



- · wie werden Klassen erstellt?
- es wird in drei Schritten vorgegangen:

## 1. Planung

aus welchen Teilen sollen die Objekte der Klasse bestehen?

- Führt zur Festlegung der Attribute der Klasse welche Initialwerte sollen die Teile der Objekte (= Attribute) beim Erzeugen erhalten?
- > führt zur Festlegung der Konstruktoren der Klasse

Hochschule für Technik und Wirtschaft Berlin

Klassen

- Klassen sind Baupläne, nach denen bei Bedarf beliebig viele Objekte "gebaut" (erzeugt) werden können
  - im selben Programm oder bei Wiederverwendung später
- formal handelt es sich bei einer Klasse c um die Deklaration eines Referenzdatentyps
  - jedes Objekt, das nach dem Bauplan c erzeugt wird, d.h. jede Instanz der Klasse c, hat den Referenzdatentyp c

© H. Brandenburg

Programmierung 2

Hochschule für Technik und Wirtschaft Berlin

Klassen

welche Fähigkeiten sollen die Objekte der Klasse haben?

- führt zur Festlegung der Methoden der Klasse
- 2. Darstellung des Ergebnis der Planungdazu dient die Unified Modeling Language (UML)
- dabei handelt es sich um eine zunächst von der Firma Rational Software (bis 1997) und dann von der Object Management Group (OMG) entwickelte Sprache zur (grafischen) Modellierung und Dokumentation (nicht nur) objektorientierter Softwaresysteme
  - die Version UML 1.4.2 ist seit 2005 eine ISO-Norm

© H. Brandenburg

Programmierung 2

© H. Brandenburg



- die aktuelle Version 2.4.1 stammt vom 05.09.2011
- da die UML in der Praxis weit verbreitet ist, werden Sie sich mit ihr ausführlich in den Lehrveranstaltungen zum Software Engineering befassen

## 3. Implementierung

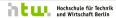
mit Hilfe einer objektorientierten Programmiersprache wie Java oder C++

- das vor allem ist unser Thema
- wir illustrieren die einzelnen Schritte anhand eines einfachen Beispiels

© H. Brandenburg

Programmierung 2

5



Klassen

aus welchen Teilen sollen unsere "Darlehensexperten" bestehen, d.h. die Objekte der Klasse AnnuitaetenDarlehen?

- es ist äußerst wichtig, nur solche Teile zu betrachten, die unabdingbar sind zur Beschreibung der Gestalt
- charakteristisch für ein (einfaches) Annuitätendarlehen sind die Höhe des Darlehens, die Laufzeit des Darlehens (in Jahren), der (effektive) Jahreszins und das Datum der Auszahlung des Darlehens (und nichts anderes!)
- es ist üblich, dass der erste Buchstabe des Names von Attributen klein geschrieben wird

Programmierung 2

Hochschule für Technik und Wirtschaft Berlin

Klassen

 wir wollen eine Klasse gestalten, deren Objekte Experten für Annuitätendarlehen sind

## 1. Planung

jede Klasse erhält einen Namen

- der Name soll immer ein Substantiv sein
- er soll einen möglichst genauen Hinweis darauf geben, was die Objekte der Klasse können, d.h. wofür sie "Experten" sind
- es ist üblich, dass der erste Buchstabe des Namens groß geschrieben wird
- wir nennen unsere Klasse AnnuitaetenDarlehen

© H. Brandenburg

Programmierung 2

Hochschule für Technik und Wirtschaft Berlin

Klassen

wir sehen vier Attribute vor und nennen sie betrag, laufzeit, zinssatz und startdatum

welche Initialwerte sollen die Attribute beim Erzeugen der Objekte erhalten?

- wir sehen zwei Konstruktoren vor:
- einen, dem die Werte für betrag, laufzeit, zinssatz und startdatum als Input übergeben werden
- einen, dem nur die Werte für betrag, laufzeit und zinssatz als Input übergeben werden
  - er soll startdatum auf das aktuelle Datum setzen

© H. Brandenburg

Programmierung 2

8

© H. Brandenburg



welche Fähigkeiten sollen unsere "Darlehenssexperten" haben, was sollen sie können?

- die Objekte der Klasse AnnuitaetenDarlehen statten wir der Einfachheit halber (zunächst) nur mit sehr einfachen Fähigkeiten aus:
- > sie sollen Auskunft darüber geben können, woraus sie bestehen, d.h. über den Wert ihrer Attribute
- > sie sollen das Datum liefern können, an dem das Darlehen getilgt ist
- > sie sollen Auskunft geben können über
  - die zu zahlende Raten
  - · die insgesamt zu zahlenden Zinsen
  - · den Gesamtaufwand

© H. Brandenburg

Hochschule für Technik Klassen liefereDarlehensBetrag liefereLaufzeit liefereZinssatz liefereAuszahlungsDatum liefereTilgungsEnde liefereJaehrlicheRate liefereZinsenBeiJaehrlicherRate liefereGesamtAufwandBeiJaehrlicherRate liefereMonatlicheRate liefereZinsenBeiMonatlicherRate liefereGesamtAufwandBeiMonatlicherRate © H. Brandenburg Programmierung 2



Klassen

- > und das jeweils bei jährlicher und bei monatlicher Ratenzahlung
- > jede Fähigkeit ist durch eine Methode zu realisieren
- den Namen der Methoden soll man entnehmen können, was sie machen
  - · sie werden daher in der Regel nach dem Schema macheEtwas

gebildet

- > es ist üblich, dass der erste Buchstabe des Names von Methoden klein geschrieben wird
- wir sehen 11 Methoden vor und nennen sie:

© H. Brandenburg

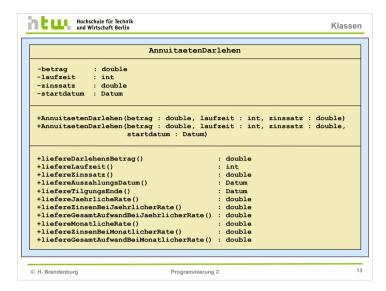
Programmierung 2



Klassen

2. Darstellung des Ergebnis der Planung mit der UML beschreibt man Klassen so:

© H. Brandenburg Programmierung 2

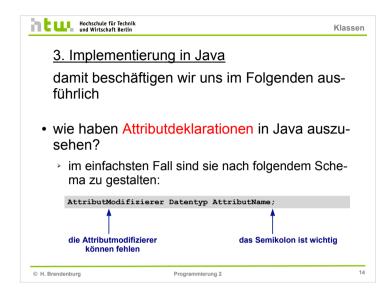




15

- die Namen von Attributen können frei gewählt werden
  - sie müssen allerdings den allgemeinen Regeln für Bezeichner in Java genügen
  - innerhalb einer Klasse müssen alle Attribute unterschiedliche Namen haben
  - sie sollen so gewählt werden, dass sie selbsterklärend (mnemonisch) sind
  - es ist üblich, den ersten Buchstaben von Attributen klein zu schreiben
    - es sei denn, sie sind mit dem Attributmodifizierer final versehen

© H. Brandenburg Programmlerung 2





Klassen

- als Datentyp von Attributen ist jeder Datentyp zugelassen
  - es kann sich insbesondere auch um eine Klasse, ein enum, ein Interface oder ein Array handeln, d.h. Attribute können (und werden oft) selbst Objekte sein Bemerkung: selbst die eigene Klasse ist zugelassen (Rekursion)
- Objekte können daher eine beliebig komplexe Gestalt haben, d.h. es ist möglich, beliebig komplexe "Experten" zu erschaffen

© H. Brandenburg

Programmierung 2

16



- die Attributmodifizierer dienen dazu, die deklarierten Attribute mit speziellen Eigenschaften zu versehen
- es gibt drei Arten von Attributmodifizierern:
  - Annotations
  - > die Zugriffsmodifizierer public, protected und private
  - > die sonstigen Attributmodifizierer static, final, transient und volatile
- eine Attributdeklaration kann mehrere Attributmodifizierer enthalten

© H. Brandenburg

© H. Brandenburg

Programmierung 2

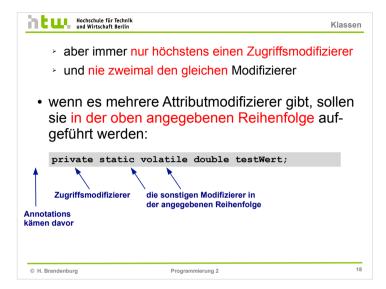
17



Klassen

- wir betrachten zunächst nur die Zugriffsmodifizierer
  - mit Annotations und den sonstigen Attributmodifizierern, die unterschiedlichen Zwecken dienen, beschäftigen wir uns zu gegebener Zeit
- die Zugriffsmodifizierer legen fest, von wo aus auf das Attribut zugegriffen werden kann
  - d.h. wo überall das Attribut benutzt werden kann.
- für jedes Attribut gilt, dass die Konstruktoren und Methoden der eigenen Klasse immer Zugriff darauf haben

Programmierung 2





Klassen

- private legt fest, dass der Zugriff nur vom Code der eigenen Klasse aus möglich ist
  - Methoden anderer Klassen können auf Attribute, die als private deklariert wurden, nicht zugreifen, d.h. sie existieren für sie nicht
- public legt dagegen fest, dass der Zugriff von überall her möglich ist
- protected wird erst im Zusammenhang mit Vererbung relevant, weshalb wir diesen Zugriffsmodifizierer zunächst ignorieren

© H. Brandenburg



- wird kein Zugriffsmodifizierer angegeben, kann das Attribut in allen Klassen des Packages benutzt werden, zu dem die Klasse gehört
  - weil die Klassen des eigenen Packages als "Freunde" betrachtet werden, wird dieser Zustand auch als "friendly" bezeichnet
  - Achtung: wird einfach nur vergessen, ein Attribut mit einem Zugriffsmodifizierer zu versehen, handelt es sich in der Regel um einen Programmierfehler!
- welche Zugriffsmodifizierer sollen wir für die Attribute unserer Klasse AnnuitaetenDarlehen wählen?

© H. Brandenburg

Programmierung 2

21

23



Klassen

- auf die Attribute betrag, laufzeit, zinssatz und startdatum eines Objektes der Klasse AnnuitaetenDarlehen können dann (bei einfachen Klassen) nur zugreifen
  - die Konstruktoren (einmalig, beim Erzeugen des Objektes)
  - und die Methoden der Klasse
- damit haben wir als Designer der Klasse
   AnnuitaetenDarlehen die volle Kontrolle
   darüber, was mit den Attributen betrag,
   laufzeit, zinssatz und startdatum
   geschehen kann

© H. Brandenburg Programmierung 2



Klassen

- wir erinnern uns an die Grundidee der objektorientierten Programmierung:
  - Aufgaben sollen durch das Zusammenwirken von Objekten (d.h. von "Experten") gelöst werden
  - dabei ist nur wichtig, was die Objekte k\u00f6nnen, nicht aber, wie sie etwas machen
  - insbesondere ist völlig unwichtig, welche Gestalt sie haben
- es ist daher sinnvoll, die Attribute der Klasse AnnuitaetenDarlehen als private zu deklarieren

© H. Brandenburg

Programmierung 2



Klassen

- wir können zum Beispiel dafür sorgen, dass niemals unsinnige Darlehensobjekte erzeugt werden können
- diese Beobachtung können wir sofort verallgemeinern:
  - als private deklarierte Attribute jeder Klasse sind vor unberechtigten Zugriffen von außen geschützt
  - ihre Werte können nur in dem Maße manipuliert werden, wie es der Designer der Klasse zulässt
- dies hat enorme Vorteile für die Wartung von Programmen

© H. Brandenburg

Programmierung 2

24

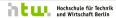


- weil der Zugriff nur über Konstruktoren und Methoden der Klasse möglich ist, müssen im Fehlerfall nur diese (und ihr Input) untersucht werden
- der Rest des (möglicherweise riesigen) Programms braucht nicht betrachtet zu werden
- der Vorteil ist so groß, dass die Möglichkeit, die Attribute einer Klasse vor dem Zugriff von außen schützen zu können, als eine der drei charakteristischen Eigenschaften objektorientierter Programmierung angesehen wird
  - man spricht von Datenkapselung (encapsulation)
  - sie hat wesentlich zum Erfolg der objektorientierten Programmierung beigetragen

© H. Brandenburg

Programmierung 2

25



Klassen

- zu beachten ist, dass wir als Datentyp für das Attribut startdatum – ganz im Sinn der Wiederverwendung – GregorianCalendar aus der Java-Klassenbibliothek gewählt haben
  - denn die Klasse ist hervorragend für die Behandlung von Datumsangaben geeignet
    - **Hinweis:** Teile der Klasse Date der Java-Klassenbibliothek gelten als veraltet (deprecated) und sollten dafür nicht mehr benutzt werden
  - startdatum ist also selbst ein "Experte", der eine komplexe Struktur hat

© H. Brandenburg Programmlerung 2 27



```
Hochschule für Technik und Wirtschaft Berlin
```

Klassen

- der Vollständigkeit halber erwähnen wir, dass es möglich ist, mit einer Deklaration mehrere Attribute zu deklarieren
  - dazu sind lediglich mehrere Attributnamen durch Kommata getrennt – anzugeben:

```
private double betrag, zinssatz;
```

 weil den einzelnen Attributen aber kein Dokumentationskommentar zugeordnet werden kann, empfehlen wir das nicht

© H. Brandenburg

bute immer automatisch einen I	nitialwert
r er hängt vom Datentyp des Attribu	uts ab:
<u>Datentyp</u>	<u>Initialwert</u>
byte, short, int, long	0
char	'\u0000'
float	+0.0f
double	+0.0
boolean	false
Referenzdatentyp	null



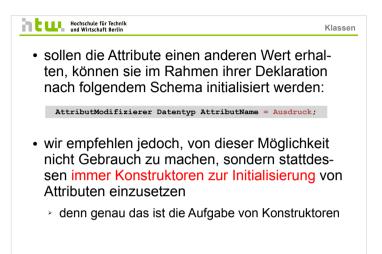
Hochschule für Technik

Klassen

Klassen

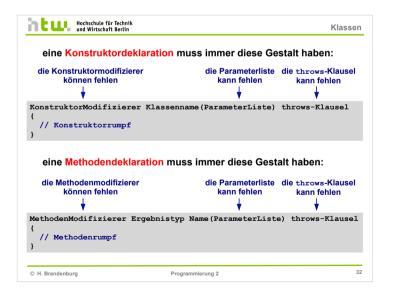
- Konstruktoren dienen dazu, Objekte in einen sinnvollen Initialzustand zu bringen
  - > analog zur Werkseinstellung von Geräten
- sie können aufgefasst werden als spezielle Methoden
  - speziell, weil sie automatisch und nur einziges Mal
     beim Erzeugen des Objektes ausgeführt werden
  - > speziell auch, weil sie nicht vererbt werden
- wie die folgenden Schemata zeigen, sehen Konstruktordeklarationen und Methodendeklarationen nahezu gleich aus

© H. Brandenburg Programmierung 2 31



Programmierung 2

© H. Brandenburg





- bei Konstruktoren gibt es gegenüber Methoden folgende Besonderheiten:
  - ihr Name muss identisch sein mit dem Namen der Klasse
  - weil sie keinen Wert liefern, fehlt bei ihnen die Angabe des Datentyps für den Rückgabewert
- als Konstruktormodifizierer sind zulässig:
  - Annotations
  - > die Zugriffsmodifizierer public, protected und private

© H. Brandenburg

Programmierung 2

33

35



Klassen

- Achtung: manche Kombinationen sonstiger Modifizierer sind nicht zulässig, weil sie Dinge bewirken würden, die sich gegenseitig ausschließen
- wie in C/C++ bestehen die Parameterlisten von Konstruktoren und Methoden aus keiner, einer, oder mehreren Parameterdeklarationen, die gegebenenfalls durch Kommata zu trennen sind
  - Achtung: wenn es keinen Input gibt, ist innerhalb der Klammern nicht – wie in C – void, sondern einfach nichts anzugeben

© H. Brandenburg Programmierung 2



Klassen

- Methodenmodifizierer sind
  - Annotations
  - > die Zugriffsmodifizierer public, protected und private
  - die sonstigen Modifizierer abstract, static, final, synchronized, native und strictfp
- sowohl Konstruktor- als auch Methodendeklarationen können mehrere Modifizierer enthalten
  - aber immer nur höchstens einen Zugriffsmodifizierer
  - und nie zweimal den gleichen Modifizierer

© H. Brandenburg

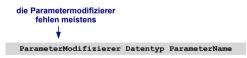
Programmierung 2

34



Klassen

 gewöhnliche Parameterdeklarationen sehen dabei im Prinzip aus wie die Deklaration von Parametern in C/C++ :



 seit der J2SE 5.0 gibt es zur Deklaration von Methoden mit variabler Parameteranzahl eine weitere Form der Parameterdeklaration, die wir erst im Zusammenhang mit Arrays betrachten

© H. Brandenburg



- Konstruktoren und Methoden können beliebig viele Parameter haben
- die Namen von Parametern können frei gewählt werden
  - sie müssen aber den allgemeinen Regeln für Bezeichner in Java genügen
  - innerhalb einer Parameterliste müssen alle Parameter verschiedene Namen
  - sie sollen so gewählt werden, dass sie selbsterklärend (mnemonisch) sind
  - es ist üblich, den ersten Buchstaben von Parametern klein zu schreiben

© H. Brandenburg

Programmierung 2

37

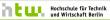
39



Klassen

- als Parametermodifizierer sind zulässig:
  - Annotations
  - final
- final entspricht const in C/C++:
  - wird ein Parameter als final deklariert, kann der empfangene Wert im Rumpf des Konstruktors oder der Methode nicht geändert werden
  - obwohl das durchaus nützlich ist, wird von dieser Möglichkeit in Java relativ selten Gebrauch gemacht
  - es gibt aber auch (sehr spezielle) Situationen, in denen ein Parameter als final deklariert werden muss

© H. Brandenburg Programmlerung 2



Klassen

- Achtung: die Namen von Parametern sollten verschieden sein von den Namen der Attribute der Klasse
  - dies ist nicht zwingend, erspart aber zusätzlichen Aufwand bei der Gestaltung mancher Anweisungen im Konstruktor- oder Methodenrumpf
- als Datentyp von Parametern ist jeder Datentyp zulässig, d.h. alle primitiven Datentypen und alle Referenzdatentypen
  - sogar die eigene Klasse, was oft nützlich ist
- dadurch kann der Input von Konstruktoren und Methoden beliebig komplex sein

© H. Brandenburg

Programmierung 2

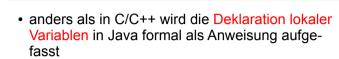
38



Klassen

- bei der Gestaltung des Rumpfes von Konstruktoren und Methoden gibt es ebenfalls keine Unterschiede
  - in beiden Fällen besteht er aus keiner, einer oder mehreren Anweisungen
- es gibt drei Arten von Anweisungen, die (nahezu) beliebig gemischt werden können
  - Anweisungen zur Deklaration lokaler Variablen
  - gewöhnliche Anweisungen
  - Anweisungen zur Deklaration innerer Klassen

© H. Brandenburg



> sie ist nach folgendem Schema zu gestalten:



- die Namen lokaler Variablen können frei gewählt werden
  - sie müssen aber den allgemeinen Regeln für Bezeichner in Java genügen

© H. Brandenburg

Programmierung 2

41

43



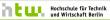
Hochschule für Technik

Klassen

Klassen

- denn sonst ist das gleichnamige Attribut im Geltungsbereich der Variablen unsichtbar, d.h. es kann dort nicht benutzt werden
- dadurch können leicht Fehler entstehen, die schwer zu finden sind
  - · weil es keine Unterstützung durch den Compiler gibt
- als Datentyp lokaler Variablen ist jeder Datentyp zulässig, d.h. alle primitiven Datentypen und alle Referenzdatentypen
  - > sogar die eigene Klasse, was gelegentlich nützlich ist

© H. Brandenburg Programmierung 2



Klassen

- sie müssen ferner verschieden sein von den Namen der Parameter des Konstruktors oder der Methode
- sie sollen so gewählt werden, dass sie selbsterklärend (mnemonisch) sind
- es ist üblich, den ersten Buchstaben lokaler Variablen klein zu schreiben
  - es sei denn, sie wurden als final deklariert
- was passiert, wenn eine lokale Variable denselben Namen erhält wie ein Attribut der Klasse?
  - das ist zulässig, sollte aber unbedingt vermieden werden!

© H. Brandenburg

Programmierung 2



Klassen

- wie bei Parametern sind nur folgende Variablenmodifizierer zulässig:
  - Annotations
  - > final
- werden lokalen Variablen als final deklariert, kann ihr Wert nach der Initialisierung nicht mehr geändert werden, d.h. sie sind in Wirklichkeit Konstanten
  - damit sie leicht zu erkennen sind, ist es üblich, sie stets vollständig groß zu schreiben und \_ zur Strukturierung langer Bezeichner zu benutzen

final double MEHRWERT STEUER SATZ = 0.19;

© H. Brandenburg

Programmierung 2

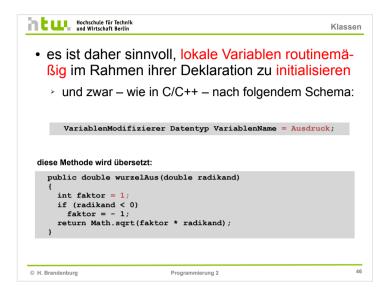
44

```
Hochschule für Technik
                                                          Klassen
 • Achtung: anders als Attribute werden lokale
   Variablen von Konstruktoren und Methoden nicht
   automatisch vom Compiler initialisiert
    > ihnen muss immer explizit ein Wert zugewiesen wer-
      den, bevor sie benutzt werden können
 diese Methode wird nicht übersetzt:
   public double wurzelAus(double radikand)
     final int FAKTOR;
     if (radikand < 0)
       FAKTOR = -1:
     return Math.sgrt(FAKTOR * radikand);
 variable FAKTOR might not have been initialized
     return Math.sqrt(FAKTOR * radikand);
 1 error
© H. Brandenburg
                           Programmierung 2
```

```
Hochschule für Technik
                                                         Klassen

    obwohl es üblich ist, als final deklarierte lokale

   Variablen bereits im Rahmen ihrer Deklaration
   zu initialisieren, ist dies nicht zwingend
    ihnen kann auch noch später – aber nur einmal –
      ein Wert zugewiesen werden
    > eine als final deklarierte Variable, die nicht initiali-
      siert wurde, wird als blank final bezeichnet
 auch diese Methode wird übersetzt:
   public double wurzelAus(double radikand)
     final int FAKTOR;
     if (radikand < 0)
       FAKTOR = -1;
       FAKTOR = 1;
     return Math.sqrt(FAKTOR * radikand);
© H. Brandenburg
```



Hochschule für Technik und Wirtschaft Berlin

 der Vollständigkeit halber erwähnen wir, dass es möglich ist, mit einer Deklaration mehrere lokale Variablen zu deklarieren, wovon wir aber abraten

Klassen

- für die Gestaltung guter Klassen ist es äußerst wichtig, den fundamentalen Unterschied zwischen Attributen und lokalen Variablen zu verstehen
  - Attribute beschreiben die Gestalt der Objekte der Klasse, ihre Werte deren aktuellen Zustand
  - Variablen sind lediglich ein lokaler Zwischenspeicher, der nur während der Ausführung des jeweiligen Konstruktors oder der Methode zur Verfügung steht

© H. Brandenburg Programmierung 2