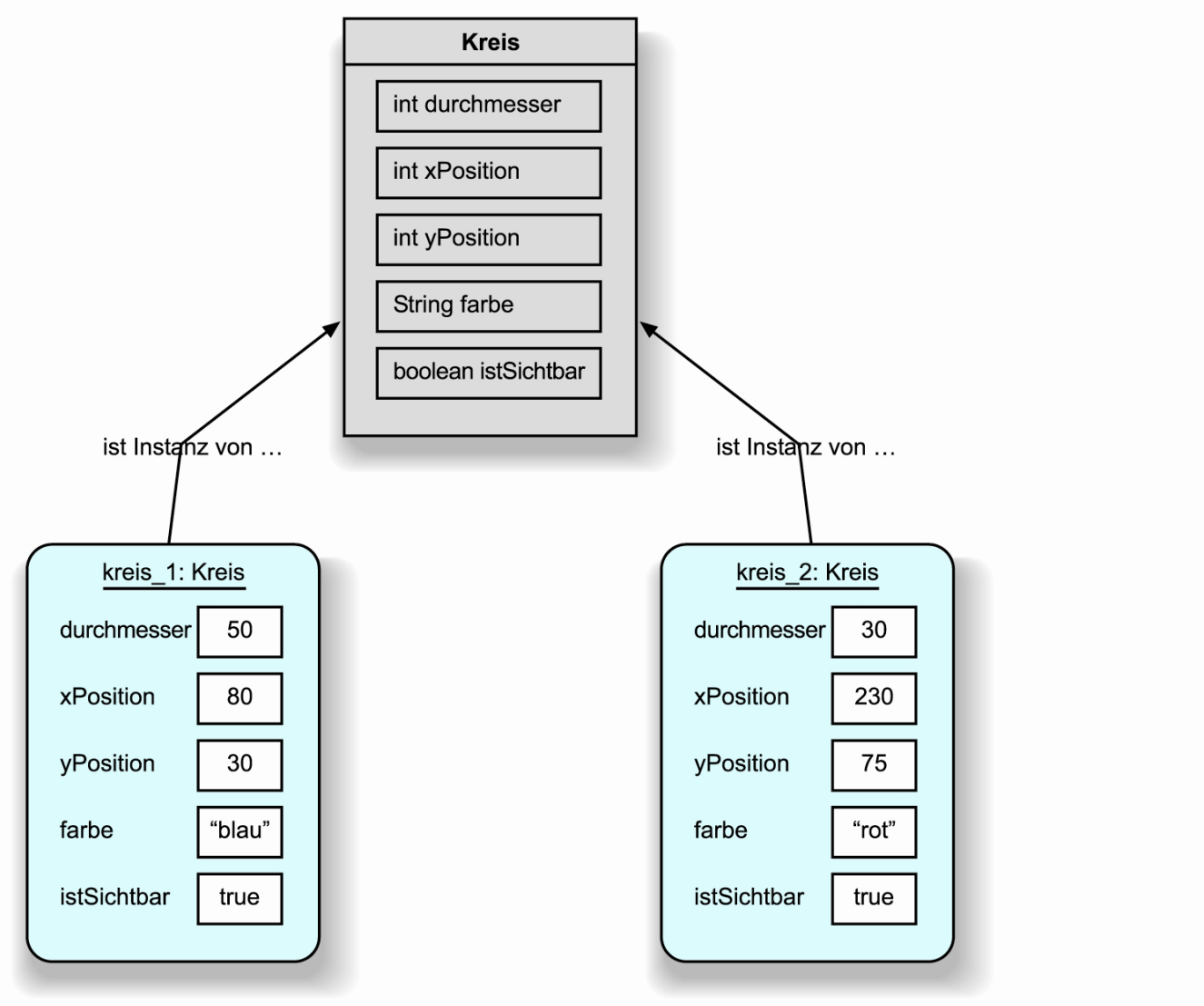
Zustand eines Objekts (Status, State)

- Beispiel der Darstellung eines Objektes in **BlueJ**:



- Die jeweiligen Attribute repräsentieren die Position, den Durchmesser, die Farbe etc. des Kreises.

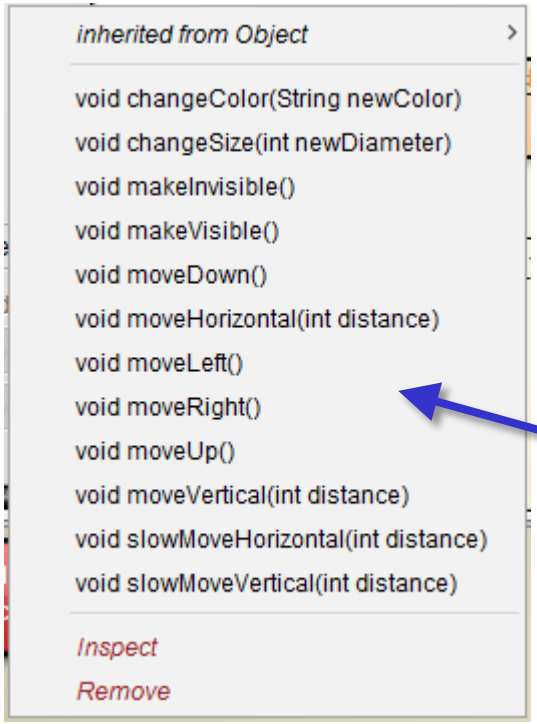
Zwei verschiedene Objekte derselben Klasse



Hinweis: Diese Darstellung entspricht nicht UML (Unified Modeling Language).

Verhalten eines Objektes (Behavior)

- Beispiel (zur Veränderung) des Zustandes eines Objektes in **BlueJ**:



Kontextmenu eines **`Circle`**-Objektes in BlueJ.

Circle
- diameter : int - xPosition : int - yPosition : int - color : String - isVisible : boolean
+ Circle() + makeVisible() : void + makeInvisible() : void + moveRight() : void + moveLeft() : void + moveUp() : void + moveDown() : void + moveHorizontal(distance : int) : void + moveVertical(distance : int) : void + slowMoveHorizontal(distance : int) : void + slowMoveVertical(distance : int) : void + changeSize(newDiameter : int) : void + changeColor(newColor : String) : void - draw() : void - erase() : void

- Durch den Aufruf von Methoden des Objektes können Attribute und somit der Zustand des Objektes verändert werden.
  - Beispiel: **`moveUp()`** verschiebt das Objekt nach «oben».

Quellcode in Java

Quellcode (Sourcecode) in Java

- Jeder Klasse ist Quelltext (Sourcecode) zugeordnet:
  - Dieser wird **pro** Klasse in **einer** Datei abgelegt.
  - Beispiel: Datei **`Circle.java`** für die Klasse **`Circle`**.
- Der Quelltext beschreibt die Details der Klasse:
  - **Attribute** – für die Repräsentation des **Zustandes**.
  - **Methoden** – für die Definition des **Verhaltens**.
- Direkt im Quelltext kann auch Dokumentation im sogenannten JavaDoc-Format enthalten sein.
- Der Quelltext wird mit dem Kompiler zu Bytecode (vergleichbar mit Maschinencode) kompiliert (übersetzt).
  - Der Bytecode kann dann vom Rechner mit Hilfe der «Java Virtual Machine» (JVM, die Laufzeitumgebung) ausgeführt werden.

Quelltext – `Circle.java` (Ausschnitt)

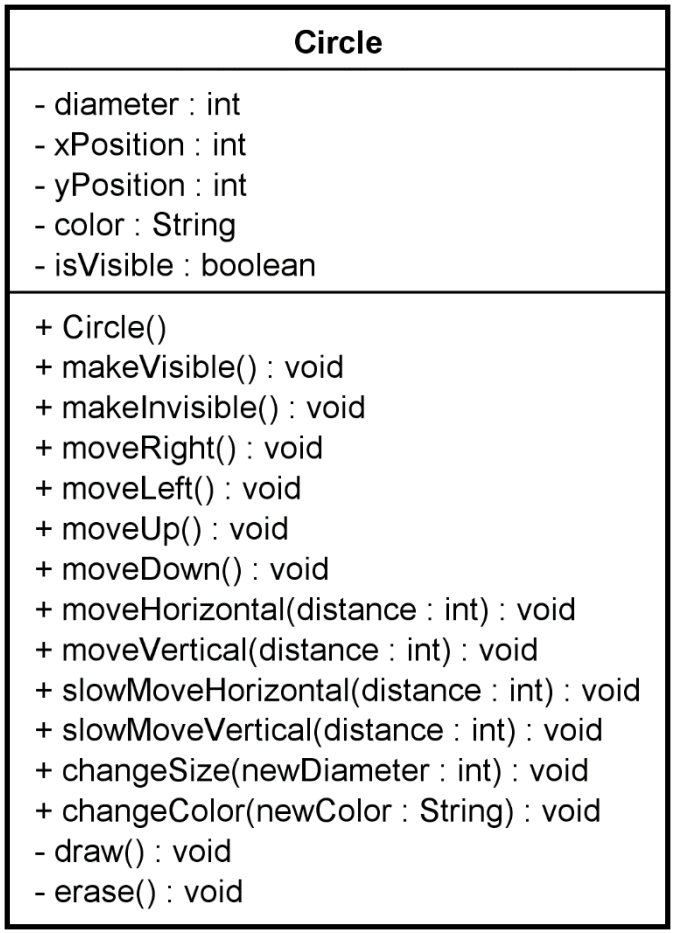
```
public class Circle {

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;

    public Circle() {
        diameter = 68;
        ...
    }

    public void makeVisible() {
        isVisible = true;
        draw();
    }

    public void makeInvisible() {
        erase();
        isVisible = false;
    }
    ...
}
```



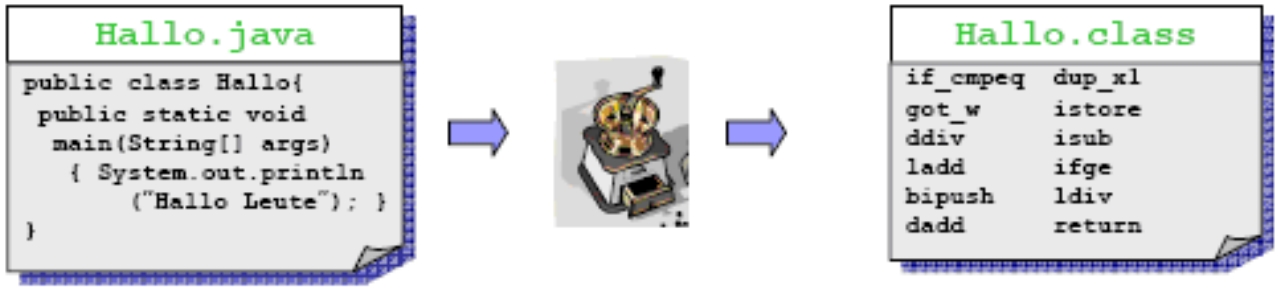
Was ist ein Programm?

Programm

- Ein Programm ist eine Folge von Anweisungen, die einem Rechner (Computer) vorschreiben, was er zu tun hat.
  - Ein Programm kann einen **Algorithmus** umsetzen.
- Programme werden in einer formalen Programmiersprache (Gegensatz: natürliche Sprache) formuliert, die
  - für Rechner präzise und unmissverständlich sind.
  - für Menschen leicht verständlich sein sollen.
- Dabei möchte man idealerweise nicht für jede(n) Rechner(-plattform) eine neue Programmiersprache lernen müssen!

Kompilation von Java Programmen

- Aus einer Textdatei mit der Endung **`.java`** (Sourcecode) erzeugt der Compiler eine Binärdatei mit gleichem Namen aber mit der Endung **`.class`**.
  - Dabei werden die Anweisungen überprüft.
  - Falls (Syntax-)Fehler vorliegen, resultieren Fehlermeldungen.



Quellcode als Datei **`Hallo.java`**

Kompiler **`javac.exe`**

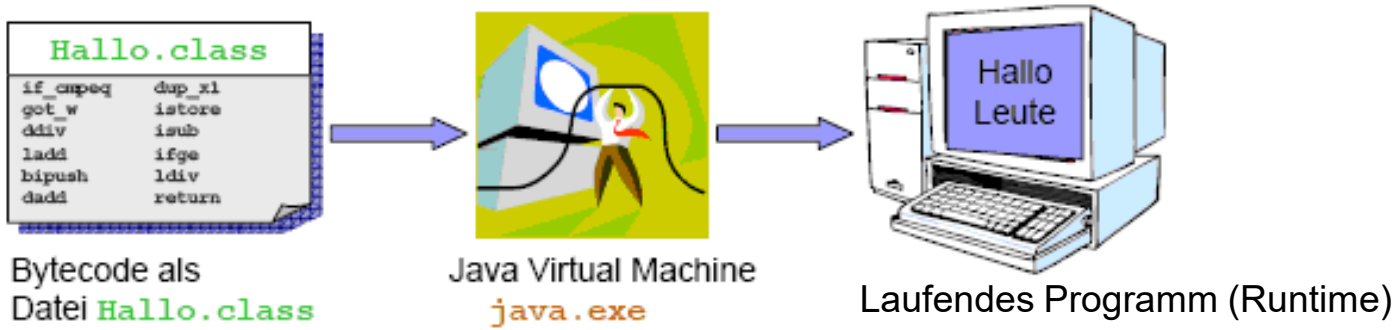
Bytecode als Datei **`Hallo.class`**

- Die **`.class`**-Datei enthält den ausführbaren Bytecode für die Java Virtual Machine (JVM).

Bild nach P. Gumm, Marburg

Ausführung von Java Programmen (Laufzeit)

- Die Datei mit dem Bytecode wird zur Ausführung der Java Virtual Machine (JVM) übergegeben.
- Der Bytecode ist noch immer hardwareunabhängig, d.h. er läuft auf allen Rechnern, für welche eine JVM verfügbar ist.
- Die JVM interpretiert den Bytecode, d.h. führt ihn aus.
  - Der «just-in-time»-Compiler übersetzt den Bytecode dabei ad-hoc in echten Maschinencode ➔ beschleunigt die Ausführung.



- Der Bytecode kann von verschiedenen Sprachen herrühren!
  - z.B. Java, Scala, Clojure, Groovy, JRuby, Jython, Kotlin etc.

Bild nach P. Gumm, Marburg

Live Demo

Aufgabe: Parkhaus

- In 2er-Teams oder alleine:
  Finden Sie Objekte und identifizieren Sie entsprechende Klassen für ein Programm, welches die Parkplätze in einem Parkhaus verwaltet.

Bild: Mulktiparker 710
von Otto Wöhr GmbH

Zusammenfassung

- Klassen beschreiben als Abstraktion wie Objekte aussehen.
  - Beispiele: Rezept, Hausplan, Zeichenschablone.
- Objekte repräsentieren konkrete Instanzen [0..n].
  - Beispiele: Kuchen, Gebäude, Kreise, Blumentöpfe…
- Klassen enthalten Attribute und Methoden.
- Attribute repräsentieren den **Zustand** eines Objektes.
  - Aktuelle Werte der Attribute.
- Methoden repräsentieren das **Verhalten** eines Objektes.
  - Veränderung von Attributwerten.
- Quellcode repräsentiert Klassen
  - In einer bestimmten Sprache.

Fragen?