Programmiermethodik 1
Programmiertechnik

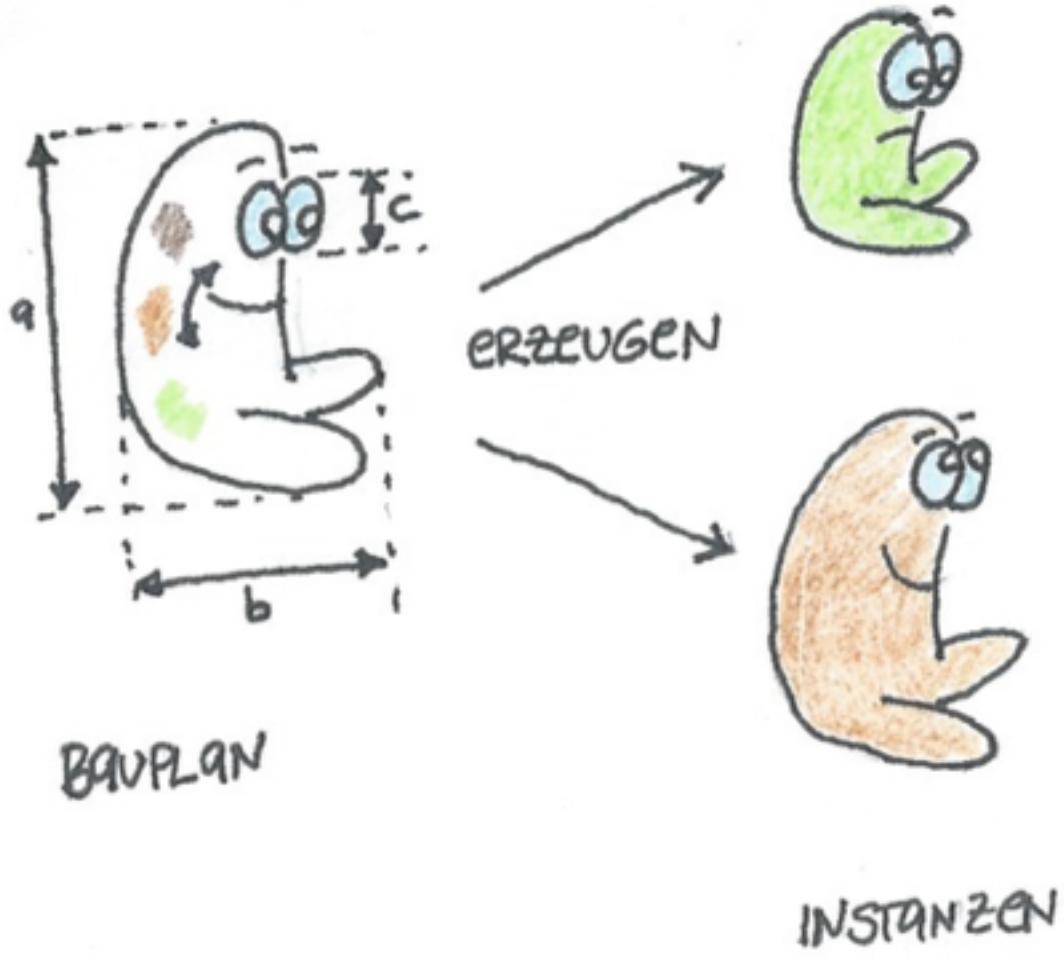
Klassen

Wiederholung

- Erzeugung und Elementzugriff
- Traversierung von Arrays
- Mehrdimensionale Arrays
- Kopieren von Arrays

Ausblick

Worum gehts?



Agenda

- Klassen und Objekte
- Referenztypen
- Objektvariablen
- Vergleich und Lebensdauer
- UML

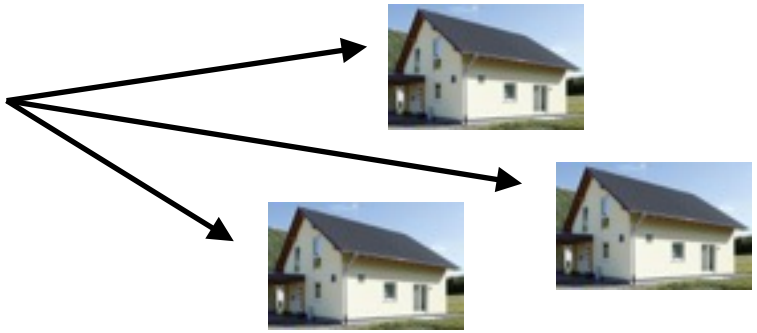
Klassen und Objekte

Klassen und Objekte

- Klasse
- Bauplan

- hat Eigenschaften aus Typ + Name (Objektvariablen)
- definiert Verhalten (Methoden)

- Objekt/Instanz
- entsprechend Bauplan gebaut (zur Laufzeit des Programms)



- Eigenschaften haben konkrete Werte
- kann Verhalten ausführen (Methodenaufrufe)

Klasse

- bisher: einfache (primitive) Datentypen:
  - byte, short, int, long, float, double, char, boolean
  - in Java fest vordefiniert!
  - Ausnahme: Array (bereits Referenztyp)
- jetzt: durch eigene Klassen werden neue Datentypen definiert
  - Definition einer Klasse = Definition eines neuen Typs!

Beispiel: Klasse Bruch

- Datentyp zur Darstellung von Brüchen

```
class Bruch {
   int zaehler;
   int nenner;
}
```

Klasse

- Klassen sind mit eindeutigen Bezeichnern benannt
- In der Regel Substantive, erster Buchstabe groß
- Syntax:

```
class <Klassenname> {
   …
}
```

- Beispiel: Klasse Bruch zur Darstellung von Brüchen

```
class Bruch {
   …
}
```

- jede Klassendefinition sollte in einer eigenen Quelltextdatei stehen
- Dateiname ⇒ <Klassenname>.java
  - Klasse: Bruch
  - Dateiname: Bruch.java

Erzeugen von Objekten

- Erzeugen eines neuen Objektes: Operator new
- auch instanziieren, konstruieren, allokieren
- Syntax:

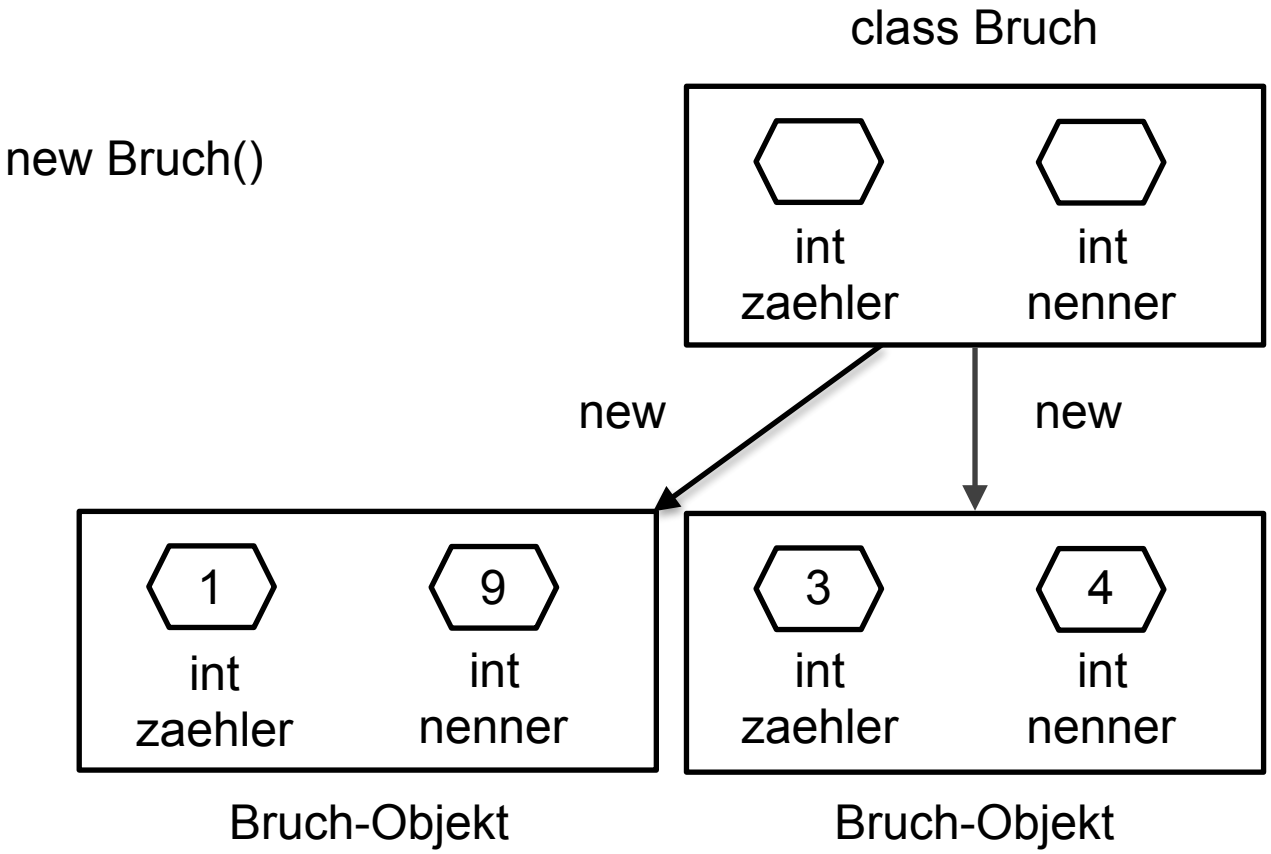
  new <Klassenname>()

- Beispiel:

```
new Bruch()
```

- new produziert aus einer Klassendefinition ein einzelnes, neues Objekt dieser Klasse
- mehrere Objekte ⇒ mehrere Aufrufe von new nötig

Erzeugen von Objekten



Operator new

- new ist unärer Operator
    - Operator: `new`
- Priorität sehr hoch
    - wie bei anderen unären Operatoren
- Operand von new: Klassenname + leere, runde Klammern
    - Operand: `Bruch()`
- Wert (Ergebnis) eines new-Ausdrucks:
    - ein neu erzeugtes Objekt vom Typ <Klassenname>

Objekt

- Klassendefinition ist
    - Bauplan
    - Konstruktionsvorschrift
    - Schema
- Objekte der Klasse müssen explizit geschaffen werden, entstehen nicht von alleine
- Objekt = Exemplar, Instanz (mit eigenen Objektvariablen)
- eine Klassendefinition erlaubt beliebig viele Objekte

Klassendefinitionen und deren Anwendung

- Beispiel: Erzeugen eines Bruch-Objektes in einer Anwendungsklasse

```
public static void main(String[] args) {
   new Bruch();
}
```

- generell: ausführbare Klassen (Programme) benötigen folgende Methode

```
public static void main(String[] args)
```

Übung: Tier

- Erstellen Sie eine Klasse Tier.
- Schreiben Sie eine main()-Methode, in der Sie zwei Instanzen der Klasse Tier erzeugen.

Referenztypen

Referenztypen vs. primitive Datentypen

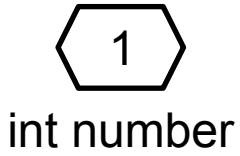
- Bruch = neuer Typ
    - gleichberechtigt neben int, double, boolean etc.
- int, double, boolean etc. sind einfache (primitive) Typen:
    - atomar (nicht unterteilbar)
- Gegensatz: Bruch ist ein Referenztyp
    - enthält separate Bestandteile, diese können einzeln angesprochen und verarbeitet werden
- jede Klasse definiert einen eigenen Referenztyp
- Auswahl primitiver Typen liegt fest
    - können nicht neu definiert werden
- erster Nutzen von Klassen: Bündeln der Bestandteile
    - im Beispiel Bruch: Zähler und Nenner eines Bruchs bleiben immer zusammen

Referenzvariablen

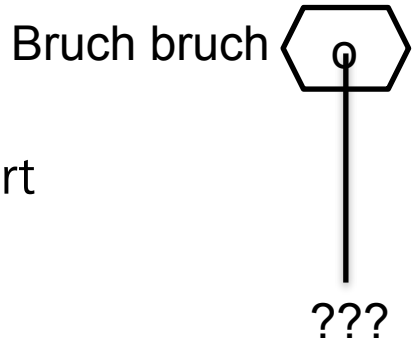
- Klassen definieren Referenztypen
- alle Typen sind für Variablendeklarationen erlaubt
- Referenzvariable = Variable für einen Referenztyp
    - Zeiger auf Objekt
- Wert einer Referenzvariablen ist Referenz ("Pfeil") auf ein Objekt
- Gegensatz: „primitive Variable“ = Variable für einfachen (primitiven) Typ

Deklaration von Referenzvariablen

- primitive Variable:
  - Variable und Speicherplatz für Wert untrennbar gekoppelt
  - einfacher Datentyp
  - Beispiel: int n = 23;



- Referenzvariable:
  - Variable (Zeiger) und Wert (Objekt) existieren unabhängig und getrennt
  - Deklaration einer Referenzvariablen erzeugt nur die Variable, kein Objekt
  - Beispiel: Bruch bruch;
    - Wert ist unbekannt, da nicht initialisiert



Initialisieren einer Referenzvariablen

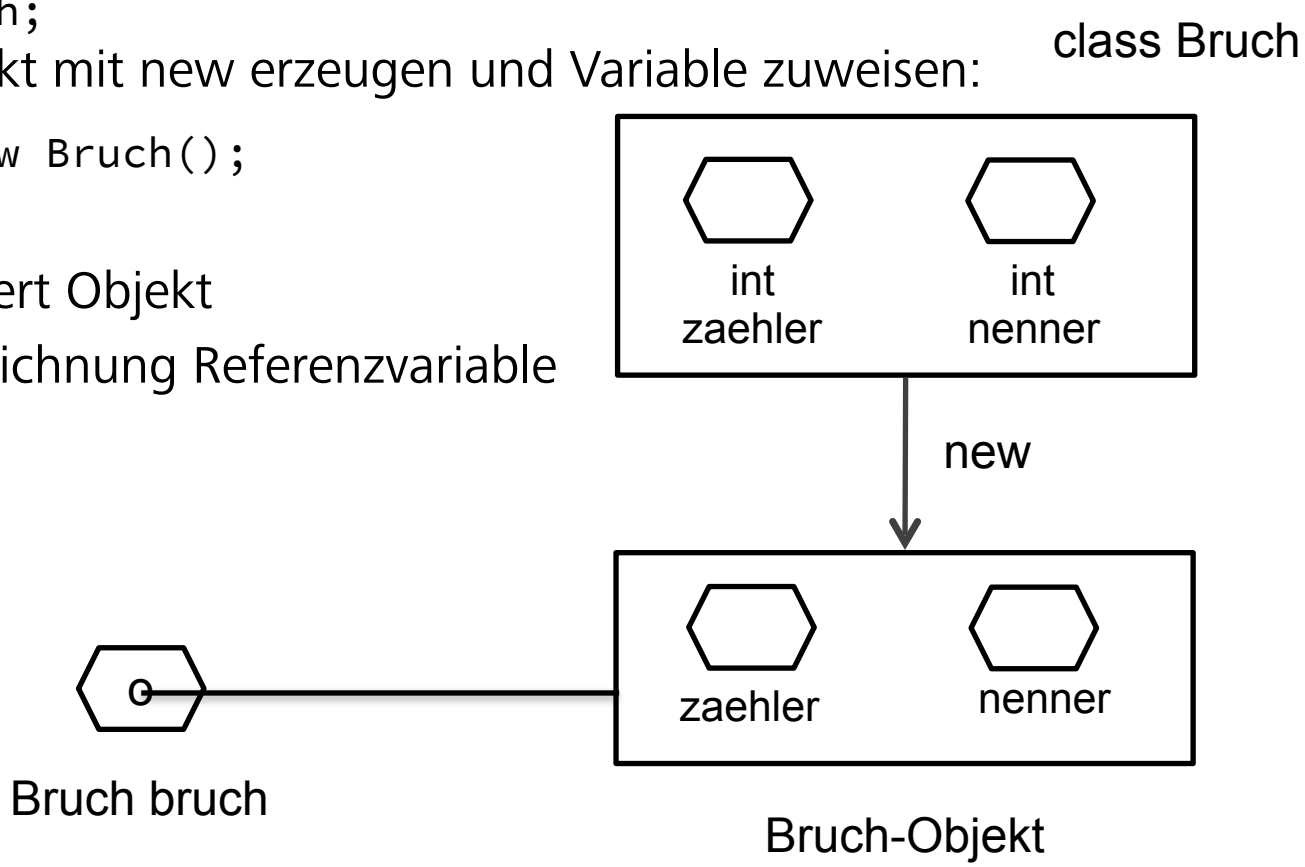
- Deklaration und Initialisierung
  - Referenzvariable deklarieren:

```
Bruch bruch;
```

  - Neues Objekt mit new erzeugen und Variable zuweisen:

```
bruch = new Bruch();
```

- Variable referenziert Objekt
  - daher: Bezeichnung Referenzvariable



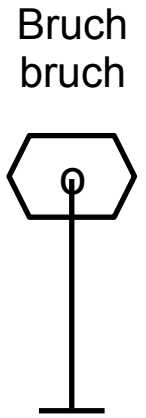
Leere Referenz

- null steht für "kein Objekt"
- null kann an jede Referenzvariable zugewiesen werden:

```
Bruch bruch = null;
```

- null = wohldefinierter Wert, kann verglichen werden Beispiel:

```
if ( bruch == null ){
   System.out.println("no object");
}
```



- neu definierte lokale Variablen sind nicht initialisiert
  - unabhängig vom Typ
- Wert null nicht identisch zu nicht initialisiert
- Best Practice: Weisen Sie einer Referenzvariablen immer null zu, wenn kein Objekt zur Hand ist

Übung: Referenztypen

- Erstellen Sie eine Skizze der Variablen im Speicher für folgenden Quellcode:

```
Tier tier1 = new Tier();
Tier tier2 = new Tier();
Tier tier3 = tier1;
tier2 = tier3;
```

Arrays: Kopieren

- Kopierschleife mit Wertzuweisungen und System.arraycopy erzeugen jeweils eine flache Kopie
    - die Werte der Array-Elemente werden in ein neues Array kopiert
- ausreichend bei Wertesemantik des Elementtyps
    - primitive und unveränderliche Typen
- unzureichend für Arrays mit veränderlichen Objekten als Elementen
    - es werden nur die Referenzen kopiert!

Arrays: Flache Kopie

- Flache Kopien mit arraycopy
  - Hinweis: System.out.println() für Objekte ohne toString()-Methode liefert den Klassennamen und die Speicheradresse

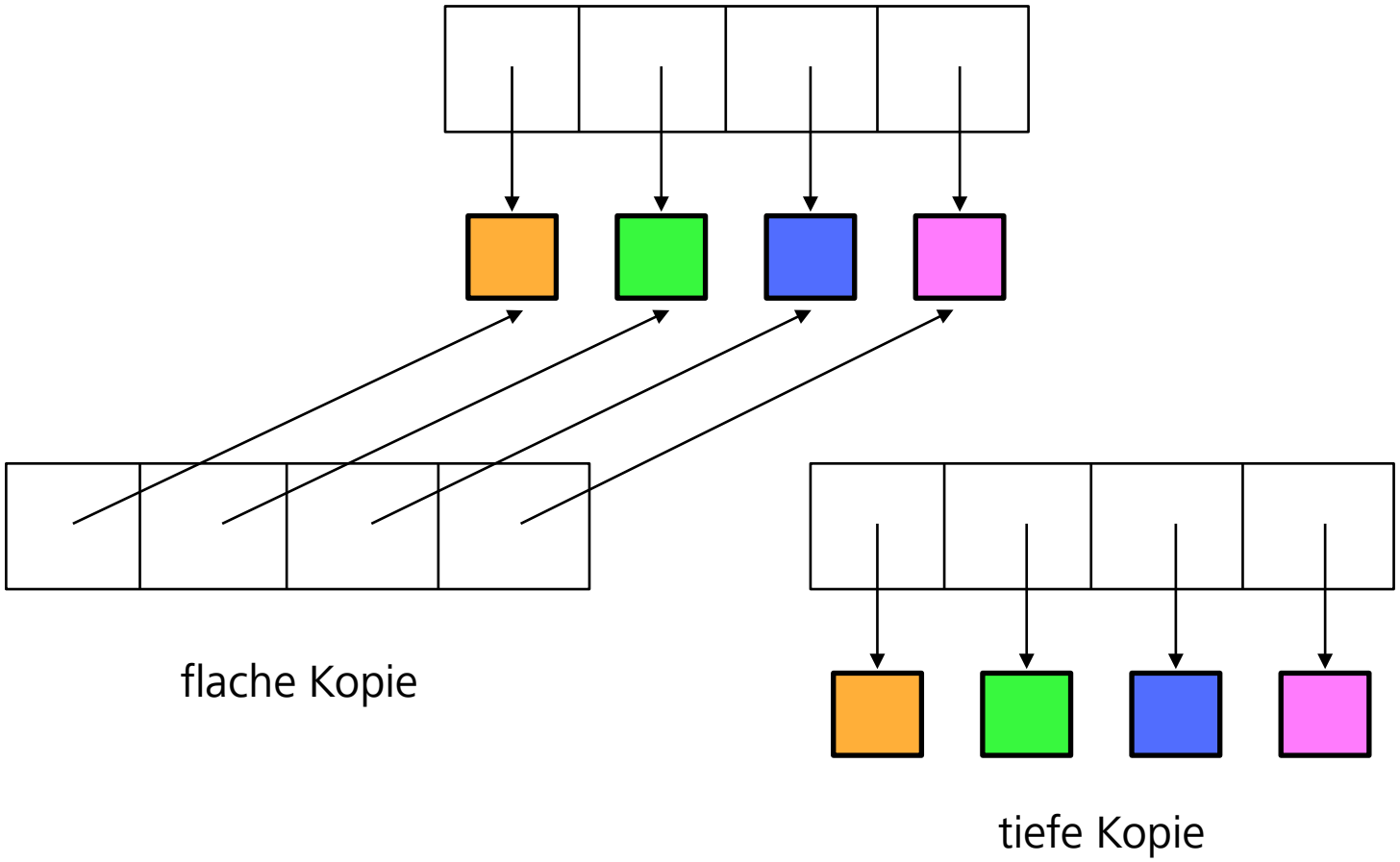
```
Bruch[] quelle = new Bruch [2];
quelle[0] = new Bruch ();
quelle[1] = new Bruch ();
Bruch [] ziel = new Bruch [quelle.length];
System.arraycopy(quelle, 0, ziel, 0, quelle.length);
System.out.println(quelle[0]);
System.out.println(ziel[0]);
```

Arrays: Tiefe Kopie

- es werden auch neue Element-Objekte erzeugt
- erst ganzes Array, dann Elemente einzeln duplizieren!
- Beispiel mit Kopier-Konstruktor:

```
Bruch[] quelle = { new Bruch (), new Bruch () };
Bruch[] ziel = new Bruch [quelle.length];
for (int i = 0; i < quelle.length; i++) {
   ziel [i] = new Bruch(quelle[i]);
}
System.out.println(quelle[0]);
System.out.println(ziel[0]);
```

Flache vs. Tiefe Kopie



Objektvariablen

Objektvariablen

- einer Klasse können Variablen zugeordnet werden
  - Objektvariablen, Instanzvariablen, Membervariablen
  - Beispiel: Bestandteile eines Bruchs: Zähler, Nenner

- alle Bestandteile werden in der Klassendefinition aufgelistet:

```
class Bruch {
   int zaehler;
   int nenner;
   …
}
```

- Anzahl und Typ der Objektvariablen beliebig, im Beispiel:
  - int zaehler; // Objektvariable für den Zähler
  - int nenner; // Objektvariable für den Nenner

Objekt

- Objektvariablen sind Variablen, ebenso wie bisher verwendete Variablen
- bisher benutzte Variablen = lokale Variablen
- Deklarationssyntax von Objektvariablen und lokalen Variablen ist gleich
- aber: Ort der Deklaration ist entscheidend!
    - Objektvariablen: Elemente von Klassen
    - lokale Variablen: Anweisungen in Methoden
- Benennung von Objektvariablen:
    - wie lokale Variablen (erster Buchstabe klein!)
    - eindeutig innerhalb einer Klasse

Objektvariablen

- jedes Objekt enthält die Objektvariablen, die in der Klassendefinition festgelegt sind
- Objektvariablen eines Objekts können einzeln angesprochen werden
    - sogenannter Elementzugriff
- Objekt, an das sich ein Elementzugriff richtet: Zielobjekt
- Syntax für den Zugriff auf eine Objektvariable:
- <Zielobjekt>.<Objektvariablenname>
- <Zielobjekt> ist meist Wert einer Referenzvariablen

Objektvariablen

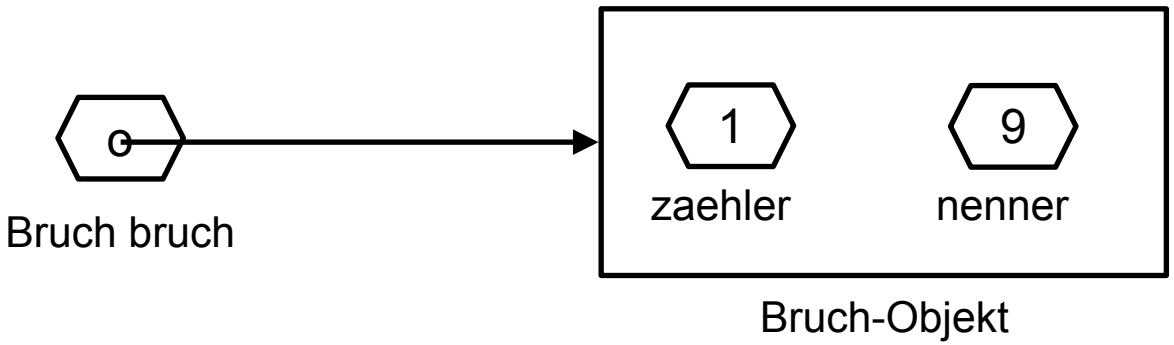
- Erzeugen eines Bruch-Objekts mit Wert 1/9:
  - Bruch-Objekt erzeugen und an Referenzvariable zuweisen:

```
Bruch bruch = new Bruch();
```

  - Werte für Zähler und Nenner des Zielobjekts einzeln zuweisen:

```
bruch.zaehler = 1;
bruch.nenner = 9;
```

  - anschließend: bruch Zielobjekt ist mit 1/9 initialisiert



Objektvariablen

- Elementzugriff spricht Objektvariablen innerhalb eines Objektes an
- gleiche Verwendung wie bei lokalen Variablen!
- Beispiel: Zähler oder Nenner eines Bruch-Objektes in einem Ausdruck verwenden:

```
int i = 5 - bruch.zaehler * 3;
```

- mit Wertzuweisung oder Inkrementoperator modifizieren:

```
bruch.zaehler = 10;
bruch.nenner++;
```

- vergleichen:

```
if ( bruch.zaehler != 0 ){ …
```

- nur die Zugriffssyntax zeigt den Unterschied zwischen Objektvariablen und lokalen Variablen an!

Objektvariablen

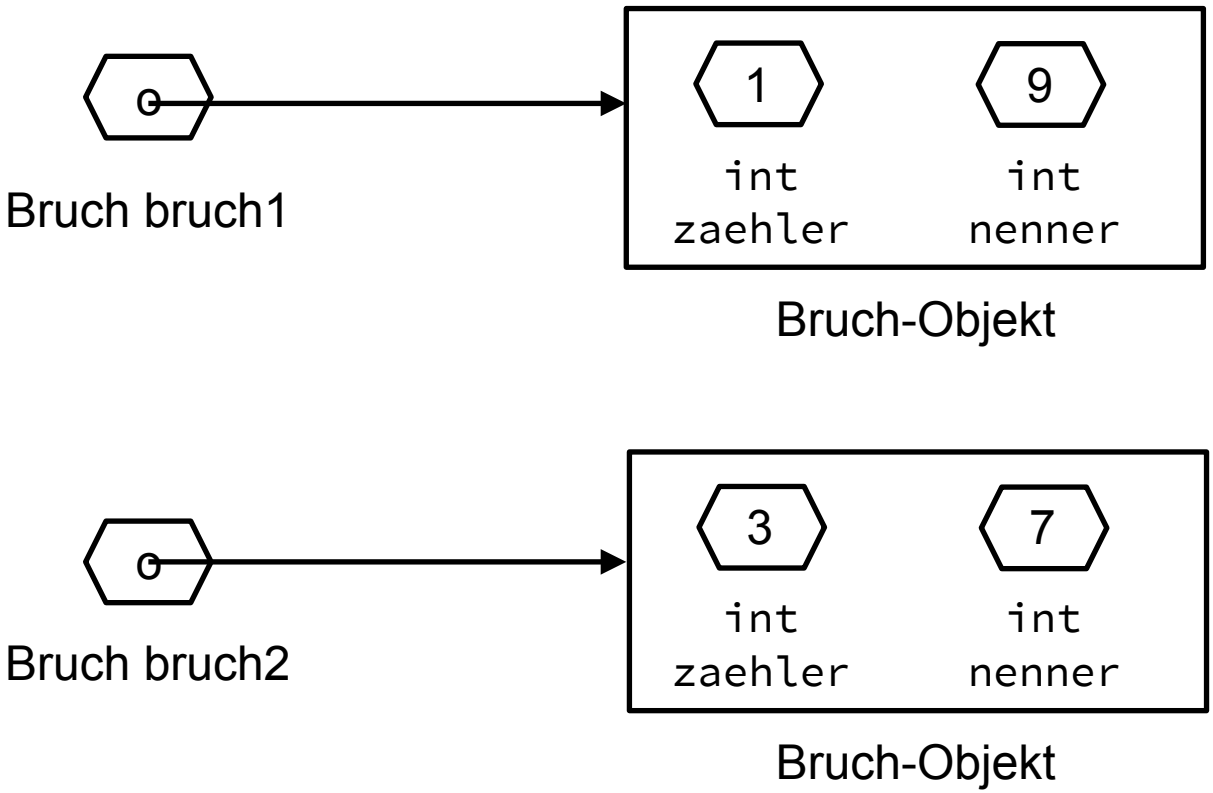
- jedes Objekt hat eigene Objektvariablen
- Elementzugriff richtet sich an eine Objektvariable innerhalb eines Objektes (des Zielobjektes)
- andere Objektvariablen des Zielobjektes und Objektvariablen anderer Objekte bleiben unberührt
- Beispiel: zwei Brüche erzeugen, mit unterschiedlichen Werten initialisieren:

```
Bruch bruch1 = new Bruch(); // 1/9
bruch1.zaehler = 1;
bruch1.nenner = 9;

Bruch bruch2 = new Bruch(); // 3/7
bruch2.zaehler = 3 * bruch1.zaehler;
bruch2.nenner = bruch1.nenner - 2;
```

Objektvariablen

- Beispiel
  - bruch1 und bruch2 sind isolierte, unabhängige Objekte!



Wertzuweisungen

- Wertzuweisung primitiver Typen kopiert den Wert

```
int a = 1;
int b = a;
```

- beide Variablen haben den Wert 1:

- Änderungen einer Variablen ist ohne Auswirkungen auf die andere:

```
…
b++;
System.out.println(a);          // gibt 1 aus
```

- b inkrementiert auf 2, a immer noch 1.

Wertzuweisungen

- Wertzuweisung bei Referenztypen kopiert den Zeiger (die Referenz), nicht das Objekt!
- Beispiel:

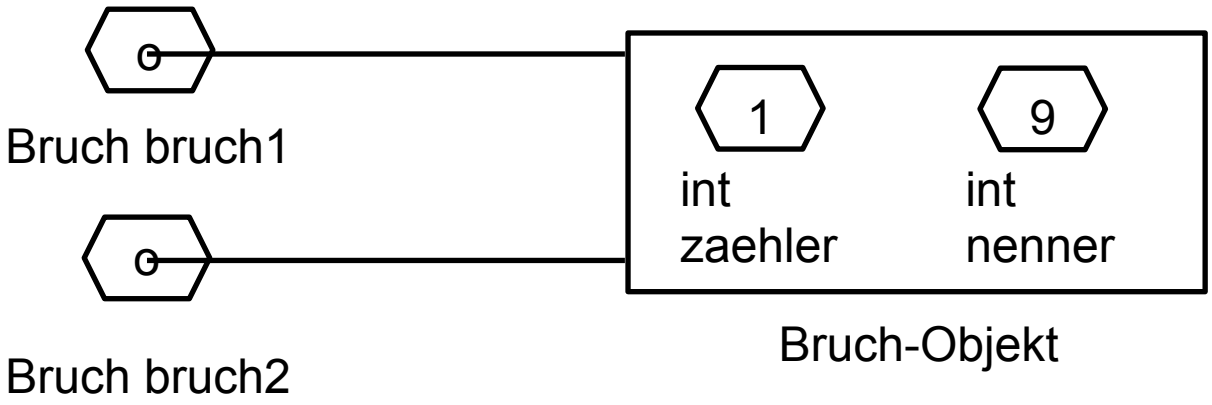
```
Bruch bruch1 = new Bruch();
bruch1.zaehler = 1;
bruch1.nenner = 9;
Bruch bruch2 = bruch1;
```

- beide Variablen referenzieren dasselbe Objekt mit Wert 1/9!
- engl. Bezeichnung, falls jemand mehrere Namen hat: "Aliasing"

Wertzuweisungen

- Änderungen des Objekts sind in beiden Variablen sichtbar:

```
…
bruch2.zaehler++;
System.out.println(bruch1.zaehler); // gibt 2 aus!
```



Übung: Objektvariablen

- Erweitern Sie die Klasse Tier um eine int-Variable alter für das Alter
- Erzeugen Sie ein Tier-Objekt
- Weisen Sie der Variable den Wert 23 zu
- Inkrementieren Sie das Alter um 1.
- Geben Sie das Alter auf der Konsole aus

```
public class Tier {
  int alter;
  public static void main(String[] args) {
    new Tier();
    new Tier();
    Tier tier = new Tier();
    tier.alter = 23;
    tier.alter++;
    System.out.println(tier.alter);
  }
}
```

Vergleich und Lebensdauer

Vergleich von Objekten

- Vergleich von Objekten mit == prüft die Identität:
  - true, wenn beide Operanden ein und dasselbe Objekt sind
  - false, wenn die Operanden verschiedene Objekte sind
- Vergleich mit == ignoriert den Inhalt der Objekte

- Textebene 1

```
Bruch bruch1 = …;
Bruch bruch2 = bruch1;

if ( bruch1 == bruch2 ){ ✔
    …
```

```
Bruch bruch1 = new Bruch();
bruch1.zaehler = 1;
bruch1.nenner = 9;

Bruch bruch2 = new Bruch();
bruch2.zaehler = 1;
bruch2.nenner = 9;

if ( bruch1 == bruch2 ){
                        ✖
…
```

Vergleich von Objekten

- alle Objektvariablen müssen für die inhaltliche Gleichheit paarweise verglichen werden!

```
Bruch bruch1 = ...;
Bruch bruch2 = ...;

if( bruch1.zaehler == bruch2.zaehler &&
    bruch1.nenner == bruch2.nenner ){
      ...
```

- Standardmethode
  - equals(...)
  - später mehr ...

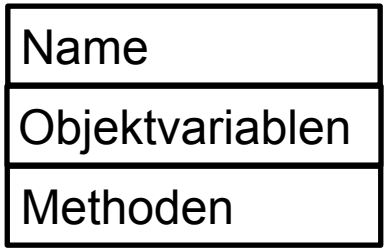
Lebensdauer von Objekten

- lokale Variablen werden ...
    - geschaffen, wenn die Deklaration erreicht wird
    - zerstört, wenn der Block der Deklaration verlassen wird
- Objektvariablen werden ...
    - geschaffen, sobald ein Objekt erzeugt wird (new)
    - zerstört, wenn das Objekt nicht mehr erreichbar ist
        - keine Referenz mehr existiert
        - automatisches "Garbage Collection" (Müll sammeln)
- das heißt:
    - die Lebensdauer von Objekten einschließlich der darin enthaltenen Objektvariablen ist unabhängig von Blockgrenzen!

UML

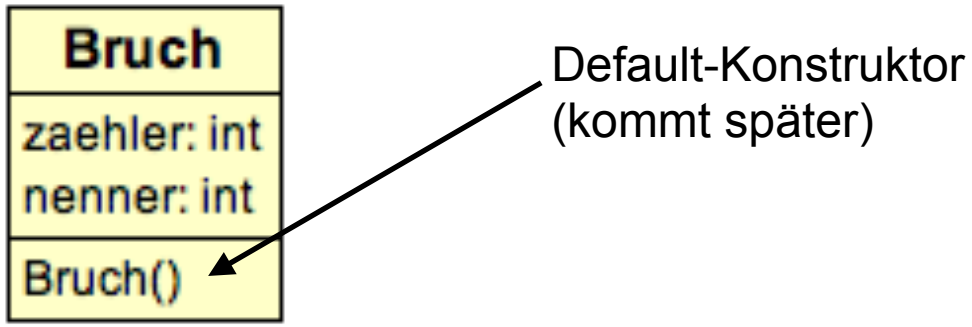
UML Klassendiagramm

- Klasse = Kasten mit drei „Fächern“
  - Name der Klasse
  - Objektvariablen mit Typ und Name
  - Methoden: später



UML Klassendiagramm

- Beispiel: Bruch



Hinweis: Klassendiagramme können in Eclipse einfach mit dem Plugin ObjectAid UML Explorer erzeugt werden (http://www.objectaid.com/).

Übung: Entwurf einer Klasse

- Aufgabe
- Entwerfen Sie eine Klasse CharDoublePaar
    - die Klasse hat zwei Eigenschaften:
        - ein Zeichen
        - eine Fließkommazahl
- Geben Sie den Quellcode an
- Zeichnen Sie ein UML-Diagramm

Zusammenfassung

- Klassen und Objekte
- Referenztypen
- Objektvariablen
- Vergleich und Lebensdauer
- UML