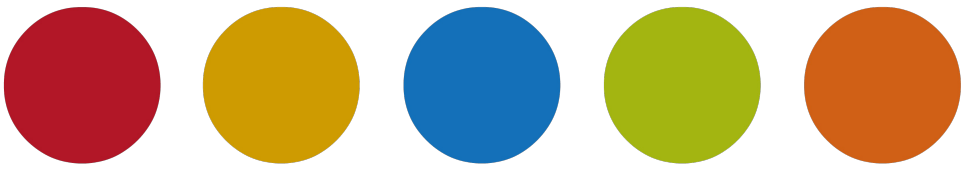
Unterrichtsmaterialien

Skript

Arbeitsmaterial

Schulung:	Grundlagen Softwareentwicklung in Java

Stand: 9. Jun 2016



Christine Janischek
Berufliches Schulzentrum Wangen

Inhaltsverzeichnis

1 Grundlagen der objektorientierten Programmiersprache Java

Grundlagen in Java

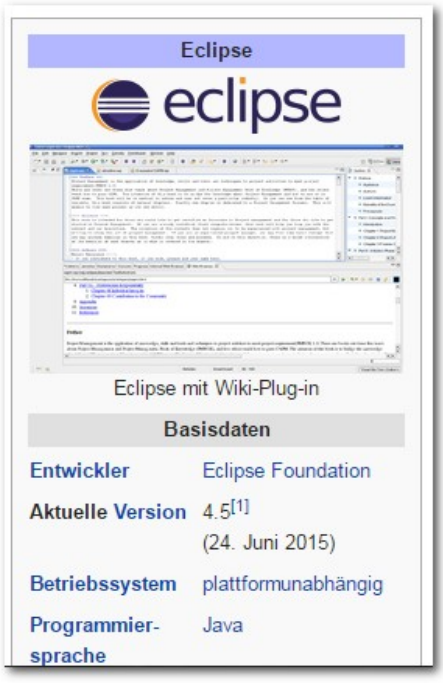
Thema:	Entwicklungsumgebungen in der objektorientierten Programmierung Urquelle: Christine Janischek Überblick: Gängige Entwicklungsumgebungen

Java-Editor	
Maintainer	Gerhard Röhner
Aktuelle Version	12.21 (9. Juni 2014)
Betriebssystem	Microsoft Windows
Programmier-sprache	Delphi
Kategorie	IDE
Lizenz	Freeware
Deutschsprachig	ja
javaeditor.org	

https://de.wikipedia.org/wiki/Java-Editor

Der JavaEditor ist mit Sicherheit eine der einfachsten Entwicklungsumgebungen für die objektorientierten Softwarentwicklung in Java.

Der Java-Editor ist ein Editor zum Programmieren mit der Programmiersprache Java unter Windows. Diese Windows-Version kann in GNU/Linux-Distributionen wie Ubuntu mit Wine benutzt werden. Laut der Wine-AppDB wird das Programm von Wine unter Linux vollständig unterstützt. Die Software ist Freeware, einfach gestaltet und stellt geringe Systemanforderungen an den Rechner. Der Editor ist wegen dieser Eigenschaften für Schulen und Schüler besonders geeignet.



Eclipse mit Wiki-Plug-in

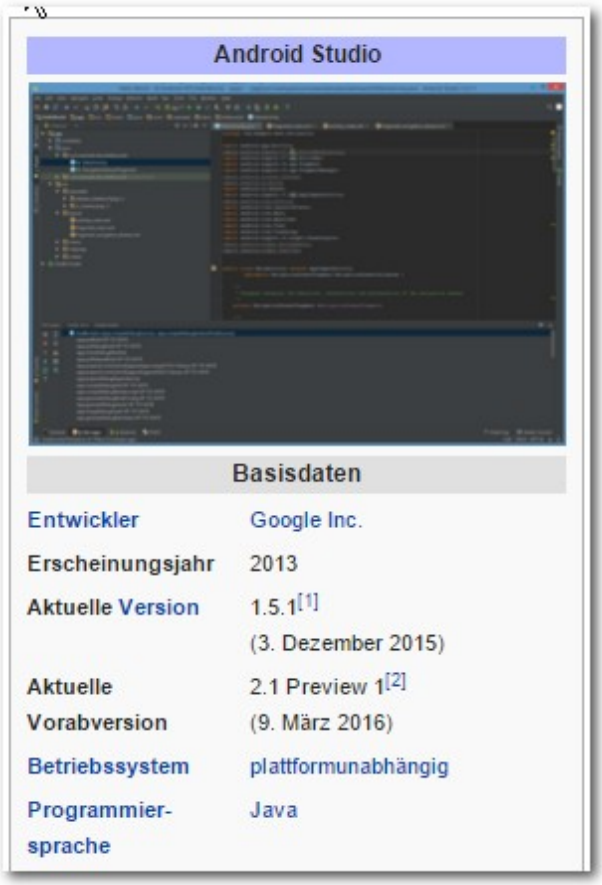
Basisdaten	
Entwickler	Eclipse Foundation
Aktuelle Version	4.5[1] (24. Juni 2015)
Betriebssystem	plattformunabhängig
Programmier-sprache	Java

https://de.wikipedia.org/wiki/Eclipse_(IDE)

Eclipse ist mit Sicherheit eine der Entwicklungsumgebungen, welche in der Softwarentwicklung am häufigsten zum Einsatz kommt.

Ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art. Ursprünglich wurde Eclipse als integrierte Entwicklungsumgebung (IDE) für die Programmiersprache Java genutzt, aber mittlerweile wird sie wegen seiner Erweiterbarkeit auch für viele andere Entwicklungsaufgaben eingesetzt. Für Eclipse gibt es eine Vielzahl sowohl quelloffener als auch kommerzieller Erweiterungen.

Die von Soyatec zur Verfügung gestellte Erweiterung eUML2 wird in der freien Variante nur bis zur Version 3.7 (Indigo) von Soyatec bereitgestellt.



https://de.wikipedia.org/wiki/Android_Studio

Das Android Studio ist die offizielle Entwicklungsumgebung für die Softwareentwicklung von Anwendungen für mobile Endgeräte mit einem Android Betriebssystem.

Android Studio ist eine freie, integrierte Entwicklungsumgebung von Google und offizielle die Entwicklungsumgebung für Android.

Thema:	Entwicklungsumgebungen in der objektorientierten Programmierung Urquelle: Christine Janischek Überblick: Eine Entwicklungsumgebungen installieren

Auf den Entwicklerseiten finden Sie i.d.R. Download-Bereiche in denen die neusten Versionen für unterschiedliche Betriebssysteme bereitgestellt werden.

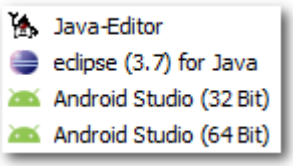
Ein Java-Entwickler benötigt außer der Entwicklungsumgebung (egal welche) das Java Runtime Environement (JRE) und den Java Developement Kit (JDK). Beide Softwarekomponenten werden aktuell von Oracle bereitgestellt. Unter dem Begriff Java SE finden Sie die benötigten Komponenten.

Which Java package do I need?

- **Software Developers: JDK** (Java SE Development Kit). For Java Developers. Includes a complete JRE plus tools for developing, debugging, and monitoring Java applications.
- **Administrators running applications on a server: Server JRE** (Server Java Runtime Environment) For deploying Java applications on servers. Includes tools for JVM monitoring and tools commonly required for server applications, but does not include browser integration (the Java plug-in), auto-update, nor an installer. Learn more
- **End user running Java on a desktop: JRE:** (Java Runtime Environment). Covers most end-users needs. Contains everything required to run Java applications on your system.

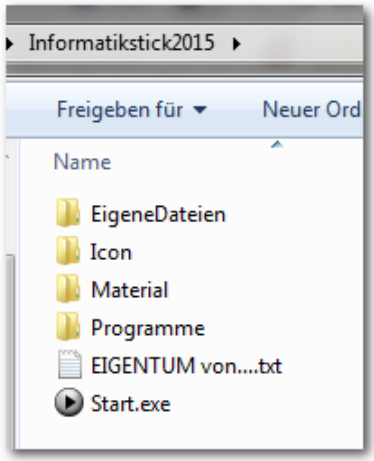
→ http://www.oracle.com/technetwork/articles/javase/index-jsp-138363.html

Jede der genannten Entwicklungsumgebungen ist zwischenzeitlich auch portable, kann also von auf einem externen Datenträger installiert werden, der Trick ist dabei, die Pfade zu den benötigten zusätzlichen Softwarekomponenten, wie u.a. der JDK anzupassen. Das Android Studio nutzt Profileinstellungen und Systempfade und kann deshalb nur bedingt portabel eingesetzt werden.



Arbeitsauftrag:

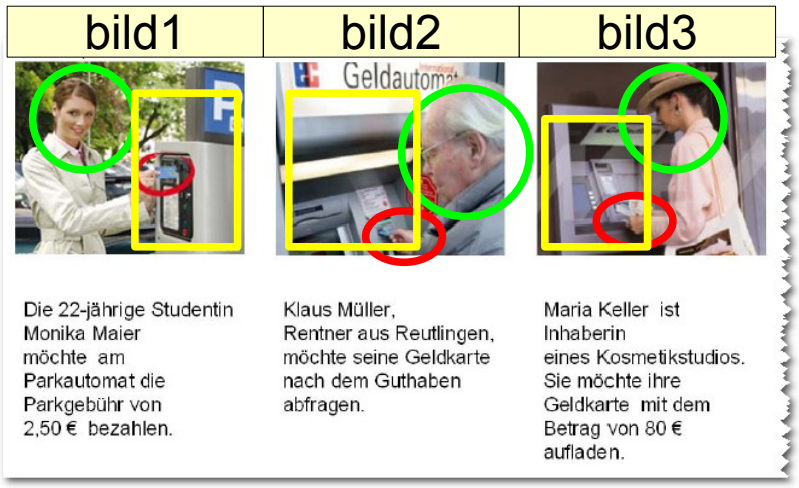
1. Nutzen Sie den Informatikstick.
2. Öffnen Sie die Entwicklungsumgebung.
3. Klären Sie die Begriffe JDK und JRE.
4. Welchen Zweck erfüllen diese beiden Softwarekomponenten?



2 Objekte und Klassen

Objekte und Klassen

Thema:	Einführung in die objektorientierte Programmierung in Java 1 Urquelle: HR- Oberwies Christoph. Informatik an beruflichen Gymnasien - Jahrgangsstufe 1. Landesinstitut für Schulentwicklung, (01), 2008 Objekte und Klassen



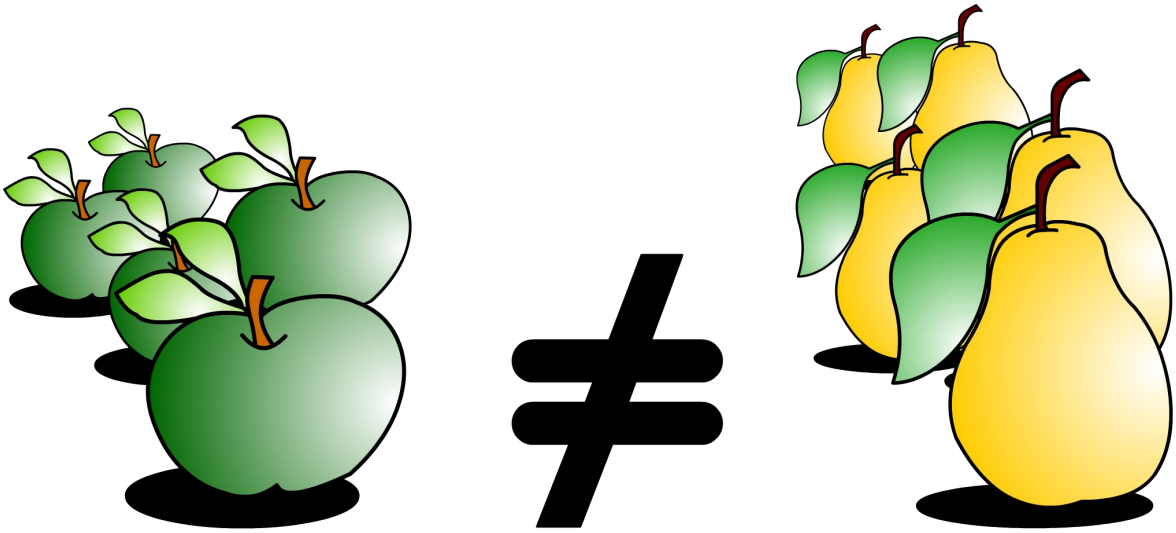
1 Geldkarte

Arbeitsauftrag:

1. Identifizieren Sie im ersten Schritt die Objekte auf den Bildern.
2. Identifizieren Sie alle im zweiten Schritt die zugehörigen Klassen.
3. Identifizieren Sie alle Attribute und Datentypen. Ordnen Sie diese den entsprechenden Klassen zu.
4. Identifizieren Sie alle Methoden (Verhaltensweisen) und ordnen Sie diese den Klassen zu.

Benutzeroberfläche (GUI): Geldkarte

Thema:	Objektorientierung Urquelle: Christine Janischek Übung: Wahr oder Falsch



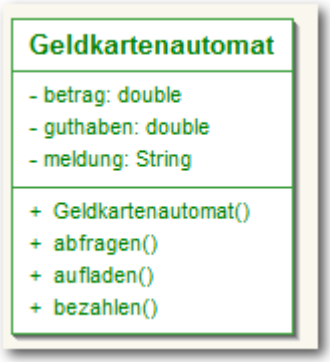
Kennzeichnen Sie alle **wahren** Aussagen.

	Attributnamen werden in der Regel kleingeschrieben.
	Jedes Attribut repräsentiert eine Eigenschaft von Objekten einer Klasse.
	Der Zustand eines Objektes ist durch seine Werte bestimmt.
	Die Set-Methode (Setter) übermittelt einen Attributwert an das Klassenattribut des aktuellen Objektes einer Klasse.
	Klassennamen werden grundsätzlich großgeschrieben und stehen im Plural (Mehrzahl).
	Methoden mit Rückgabewert sind vom Typ „void“.
	Von einer Klasse kann man genau ein Objekte erzeugen.
	Der Dateiname einer Klasse muss mit dem Klassennamen übereinstimmen.
	int, String, double und boolean sind primitive Datentypen.
	Methoden werden in der Regel mit dem Zugriffsmodifikator „private“ deklariert.
	Der Konstruktor einer Klasse entspricht nicht dem Klassenname.
	Methoden sind Verhaltensweisen (Adjektive, Verben) und werden in objektorientierten Sprachen am Anfang großgeschrieben.
	Der Konstruktor einer Klasse verwendet den Zugriffsmodifikator „private“.
	Methoden beinhalten den Quellcode für Verhaltensweisen von Objekten einer Klasse.

3 Grundgerüst einer Klasse

Grundgerüst einer Klasse

Thema:	Grundgerüst einer Klasse Urquelle: Christine Janischek Übung: Die Fachklasse Geldkartenautomat

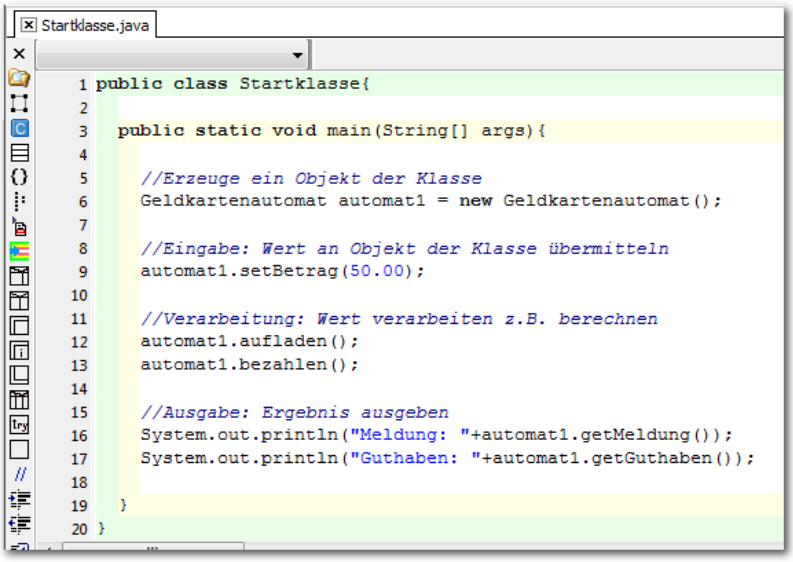


UML-Klasse: Fachklasse Geldkartenautomat.java

Benutzeroberfläche: Hauptfenster.java

UML-Klasse: Startklasse mit *Main-Methode

* Main-Methode dient als Startpunkt für einer Java Anwendung. Im vorliegenden Fall wird Sie in die Start-klasse eingebettet. Hier dient Sie vorübergehend zum testen der Anwendung. Später wird die Startklasse durch die Benutzeroberfläche ersetzt.

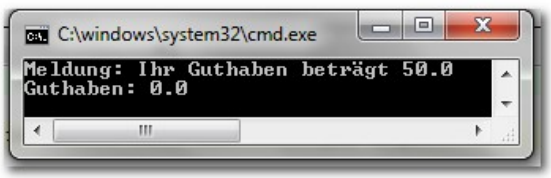


Testklasse: Startklasse.java

Mit der Benutzeroberfläche beschäftigen wir uns zu einem späteren Zeitpunkt!

Arbeitsauftrag:

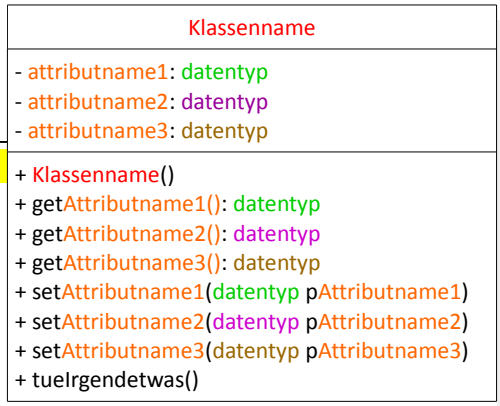
1. Erzeugen Sie den Quellcode für das Grundgerüst der Fachklasse in Java.
2. Erzeugen Sie den Quellcode für die Testklasse in Java.
3. Testen und Dokumentieren Sie die Ergebnisse.



Thema:	Grundgerüst einer Klasse Urquelle: Christine Janischek Informationsblatt: Attribute, Datentypen, Konstruktoren und Methoden

Die Klasse ist ein Muster, eine Vorlage die eine ganze Menge an Objekten mit ihren Eigenschaften und Verhaltensweisen beschreibt (definiert = deklariert). In Java steht der Quellcode einer Klasse in einer eigenen Datei. Der Dateiname einer Klasse muss dabei zwingender Weise identisch sein mit dem Klassennamen!

Arbeitsauftrag:
Klären und dokumentieren Sie alle Begriffe!



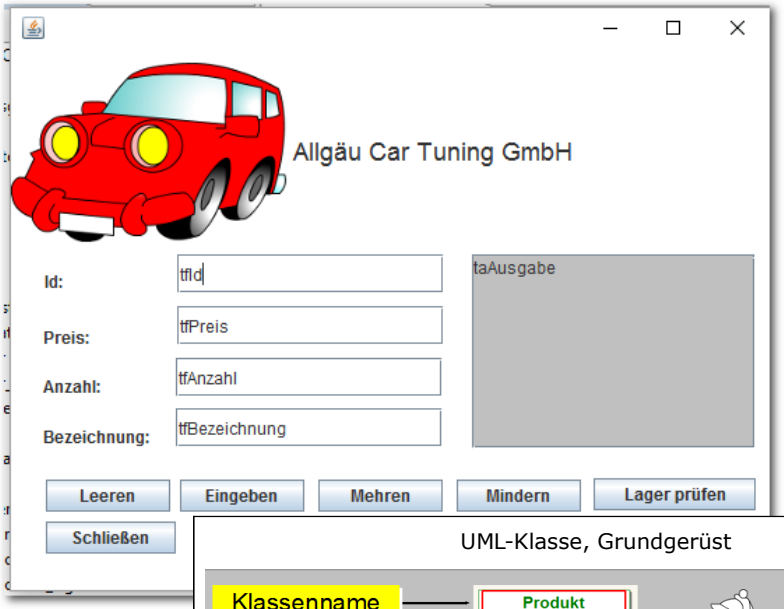
UML-Klasse

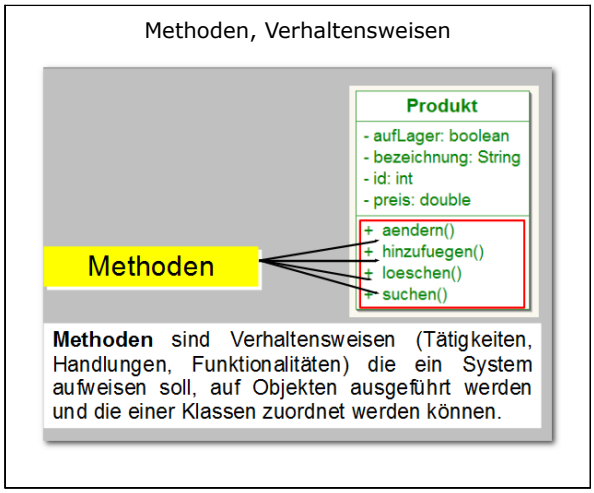
```
public class Klassenname {

// Deklaration der Eigenschaften (Attribute)
    private datentyp attributname1;
    private datentyp attributname2;
    private datentyp attributname3;

// Standard (Default) Konstruktor
    public Klassenname() {
    }

// Getter-Methoden: Ermittelt Eigenschaftswert eines Objektes
    public datentyp getAttributname1 () {
        return this.attributname1;
    }
    public datentyp getAttributname2() {
        return this.attributname2;
    }
    public datentyp getAttributname3() {
        return this.attributname3;
    }

// Setter-Methoden: Übermittelt Eigenschaftswert an das Attribut des Objektes
    public void setAttributname1( datentyp pAttributname1) {
        this.attributname1 = pAttributname1;
    }
    public void setAttributname2( datentyp pAttributname2) {
        this.attributname2 = pAttributname2;
    }
    public void setAttributname3( datentyp pAttributname3) {
        this.attributname3 = pAttributname3;
    }

// Sonstige Methoden: Methoden die mehr können als nur er- und übermitteln
    public void tueIrgendetwas(){
            // Sonstige Methoden
    }

}
```

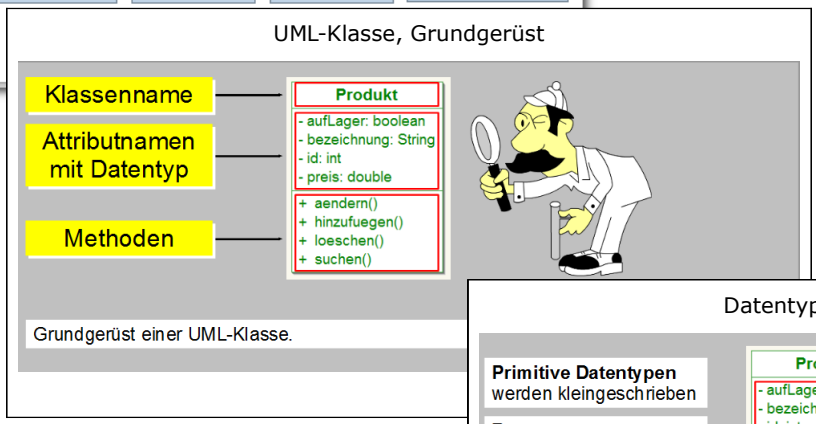
Thema:	Grundgerüst einer Klasse Urquelle: Christine Janischek Die Fachklasse: Information und Übung

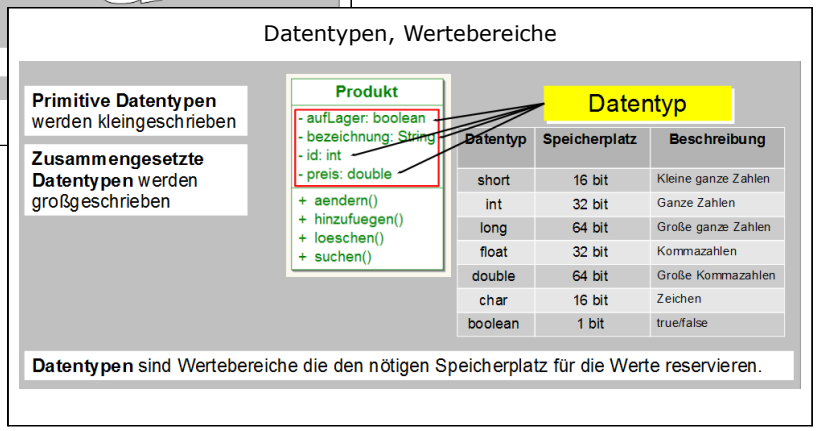
Systeme dienen einem Zweck!

Der Zweck eines Systems begründet die Deklaration (Definition) von Eigenschaften und Verhaltensweisen. Die Objektorientierung stellt sicher, dass die Definitionen in der Klasse erfolgen dessen Objekte die Eigenschaft oder Verhaltensweise später aufweisen sollen.









Arbeitsauftrag:

1. Erzeugen Sie den Quellcode für das Grundgerüst der Fachklasse → Produkt.java. Erweitern Sie die Fachklasse, sodass Sie den Anforderungen der Benutzeroberfläche (GUI) entspricht.
2. Klären Sie den Begriff Wrapper-Klasse.
3. Dokumentieren Sie Ihre Kenntnisse zum Umgang mit Datentypen in Java.

Thema:	Grundgerüst einer Klasse Urquelle: Christine Janischek Informationsblatt: Datentypen in Java

In Java werden Datentypen in zwei Kategorien unterteilt, in die primitive (einfache) Datentypen und in die Referenztypen (Klassentypen, zusammengesetzte bzw. erweiterte Datentypen). Einfache Datentypen sind in eine Programmiersprache eingebaut. Die Referenztypen sind Objekte einer zugehörigen Klasse (z.B. String, Double, Integer) die mächtiger sind, d. h. Sie sind um viele nützliche Verhaltensweisen (Methoden) erweitert.

Die Programmiersprache Smalltalk kennt beispielsweise gar keine einfachen (primitiven) Datentypen, sondern organisiert alles über Objekte.

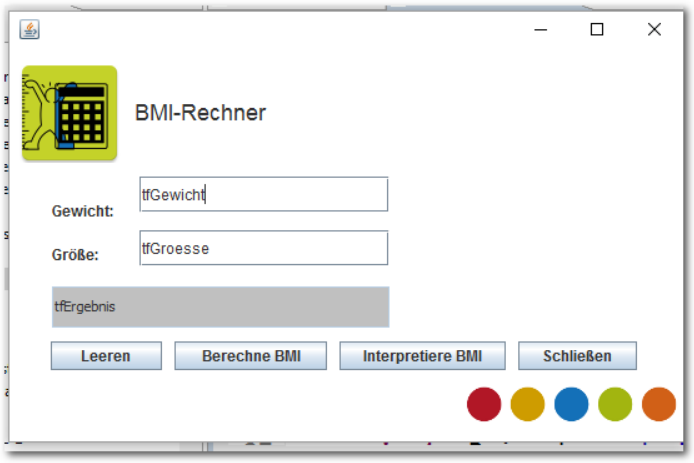
Soll nicht nur ein einzelnes Zeichen gespeichert werden, sondern eine Zeichenkette, muss der Referenztyp >>String<< verwendet werden. Dieser ist so konstruiert, dass er genauso wie ein einfacher Datentyp gehandhabt werden kann.

Für ein Erfassen von Kalenderdaten, insbesondere in Anlehnung an den SQL-Datentypen Date, existiert eine Klasse Date für Java. Diese muss aber erst eingebunden werden und lässt sich nicht so einfach handhaben wie String. In der Praxis wird deshalb häufig der Datentyp >>String<< verwendet um ein Datum zu verarbeiten bzw. zu speichern. *Recherchieren Sie selbst!*

Datentyp	**Speicherplatz** in Byte	**Wertebereich** von … bis	**default** Standardwert	**Bemerkung**
byte	1 byte	Von -128 bis 127	0	Maßeinheit der Digitaltechnik (8 Bit = 1 byte)
short	2 byte	Von -32768 bis 32767	0	(kleine) Ganze Zahlen
int	4 byte	Von -2147483648 bis 2147483647	0	(mittel) Ganze Zahlen
long	8 byte	Von - 9223372036854775808 Bis 9223372036854775807	0l	(große) Ganze Zahlen

float	4 byte	Von -3.40282347E+38 Bis 3.40282347E+38	0.0f	(einfache G.) Kommazahl
double	8 byte	Von - 1.7976931348623 1570E+308 Bis 1.7976931348623 1570E+308	0.0f	(doppelte G.) Kommazahl
char	2 byte	Von \u0000 bis \uffff	\u0000	einzelnes Zeichen
boolean	1 bit	true false	false	Zweiwertiges Attribut

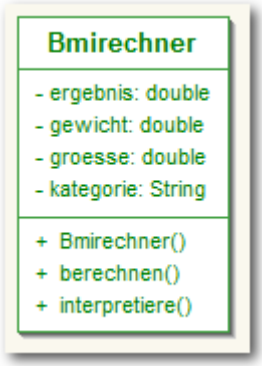
Thema:	Grundgerüst einer Klasse Urquelle: Christine Janischek Übung: Die Fachklasse Bmirechner



Benutzeroberfläche (Swing): Hauptfenster.java

Benutzeroberfläche eines Mobilen Endgeräts: activity_main.xml

```
bmi =  gewicht / (groesse*groesse);
```



UML-Klasse: Fachklasse Bmirechner.java

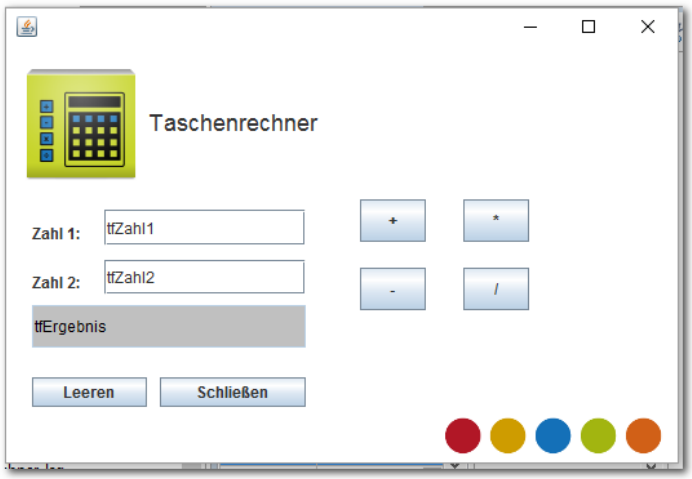
Arbeitsauftrag:

1. Erzeugen Sie ein Projektverzeichnis → Bmirechner
2. Erzeugen Sie darin eine Datei für den Quellcode, das Grundgerüst der Fachklasse in Java.
3. Dokumentieren Sie die Ergebnisse.

Zusatzaufgabe:

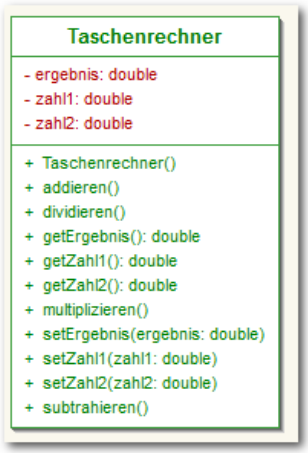
Erzeugen Sie den Quellcode für eine geeignete Startklasse mit Main-Methode. Testen Sie mit einer geeigneten Entwicklungsumgebung.

Thema:	Grundgerüst einer Klasse Urquelle: Christine Janischek Übung: Fachklasse Taschenrechner



Benutzeroberfläche: Hauptfenster.java

Benutzeroberfläche eines Mobilen Endgeräts: activity_main.xml



UML-Klasse: Fachklasse Taschenrechner.java

Mit der Benutzeroberfläche beschäftigen wir uns zu einem späteren Zeitpunkt!

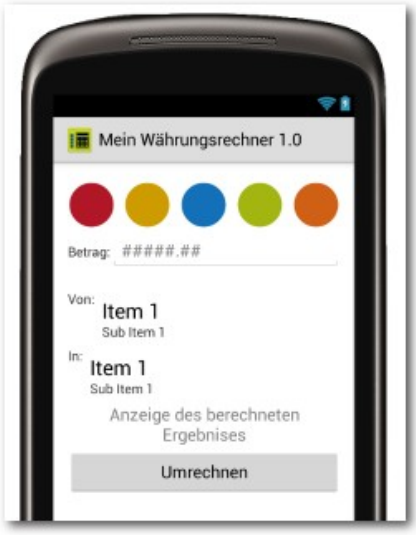
Arbeitsauftrag:

1. Erzeugen Sie ein Projektverzeichnis → Taschenrechner
2. Erzeugen Sie darin eine Datei für den Quellcode, das Grundgerüst der Fachklasse in Java.
3. Dokumentieren Sie die Ergebnisse.

Zusatzaufgabe:
Erzeugen Sie den Quellcode für eine geeignete Startklasse mit Main-Methode. Testen Sie mit einer geeigneten Entwicklungsumgebung.

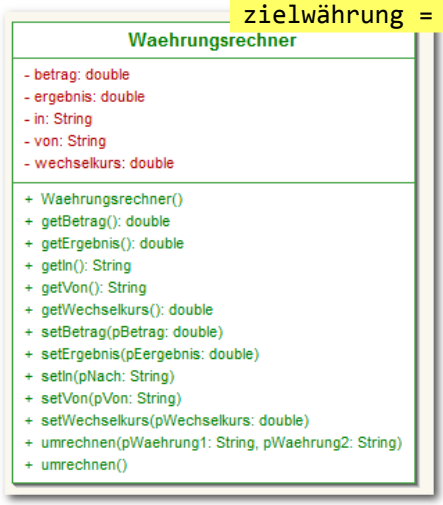
Thema:	Grundgerüst einer Klasse Urquelle: Christine Janischek Übung: Fachklasse Währungsrechner

Benutzeroberfläche: Hauptfenster.java



Benutzeroberfläche: activity_main.xml

```
zielwährung =  ausgangswährung * wechselkurs;
```



UML-Klasse: Fachklasse Waehrungsrechner.java

Mit der Benutzeroberfläche beschäftigen wir uns zu einem späteren Zeitpunkt!

Arbeitsauftrag:

1. Erzeugen Sie ein Projektverzeichnis → Waehrungsrechner
2. Erzeugen Sie darin eine Datei für den Quellcode, das Grundgerüst der Fachklasse in Java.
3. Dokumentieren Sie die Ergebnisse.

Zusatzaufgabe:

Erzeugen Sie den Quellcode für eine geeignete Startklasse mit Main-Methode. Testen Sie mit einer geeigneten Entwicklungsumgebung.

4 Methoden

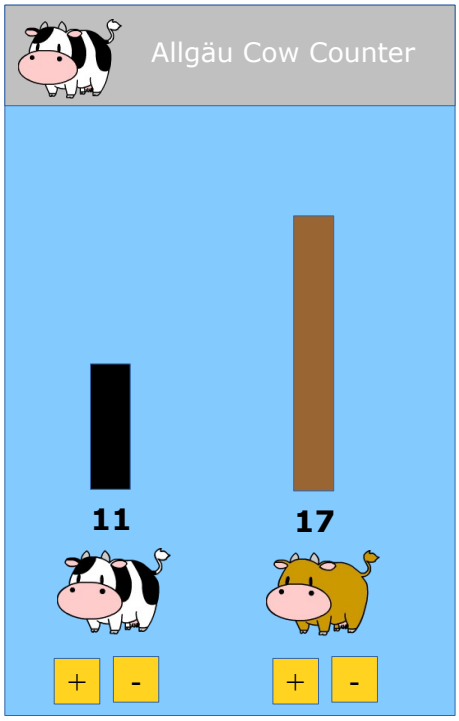
Methoden

Thema:	Methoden (Verhaltensweisen) Urquelle: Christine Janischek Übung: Allgäu Cow Counter (Inkrementieren/Dekrementieren)

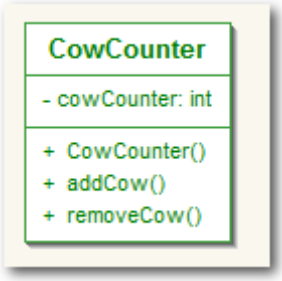
Die Firma Allgäu Cows GmbH möchte eine App entwickeln die es für Ihre Landwirte möglich macht Kühe zu zählen. Regelmäßig soll dazu das Fleckvieh (Schwarz-Weiße Kühe) getrennt von den braunen Kühen gezählt werden können.

Arbeitsauftrag:

1. Erzeugen Sie ein Projektverzeichnis → CowCounter
2. Erzeugen Sie dann die Fachklasse.
3. Implementieren Sie den Quellcode für die Methode *addCow()* und testen Sie welche der genannten Methoden den Zähler erfolgreich um 1 erhöht.



Storyboard: Allgäu Cow Counter



UML-Klassendiagramm

Zusatzaufgabe:
Implementieren Sie den Quellcode für die Methode *removeCow()*. Für den Fall, dass der Zähler in den Minusbereich „rutscht" soll der Zähler auf 0 gesetzt werden, ansonsten soll der Zähler dekrementiert (um 1 dezimiert werden) werden.

```
/*Erhöht (inkrementiert)
 * den CowCounter um 1*/
public void addCow(){
    cowCounter++;
}
```

Variante 1

```
/*Erhöht (inkrementiert)
 * den CowCounter um 1*/
public void addCow(){
    cowCounter = cowCounter + 1;
}
```

Variante 2

```
/*Erhöht (inkrementiert)
 * den CowCounter um 1*/
public void addCow(){
    cowCounter += 1;
}
```

Variante 3

Thema:	Methoden (Verhaltensweisen) Urquelle: Christine Janischek Übung: Syntaxfehler

Identifizieren Sie die Syntaxfehler:

Nr.	Typ	r oder f
1.	`private int[][] keys = new int[4][5000];`	

2.	`public String entferne_(){`
	`String mWort = "Hallo_meine_lieben_Freunde";`
	`mWort = mWort.replace("_", " ");`
	`return mWort;`
	`}`
	
	Ausgabe: Hallo meine lieben Freunde
	Richtig?	

3.	`private void bestimme_5_Buchstaben(){`
	`String mWort = "Hallo ";`
	`for(int i = 0; i < mWort.length(); i++){`
	`switch(i){`
	`case 0:`
	`this.setBuchstabe1(mWort.charAt(i));`
	`break;`
	`case 1:`
	`this.setBuchstabe2(mWort.charAt(i));`
	`break;`
	`case 2:`
	`this.setBuchstabe3(mWort.charAt(i));`
	`break;`
	`case 3:`
	`this.setBuchstabe4(mWort.charAt(i));`
	`break;`
	`case 4:`
	`this.setBuchstabe5(mWort.charAt(i));`
	`break;`
	`default:`
	`break;`
	`}`
	`}`
	`}`	

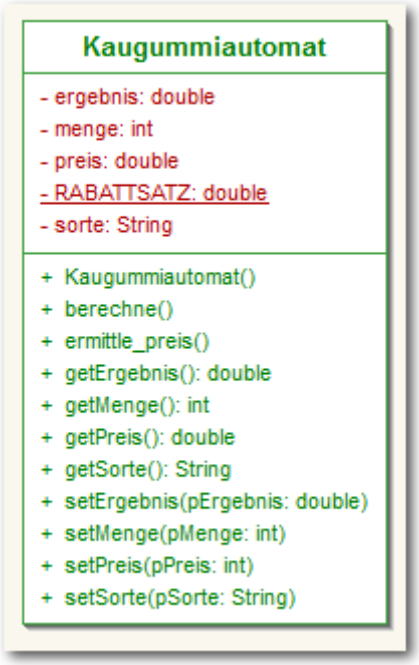
4.	`public double setUmsatz(double pUmsatz){`
	`this.umsatz = pUmsatz;`
	`}`	

5.	`private Artikel artikeldaten = new Artikel();`	

5 Kontrollstrukturen

Kontrollstrukturen

Thema:	Methoden (Verhaltensweisen) Urquelle: Christine Janischek Übung: Kaugummiautomat, Fallunterscheidungen



Arbeitsauftrag:

1. Erzeugen Sie ein Projektverzeichnis → Kaugummiautomat
2. Erzeugen Sie dann die Fachklasse.
3. Implementieren Sie den Quellcode für die Methode *ermittle_preis()* und *berechne()*.

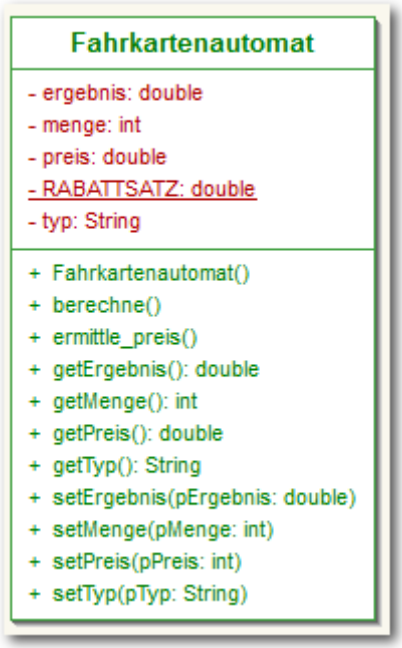
Zusatzinformation:

Sorte	Preis in €
Erdbeere	0.13
Zitrone	0.15
Orange	0.23
Vanille	0.25

Menge	Rabatt in %
kleiner gleich 10 Stück	0.00
größer 10 Stück	5.00

Berechnungsbeispiel:

Thema:	Methoden (Verhaltensweisen) Urquelle: Christine Janischek Übung: Fahrkartenautomat, Fallunterscheidungen



Arbeitsauftrag:

1. Erzeugen Sie ein Projektverzeichnis → Fahrkartenautomat
2. Erzeugen Sie dann die Fachklasse.
3. Implementieren Sie den Quellcode für die Methode *ermittle_preis()* und *berechne()*.

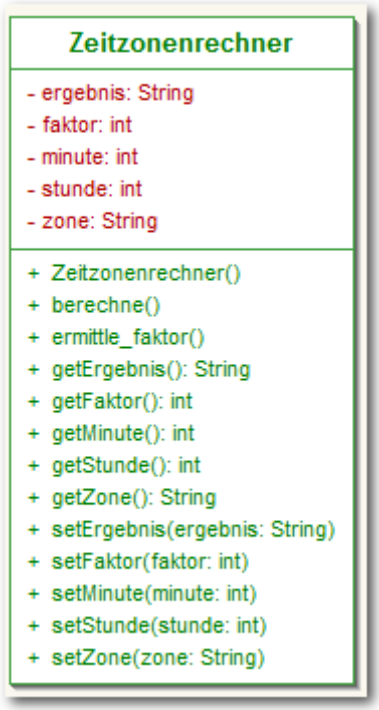
Zusatzinformation:

Sorte	**Preis in €**
Zone A	2.30
Zone AB	4.20
Zone ABC	6.50
Zone ABCD	8.00

Menge	**Rabatt in %**
kleiner 5 Stück	0.00
mindestens 5 Stück	4.00

Berechnungsbeispiel:

Thema:	Methoden (Verhaltensweisen) Urquelle: Christine Janischek Übung: Zeitzonenrechner, Fallunterscheidungen



Arbeitsauftrag:

1. Erzeugen Sie ein Projektverzeichnis → Kaugummiautomat
2. Erzeugen Sie dann die Fachklasse.
3. Implementieren Sie den Quellcode für die Methode *ermittle_faktor()* und *berechne()*.

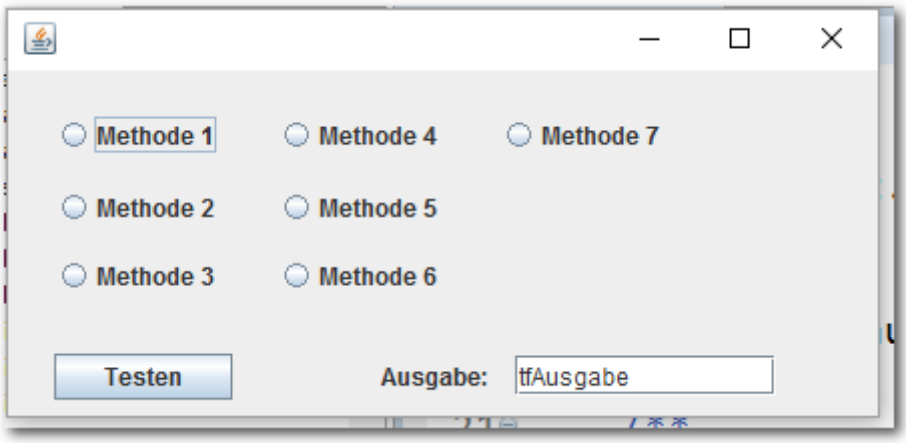
Zusatzinformation:

Zone	Faktor
New York	-6
London	-1
Tokio	+8
Sydney	+10

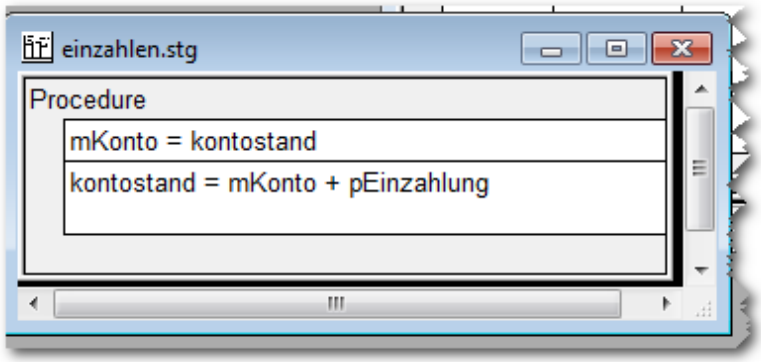
Für	ergebnis
(stunde + faktor) < 24	stunde + faktor
(stunde + faktor) >= 24	stunde + faktor - 24

Berechnungsbeispiel:

Thema:	Kontrollstrukturen Urquelle: Christine Janischek Übung: Methodentester - Methoden und Fallunterscheidungen



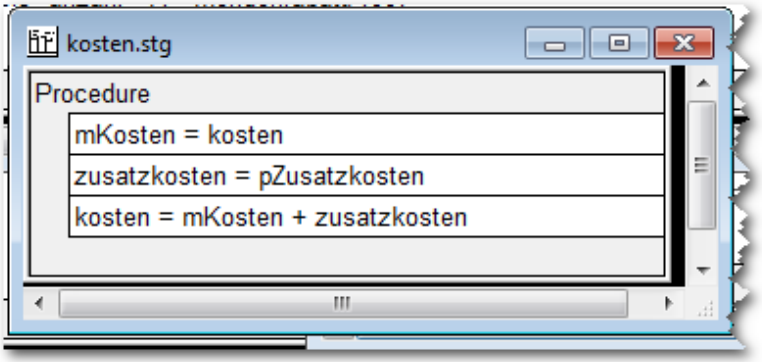
Programmieren Sie zunächst händisch auf Papier!



Die Einzahlung auf einem Konto soll ermöglicht werden.

Implementierung:
Schreiben Sie die JAVA-Anweisungen in eine Methode → einzahlen der Klasse Konto.

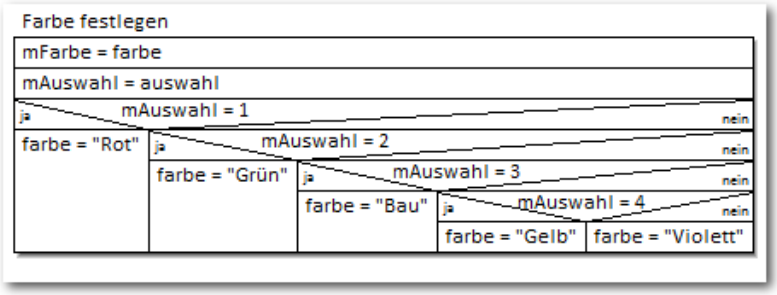
Erstellen Sie 2 Varianten:
1.V. → mit Rückgabewert
2.V → ohne Rückgabewert



Die Kosten für ein Haus ist mit Zusatzkosten verbunden.

Implementierung:
Schreiben Sie die JAVA-Anweisungen in eine Methode → berechne Kosten der Klasse Haus.

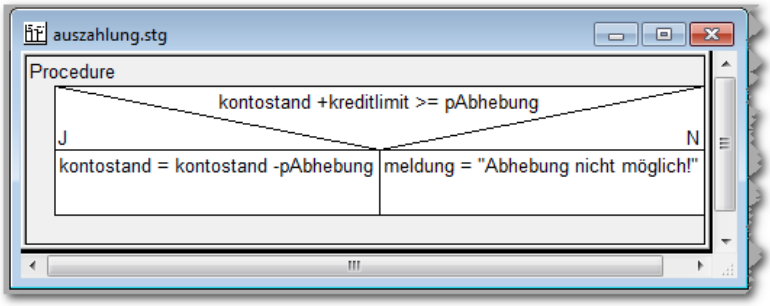
Erstellen Sie 2 Varianten:
1.V. → mit Parameter und mit Rückgabewert
2.V → mit Parameter und ohne Rückgabewert



Die Farbe soll für Grafiken individuell bestimmbar sein.

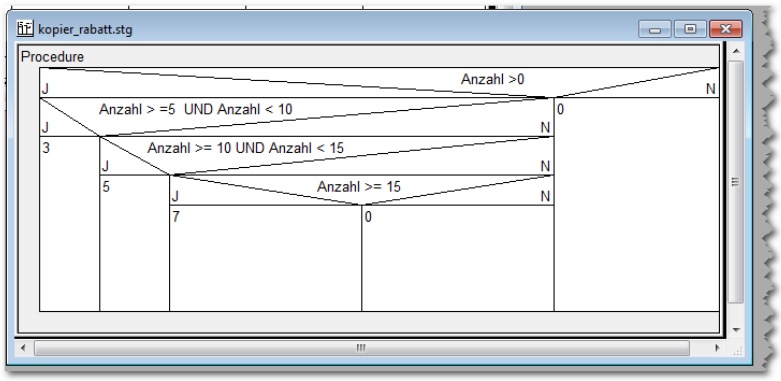
Implementierung:
Schreiben Sie die Methode → Farbe festlegen der Klasse Grafik. Implementieren Sie ergänzend die ganze Klasse und schreiben Sie für den UNIT Test eine entsprechende Startklasse.

Thema:	Kontrollstrukturen Urquelle: Christine Janischek Übung: Methodentester - Methoden und Fallunterscheidungen (Forts.)



Die Auszahlung auf einem Konto soll nur nach den genannten Bedingung erfolgen.

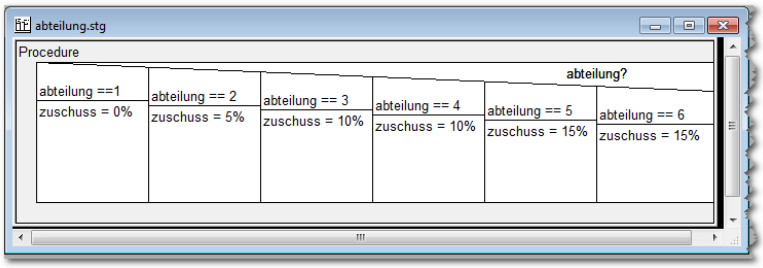
Implementierung:
Schreiben Sie die JAVA-Anweisungen (Quellcode) in eine Methode → auszahlen der Klasse Konto.



Ein Copy-Shop gibt Dir als Kunde Rabatt für Farbkopien.

Der Rabatt ist abhängig von der Anzahl der Kopien und soll nur nach den genannten Bedingung erfolgen.

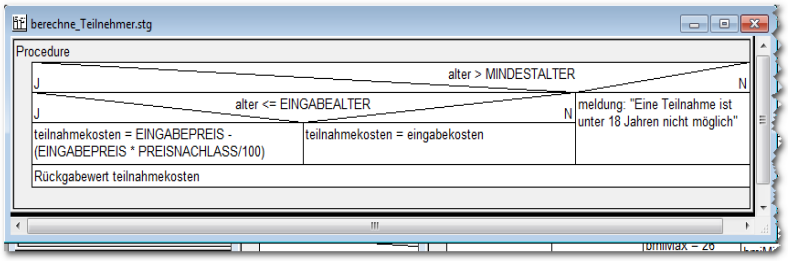
Implementierung:
Schreiben Sie die JAVA-Anweisungen in eine Methode → bestimme Rabatt der Klasse Farbkopie.



Der Zuschuss reduziert die Kosten für die Fortbildung die ein Mitarbeiter einer bestimmten Abteilung selbst bezahlen muss.

Erzeugen Sie ein Klassendiagramm. Bestimmen Sie Attribute, Assoziationen und Methoden.

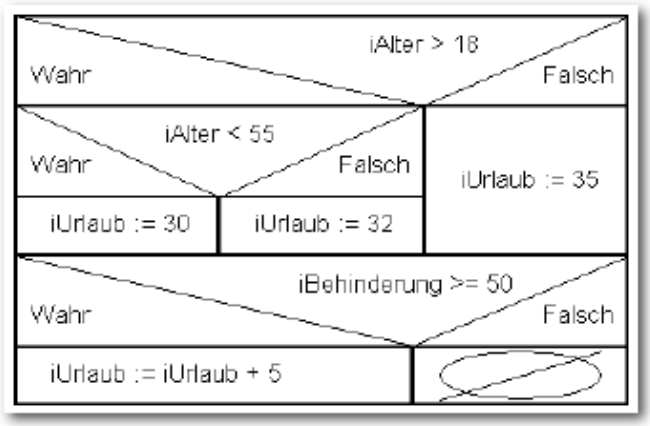
Implementierung:
Schreiben Sie die JAVA-Anweisungen (Quellcode) in eine Methode → bestimme Zuschuss der Klasse Fortbildung.



Teilnehmer einer Veranstaltung erhalten unter den genannten Bedingungen einen Preisnachlass.

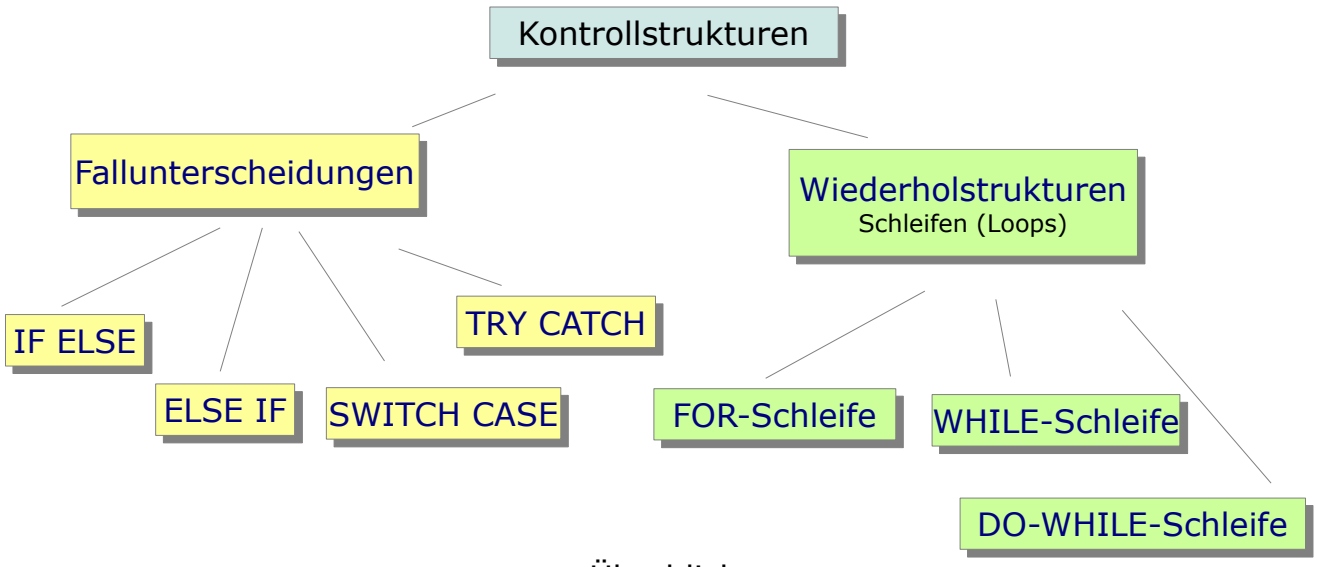
Implementierung:
Schreiben Sie die JAVA-Anweisungen (Quellcode) in eine Methode → berechne Teilnahme der Klasse Teilnehmer.

Thema:	Methoden (Verhaltensweisen) Urquelle: Christine Janischek Übung: Fallunterscheidungen



Erzeugen Sie für die Klasse Urlaubsrechner eine Methode `ermittleUrlaubstage(int pAlter)`. Nutzen Sie dazu die Vorgaben aus dem Struktogramm.

Thema:	Kontrollstrukturen Urquelle: Christine Janischek Informationsblatt: Kontrollstrukturen (Überblick)



Überblick

Hinweis: Fallunterscheidungen sind Bestandteil von Verhaltensweisen (Methoden)

Fallunterscheidungen

IF ELSE

```
if(){

}else{

}
```

```
if (this.guthaben >= this.betrag) {
  this.guthaben = this.guthaben - this.betrag;
  this.meldung = "Ihr neues Guthaben beträgt "+ this.guthaben;
} else{
  this.meldung = "Zu geringes Guthaben!";
}
```

ELSE IF

```
if(){

}else if(){

}else if(){

}else{

}
```

```
if(pKundenart == 1){
  this.setKundenart(pKundenart);
  return "ok";
}else if(pKundenart == 2){
  this.setKundenart(pKundenart);
  return "ok";
}else if(pKundenart == 3){
  this.setKundenart(pKundenart);
  return "ok";
}else{
  this.setKundenart(0);
  return "keiner";
}
```

SWITCH CASE

```
switch () {
      case 1:
      break;

      case 2:
      break;

      case 3:
      break;

      default:

    }
```

```
switch (this.kundenart) {
  case 1:
  return "ok";

  case 2:
  return "ok";

  case 3:
  return "ok";

  default:
  return "keiner";
}
```

* wenn es sich um eine Methode mit Rückgabewert handelt können die „breaks" weggelassen werden.

Begründung:

Mit dem → break verlässt man die Kontrollstruktur. Da mit dem → return derselbe Effekt erreicht wird müssen wir auf den break verzichten.

Besonderheiten von SWITCH CASE:

- Übersichtlich
- Funktioniert nur bei Prüfung primitiver Datentypen
- Ein Attribut für die Prüfung in der SWTICH Bedingung, dessen Zustand (Wert) geprüft wird.

TRY CATCH

```
try{
      Anweisung;
}
```

RuntimeException sind Laufzeitfehler die nicht zwingender maßen abgefangen werden müssen. Sie werden aber auf jeden Fall von der JVM abgefangen, welche eine entsprechende Fehlermeldung auf der Konsole ausgibt.

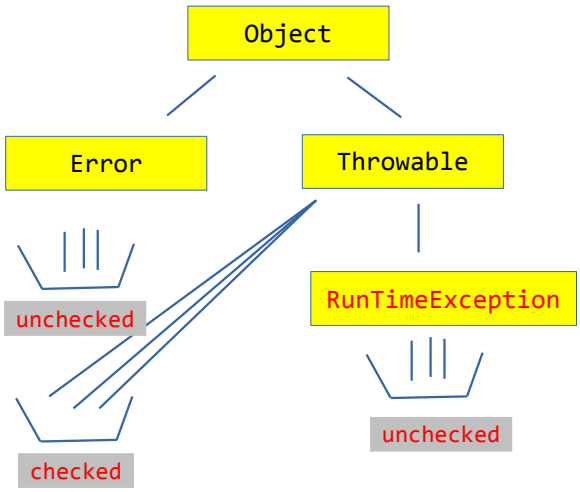
Die wohl am meisten auftretenden RuntimeExcepti-

```
try{
      Anweisung;
}catch(Exception e){
      Fehlerbehandlung;
}finally{
      Aufräumarbeiten;
}
```

```
try{
      Anweisung;
}catch(Exception e){
      Fehlerbehandlung;
}
```

```
try{
      Anweisung;
}finally{
      Aufräumarbeiten;
}
```

Hierarchie Ausnahmebehandlung in Java:



ons sind die:

- java.lang.NullPointerException und
- java.lang.ArrayIndexOutOfBoundsException.

Behandlung mit Try Catch

Soll eine Behandlung erfolgen muss bei mehreren sequentiellen catch-Blöcken vom Speziellen zum Allgemeinen abgefangen werden. Im anderen Falle würde unerreichbarer Code entstehen. Das gilt für alle Exceptions! Um über die Reihenfolge entscheiden zu können, müsste man die Vererbungshierarchie der Exceptions kennen. Nach einer try-catch-Anweisung kann optional eine finally-Anweisung stehen. Bei Try ohne Catch-Anweisung ist finally allerdings vorgeschrieben.
Die Anweisungen im Finally-Block werden grundsätzlich ausgeführt, unabhängig ob im try-Block eine Exception aufgetreten ist oder nicht. In dem finally-Block können demnach „Aufräumarbeiten programmiert werden, die in jedem Fall ausgeführt werden müssen.

```
- java.lang.RuntimeException
--- java.lang.ArithmeticException
--- java.lang.ArrayStoreException
--- java.lang.ClassCastException
--- java.lang.IllegalArgumentException
------ java.lang.IllegalThreadStateException
------ java.lang.NumberFormatException
--- java.lang.IllegalMonitorException
--- java.lang.IllegalStateException
--- java.lang.IndexOutOfBoundsException
------ java.lang.ArrayOutOfBoundsException
------ java.lang.StringOutOfBoundsException
--- java.lang.NegativeArraySizeException
--- java.lang.NullPointerException
--- java.lang.SecurityException
--- java.lang.UnsupportedOperationException
```

<table>
<tr><th colspan="3">Hinweis: Wiederholstrukturen sind Bestandteil von Verhaltensweisen (Methoden)</th></tr>
<tr><th>Wiederholstrukturen</th><th>Grundgerüst</th><th>Besonderheit</th></tr>
<tr><td>FOR-SCHLEIFE</td><td>

```
for (ausdruck1; ausdruck2; aus-
druck3){
      //Anweisung(en)
}
```

Beispiel:

```
for(int i = 10;i > 0;i--) {
      System.out.println(i);
}
```

</td><td>Ist die komplexeste Schleife.

Ausdruck1 wird vor der Schleife ausgeführt und enthält in der Regel die Deklaration und Initatialisierung der Schleifenzählervariablen.

Am Anfang eines jeden Schleifendurchlaufs wird die Anweisung in Ausdruck2 ausgeführt. Wenn diese → True ist wird die Schleife fortgesetzt und die darunterliegenden Anweisungen werden ausgeführt. Anderenfalls (→ False) wird der Vorgang abgebrochen.

Am Ende eines jeden Schleifendurchlaufs wird die Anweisung in Ausdruck3 ausgeführt.</td></tr>
<tr><td>WHILE-SCHLEIFE</td><td>

```
while (bedingung){
      //Anweisung(en)
      //Abbruchbedingung
}
```

Beispiel:

```
int i = 10;
while(i > 0) {
      System.out.println(i);
      i--;
}
```

</td><td>Ist die einfachste Schleife.

Die Anweisungen innerhalb der Schleife werden wiederholt ausgeführt, solange die Bedingung zutrifft, also zu → True evaluiert.

Eine Abbruchbedingung z.B. ein Zähler stellt in der Regel sicher, dass die Schleife nicht endlos ausgeführt wird, die Bedingung zu ge-</td></tr>
</table>

		gebener Zeit zu → False evaluiert und der Vorgang abgebrochen wird.

DO-WHILE-SCHLEIFE	```do{ //Anweisung(en) //Abbruchbedingung }while(bedingung);```
	
	 Beispiel:
	 ```int i = 10; do { System.out.println(i); i--; } while (i > 0);```
		Ist der While-Schleife sehr ähnlich.
		
		 Der Unterschied liegt daran, dass die Bedingung erst am Ende des Schleifendurchlaufs geprüft wird.

6 Stringmanipulation

Stringmanipulation

Thema:	Stringverarbeitung Urquelle: Christine Janischek Arbeitsblatt: Wortspiele - Wiederholstrukturen (Schleifen)

Arbeitsauftrag:

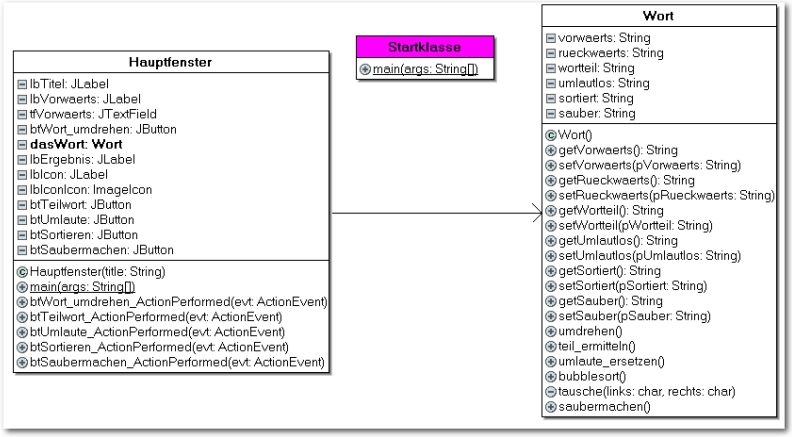
1. Erzeugen Sie ein neues Projekt → Wortspiel.
2. Implementieren Sie das Modell → Fachklasse Wort.
3. Erzeugen Sie die Interaktion mit der Benutzeroberfläche.

Zusatzaufgaben:

1. Erweitern Sie das Projekt Stück für Stück um die noch fehlenden Funktionalitäten.
2. Dokumentieren Sie alle Ergebnisse.

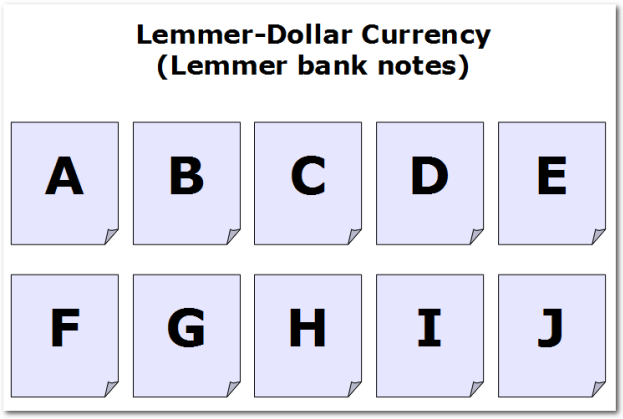
Hinweise:

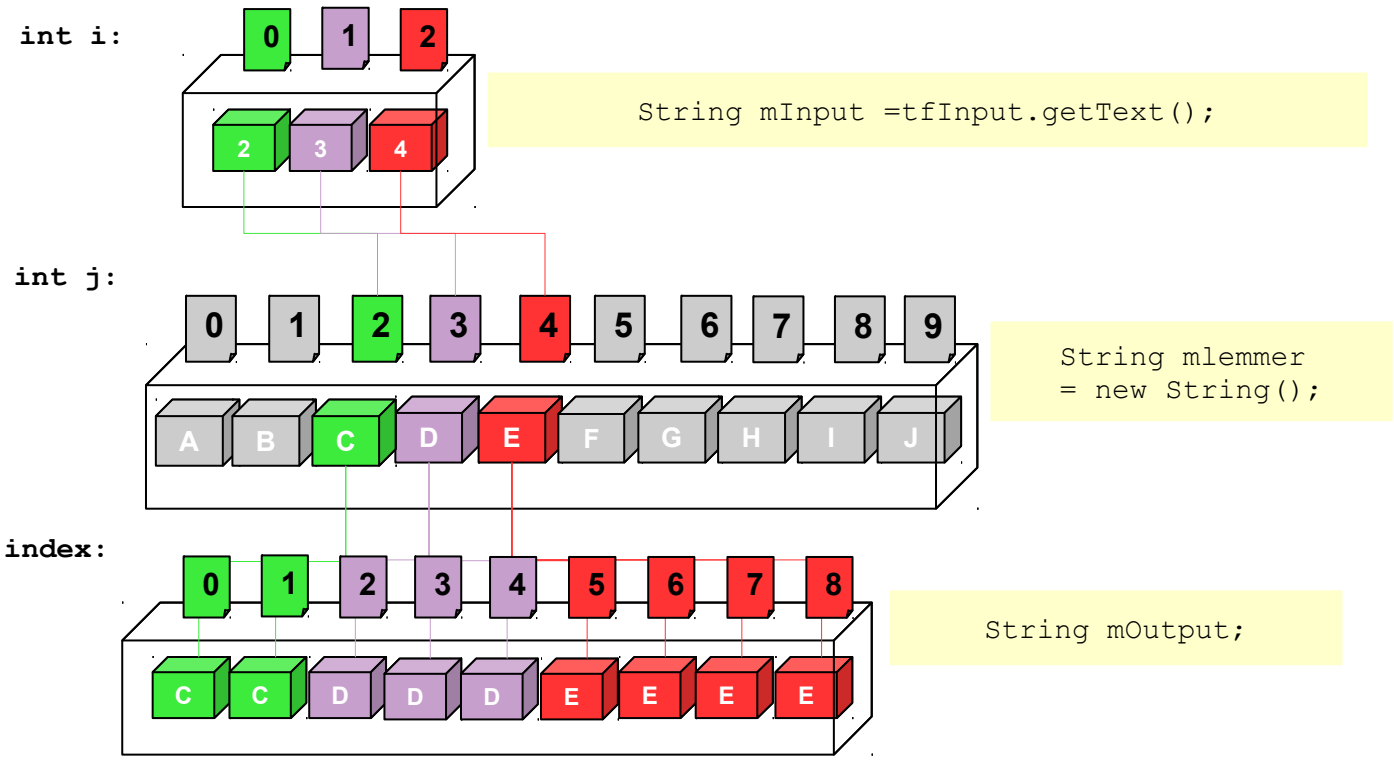
1. Die Methode das eingegebene Wort umdrehen
2. Die Methode soll die ersten drei Buchstaben in einem Wort ermitteln
3. Die Methode soll das Wort durchlaufen und Umlaute ermitteln und ersetzen
   - ü --> ue
   - ä --> ae
   - ö --> oe
   - ß --> ss
4. Sortieren von Buchstaben: Vergleichen und austauschen – Bubblesort
5. Die Methode soll einen String säubern: Dazu sollen # und Leerzeichen sollen aus einer Zeichenkette entfernt werden



UML-Klassendiagramm

Thema:	Kontrollstrukturen Urquelle: Hartmut Hug Erweiterte Übung: Stringverarbeitung, Wiederholstrukturen (Schleifen)





Thema:	Kontrollstrukturen Urquelle: oszinddv – 2009 Übung: Schleifen - Wiederholstrukturen

Arbeitsauftrag:
Testen Sie die dargestellten Programmausschnitte für die jeweils angegebenen Werte und notieren Sie die Anzahl der Schleifendurchläufe und die Ausgaben auf dem Bildschirm.

```
public void teste1(int startwert){
  int iZahl = startwert;
  int zaehler = 0;
  while (iZahl < 100){
    System.out.println(iZahl);
    iZahl = iZahl + 10;
    zaehler++;
  } // end of while
  System.out.println("Durchlauf: "+zaehler);
}
```

Startwert	Anzahl der Durchläufe	Bildschirmausgabe
0		
22		
100		

```
public void teste2(int startwert){
  int x = startwert;
  int zaehler = 0;
  do{
    int iQuadrat = x*x;
    System.out.println(iQuadrat);
    x= x+1;
    zaehler++;
  } while (x <=12);
  System.out.println("Durchlauf: "+zaehler);
}
```

Startwert	Anzahl der Durchläufe	Bildschirmausgabe
0		
10		
13		

```
public void teste3(int startwert){
  int iErgebnis = 0;
  int iWert = startwert;
  int zaehler = 0;

  while(iWert>= 100){
    iErgebnis = iWert*2;
    iWert = iWert -5;
    zaehler++;

  }
  System.out.println(iErgebnis);
  System.out.println("Durchlauf: "+zaehler);
}
```

Startwert	Anzahl der Durchläufe	Bildschirmausgabe
0		
100		
120		

```
public void teste4(int startwert){
  int y = startwert;
  int zaehler = 0;
  do{
    y = y+2;
    System.out.println(y);
    zaehler++;
  }while(y>3);

  System.out.println("Durchlauf: "+zaehler);
}
```

Startwert	Anzahl der Durchläufe	Bildschirmausgabe
0		
1		
2		

```
public void teste5(int start1,int start2){
  int x = start1;
  int y = start2;
  int zaehler1 = 0;
  int zaehler2 = 0;
  while (x <10){
    zaehler2++;
    // end of while
    do{
      System.out.println("x:" + x +"y:"+y);
      y = y+1;
      zaehler2++;
    }while(y<10);
    x = x+2;
  }
```

Start1	Start2	Anzahl der Durchläufe	Bildschirmausgabe
0	6		
1	10		
10	10		

Thema:	Kontrollstrukturen Urquelle: Hartmut Hug Übung: Die Klasse String (Zeichenketten manipulieren)

Erzeugen Sie eine Methode die aus der Zeichenkette die X-Zeichen und Leerzeichen entfernt und das Ergebnis auf der Konsole ausgibt. Nutzen Sie dazu die String-Methoden substring(), concat() und trim(). Dokumentieren Sie die Funktion der einzelnen Methoden.

String sprichwort:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
X	X	X	E	n	e	X	d	e	n	e	X	d	i	b	b	e	X	d	e	n	e	X			

[1]Beispiel

7 Unified Modelling Language (Vertiefung)

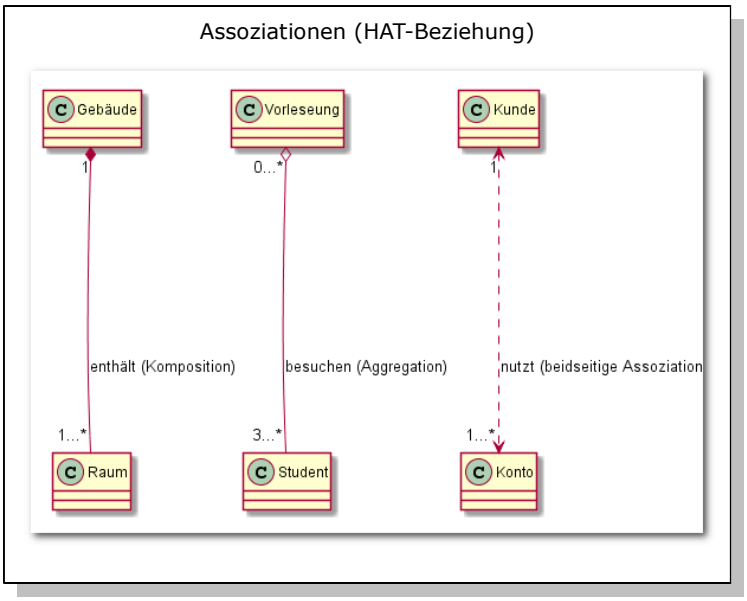
Unified Modelling Language
(Vertiefung)

Thema:	Unified Modelling Language Urquelle: Christine Janischek Übung zur Vertiefung Modellierung in UML: Klassen und Beziehungen

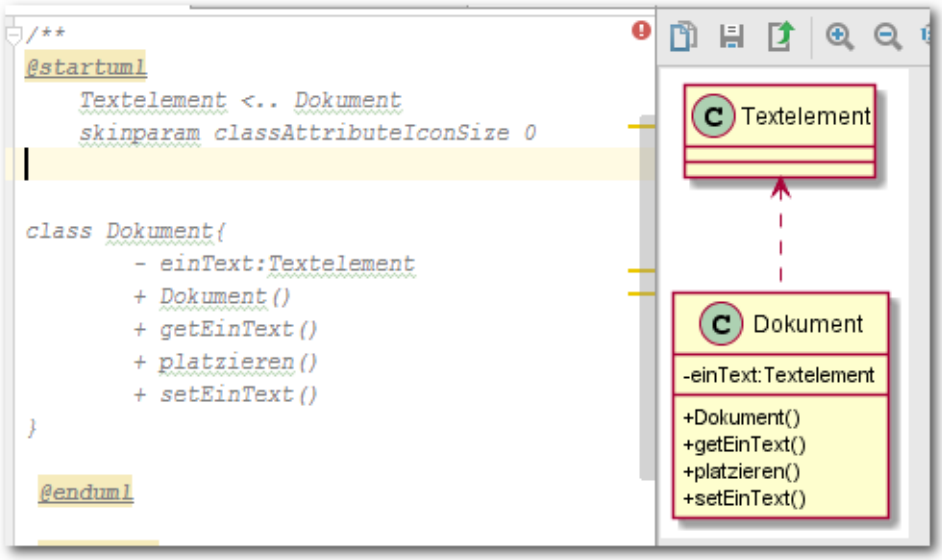
Zur Modellierung in UML benötigt man i.d.R. zusätzliche Erweiterungen für die Entwicklungsumgebung. Diese müssen installiert sein um die grafische Darstellung zu realisieren.

Arbeitsauftrag:

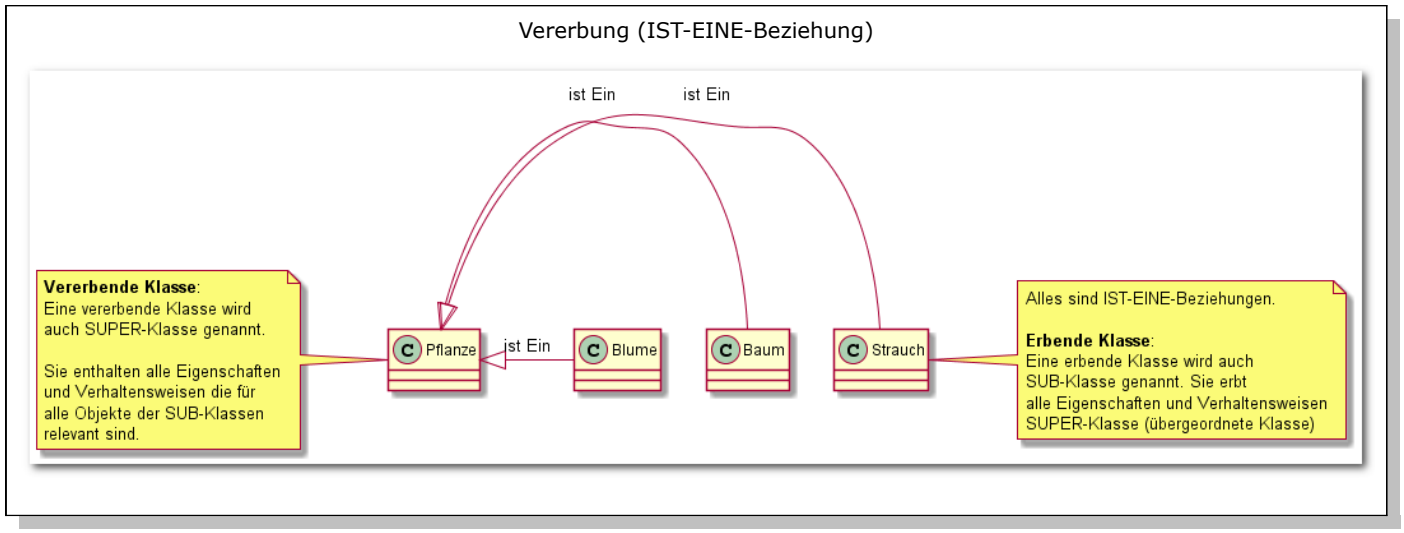
1. Installieren Sie ggf. die Erweiterungen.
2. Realisieren Sie die gezeigten Beispiele.
3. Dokumentieren Sie die Notation der Beziehungen und der Klasse selbst.



UML Notation Einführung → wichtige Verbinder (Pfeile) und Bedeutungen für Assoziationen



UML Notation Einführung → Klassen mit einseitig, gerichteter Assoziation

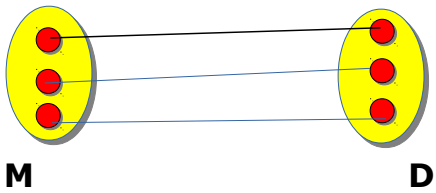


UML Notation Einführung → wichtige Verbinder (Pfeile) und Bedeutungen für erbende und vererbende Klassen

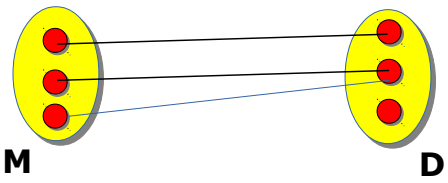
8 Assoziationen

Assoziationen

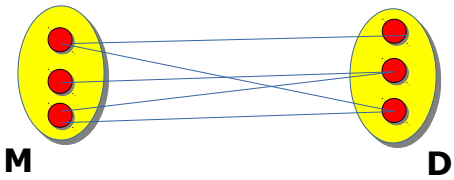
Thema:	Containerklassen Urquelle: BKW Handreichung OOP Übung: Dienstwagen - Assoziationen, Multiplizitäten, Containerklassen (ArrayList)



Multiplizität: 1:1



Multiplizität: 1:N



Multiplizität: N:M

Sachverhalt:

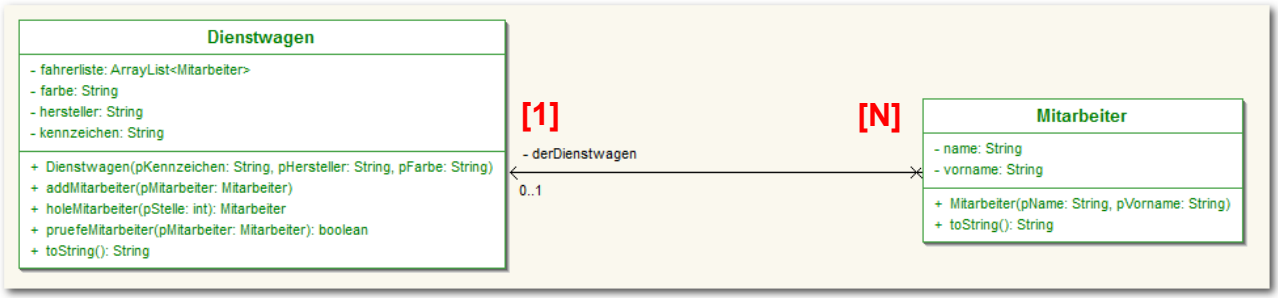
Die Firma Allgäu Cars GmbH stellt ihren Außendienstmitarbeitern Geschäftswagen zur Verfügung. Bisher konnte jeder Mitarbeiter immer nur ein und dasselbe Auto benutzen. Da sich in den Absatzgebieten Oberschwaben/Bodensee und Stuttgart die Nachfrage stark erhöht hat, werden für diese Absatzgebiete jeweils eine Mitarbeiterin eingestellt. Da immer nur ein Mitarbeiter/eine Mitarbeiterin in diesen beiden Absatzgebieten Außendienst macht und der/die andere dann im Innendienst arbeitet, teilen sich Mitarbeiter zum Teil einen Dienstwagen.

Arbeitsauftrag:

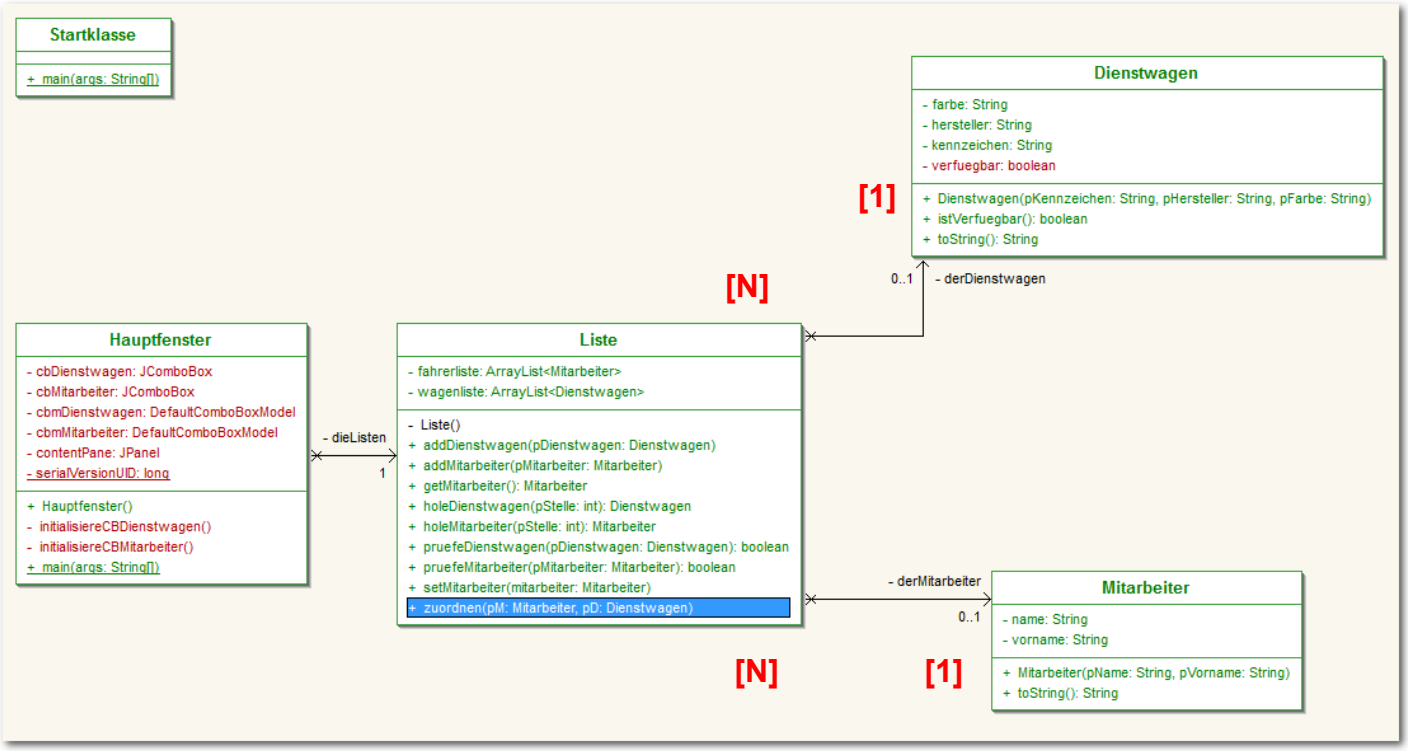
1. Vervollständigen Sie die angezeigten Fenn-Diagramme, um die Multiplizitäten darzustellen.
2. Welche der genannten Lösungen (UML-Klassendiagramme) entspricht dem geschilderten Sachverhalt?
3. Welche der gegebenen Startklassen (Unit-Tests) testet die → Systemarchitektur 2?
4. Implementieren Sie die Ereignissteuerung für die Schaltfläche → Zuordnen. Nutzen Sie das EVA-Prinzip.

Thema:	Assoziationen Urquelle: BKW Handreichung OOP Informationsblatt: Assoziationen, Multiplizitäten, Containerklassen (ArrayList)

Assoziationen und Multiplizizäten



Systemarchitektur 1: Multiplizität: 1:N



Systemarchitektur 2: Multiplizität N:M → 1:N und N:1

Thema:	Dienstwagenverwaltung Urquelle: BKW Handreichung OOP Informationsblatt: Startklasse (Unit-Tests 1)

Unit-Test 1: Startklasse

```
3  public class Startklasse {
4      public static void main(String[] args){
5          //Objekt der Klasse
6          Liste dieListen = new Liste();
7
8
9          //VERARBEITUNG
10         //Fahrer kann folgende Dienstwagen fahren
11
12         for(int i = 0; i < dieListen.getFahrerliste().size(); i++ ){
13             Mitarbeiter mMitarbeiter = dieListen.holeMitarbeiter(i);
14             System.out.println("-----------------------------------------------------------------");
15             System.out.println("Fahrer: "+mMitarbeiter.toString()+" fährt Dienstwagen:");
16             System.out.println("-----------------------------------------------------------------");
17             sprungmarke1:
18                 for(int j = 0; j < dieListen.getWagenliste().size();j++){
19                     Dienstwagen mDienstwagen = dieListen.holeDienstwagen(j);
20                     if(mDienstwagen.istVerfuegbar()){
21                         dieListen.zuordnen(mMitarbeiter, mDienstwagen);
22                         mDienstwagen.setVerfuegbar(false);
23
24                         //AUSGABE
25                         System.out.println(mDienstwagen.toString());
26                         System.out.println(mDienstwagen.toString()+" jetzt belegt!");
27                         break sprungmarke1;
28                     }else{
29
30                         System.out.println(mDienstwagen.toString()+ " ist immer noch belegt!");
31                         continue sprungmarke1;
32                     }
33                 }
34
35         }
36     }
37 }
```

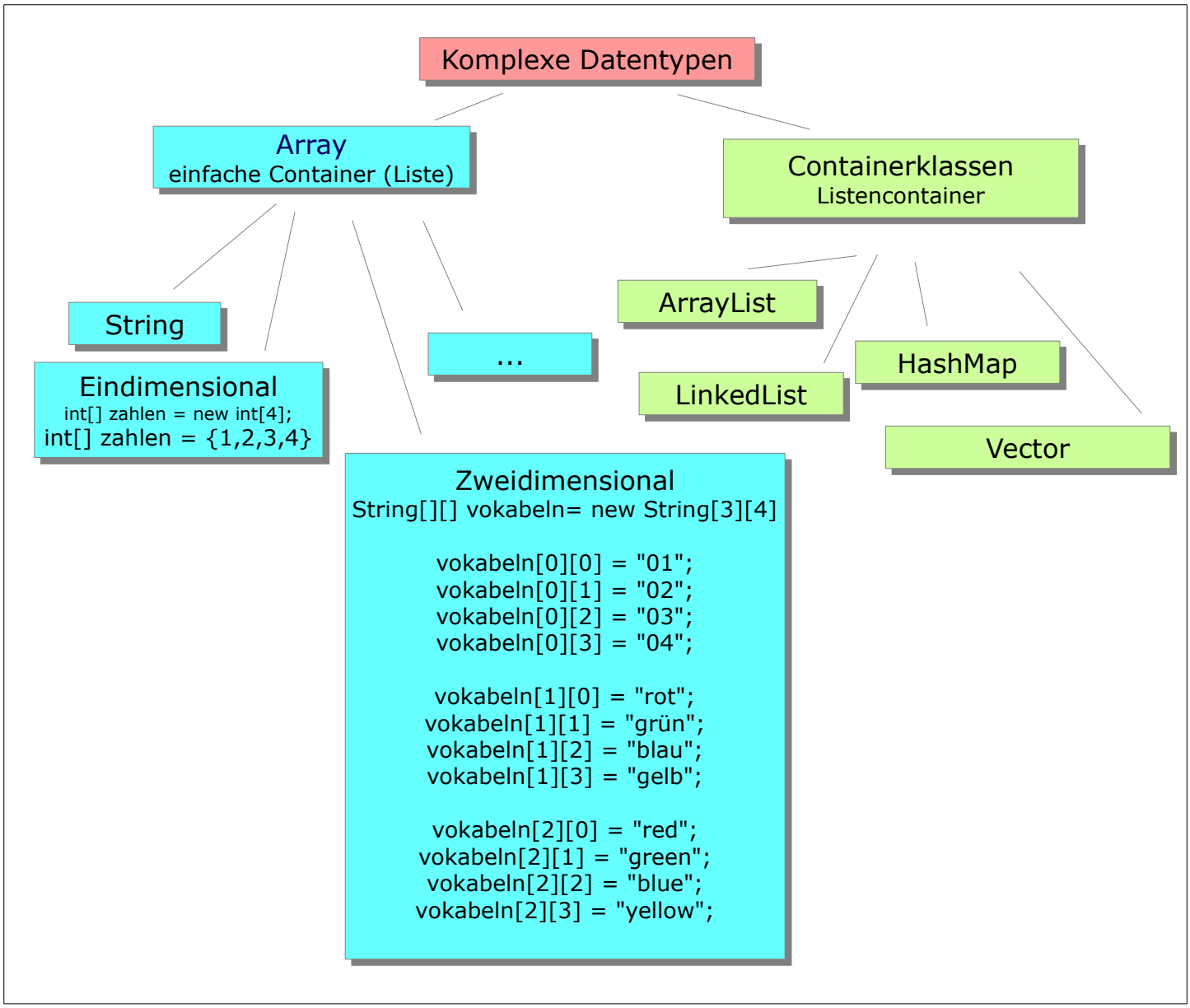
Thema:	Dienstwagenverwaltung Urquelle: BKW Handreichung OOP Informationsblatt: Startklasse (Unit-Tests 2)

Unit-Test 2: Startklasse

```
3  public class Startklasse {
4    public static void main(String[] args){
5
6      //EINGABE
7        //Mitarbeiter Objekte
8        Mitarbeiter mitarbeiter1 = new Mitarbeiter("Müller","Lisa");
9        Mitarbeiter mitarbeiter2 = new Mitarbeiter("Maier","Kurt");
10       Mitarbeiter mitarbeiter3 = new Mitarbeiter("Keller","Fritz");
11       Mitarbeiter mitarbeiter4 = new Mitarbeiter("Mauser","Karl");
12       Mitarbeiter mitarbeiter5 = new Mitarbeiter("Hauser","Fred");
13
14       //Dienstwagen Objekte
15       Dienstwagen dienstwagen1 = new Dienstwagen("RV-LM-2002","Audi","Blau");
16       Dienstwagen dienstwagen2 = new Dienstwagen("RV-KK-2121","Mercedes","Rot");
17       Dienstwagen dienstwagen3 = new Dienstwagen("RV-KF-0404","VW","Gelb");
18
19
20     //VERARBEITUNG
21       //Fahrer fährt mit Dienstwagen
22       mitarbeiter1.setDerDienstwagen(dienstwagen1);
23       mitarbeiter2.setDerDienstwagen(dienstwagen2);
24       mitarbeiter3.setDerDienstwagen(dienstwagen2);
25       mitarbeiter4.setDerDienstwagen(dienstwagen3);
26       mitarbeiter5.setDerDienstwagen(dienstwagen3);
27
28
29       //Fahrerliste des Dienstwagens
30       dienstwagen1.addMitarbeiter(mitarbeiter1);
31       dienstwagen2.addMitarbeiter(mitarbeiter2);
32       dienstwagen2.addMitarbeiter(mitarbeiter3);
33       dienstwagen3.addMitarbeiter(mitarbeiter4);
34       dienstwagen3.addMitarbeiter(mitarbeiter5);
35
```

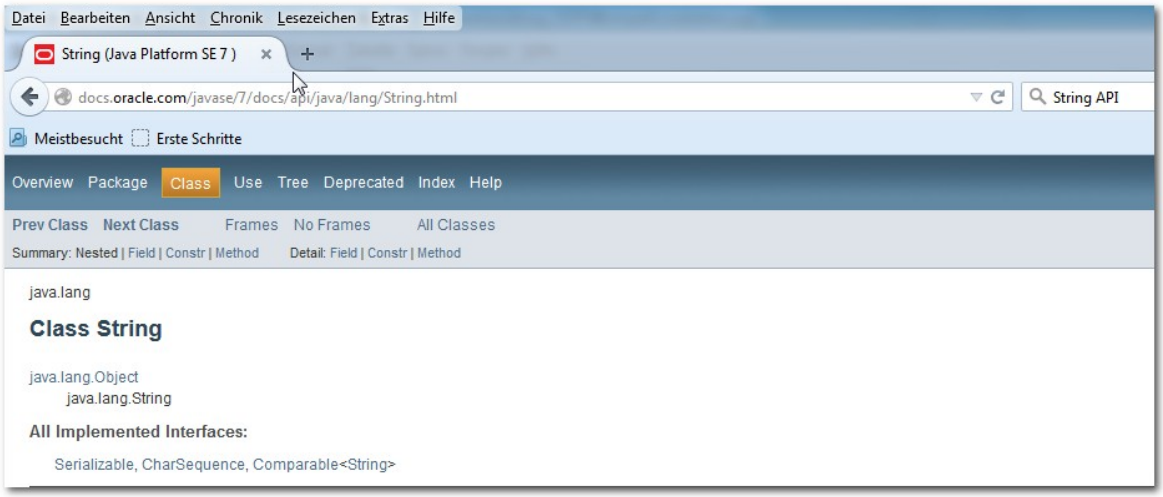
```
37     //AUSGABE
38
39     String mTab ="\t";
40
41     System.out.println("-----------------------------------------------------------------");
42     System.out.println("Mitarbeiter " + mTab + "fahrt mit " + mTab + "Dienstwagen" );
43     System.out.println("-----------------------------------------------------------------");
44     System.out.println(mitarbeiter1.toString());
45     System.out.println(mitarbeiter2.toString());
46     System.out.println(mitarbeiter3.toString());
47     System.out.println(mitarbeiter4.toString());
48     System.out.println(mitarbeiter5.toString());
49
50     System.out.println("-----------------------------------------------------------------");
51     System.out.println("Fahrer des Dienstwagen: "+dienstwagen2.getHersteller()+mTab
52                 + dienstwagen2.getKennzeichen() );
53     System.out.println("-----------------------------------------------------------------");
54     for(int i = 0; i < dienstwagen2.getFahrerliste().size(); i++ ){
55       Mitarbeiter mMitarbeiter = dienstwagen2.holeMitarbeiter(i);
56       System.out.println(mMitarbeiter.toString());
57     }
58
59   }
60 }
61
```

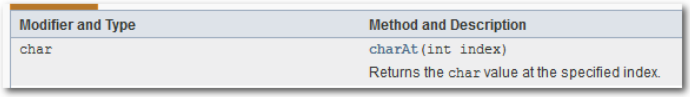
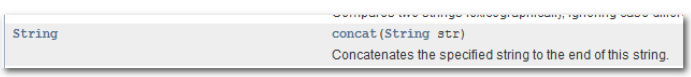
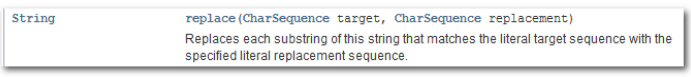
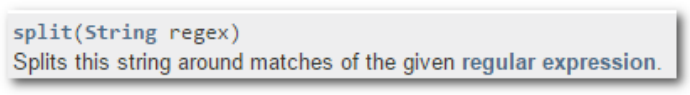
Thema:	Komplexe Datentypen Urquelle: Christine Janischek Informationsblatt: Überblick Komplexe Datentypen



Thema:	Applikation Programming Interface Urquelle: Christine Janischek Informationsblatt: Die Klasse String

Stringverarbeitung

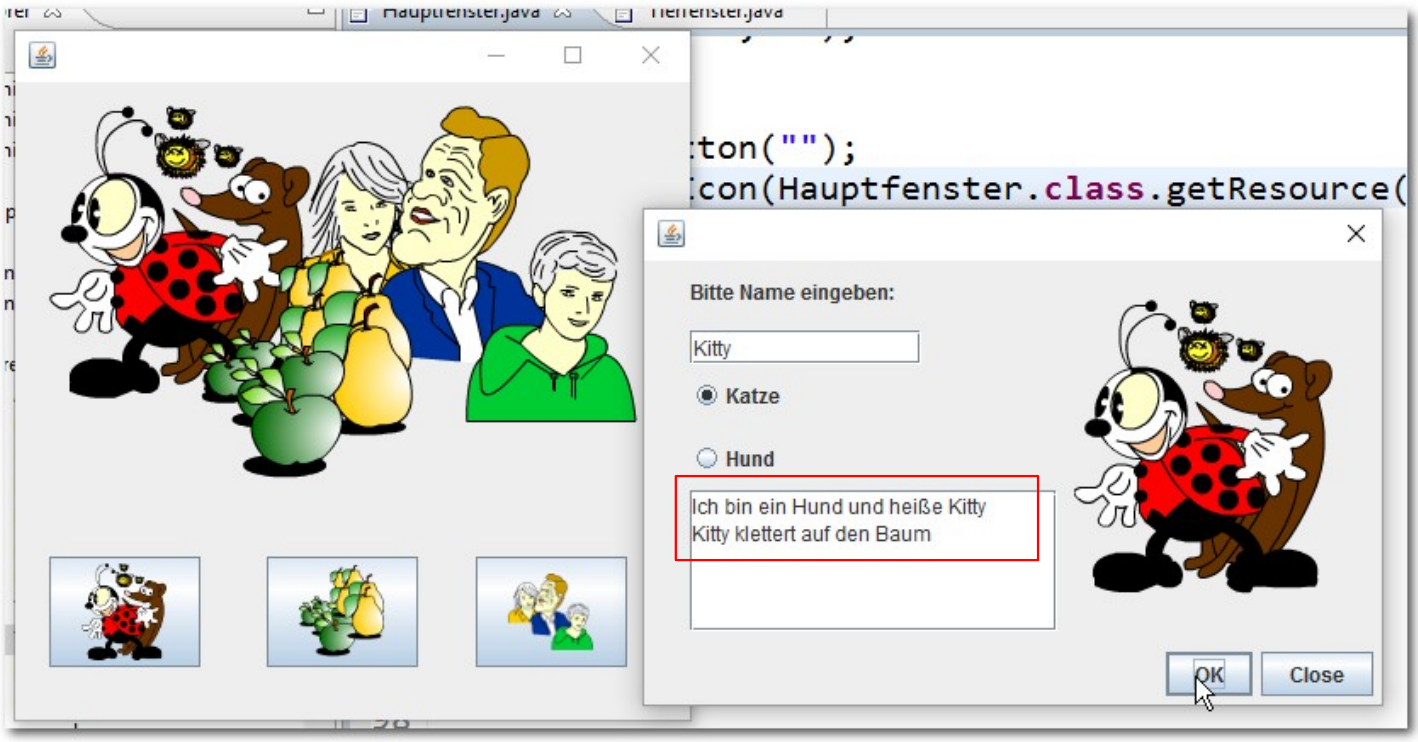


charAt(int i)	
concat(String str)	
length()	
replace(char oldChar, char newChar)	
split(String regex)	
...	...

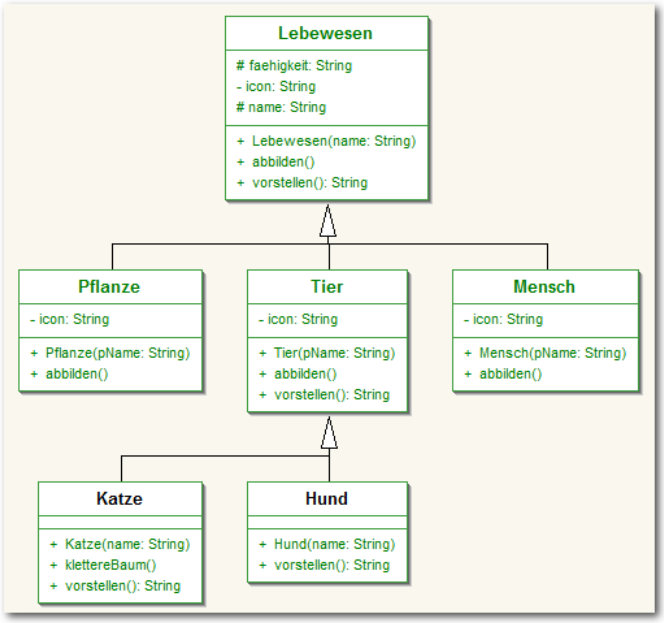
9 Vererbung

Vererbung

Thema:	Vererbungskonzept in objektorientierten Sprachen Urquelle: Christine Janischek Übung: Lebewesen



Hier hat sich ein Fehler eingeschlichen!



Thema:	Vererbungskonzept in objektorientierten Sprachen Urquelle: Christine Janischek Arbeitsblatt: Lebewesen

Arbeitsauftrag:

1. Setzen Sie die Vererbungsstruktur → Lebewesen in Quellcode um?
2. Implementieren Sie an geeigneter Stelle die Verhaltensweisen.
3. Testen Sie die Anwendung.
4. Dokumentieren Sie die Erkenntnisse zur Vererbung in Java.

Hilfestellung

SUPER-Klasse
Vererbende Klasse

Lebewesen, Tier

Lebewesen → Konstruktor:

```
public Lebewesen(String name) {
                this.name=name;
        }
```

SUB-Klasse
Erbende Klasse

Pflanze, Tier, Mensch

Tier → Klassendeklaration:

```
public class Subklasse extends Superklasse
```

Tier → Konstruktor:

```
public Tier(String pName) {
                super(pName);
        }
```

Polymorphie
Überschreiben von Methoden

Bietet die Möglichkeit im Rahmen der Spezialisierung von Verhaltensweisen (Methoden) die Implementierung anzupassen.

Lebewesen → vorstellen() : String

```
public String vorstellen() {
    return "Ich bin ein Lebewesen und heiße "+name+"\n\n"
            + name + " kann nichts besonderes!";
}
```

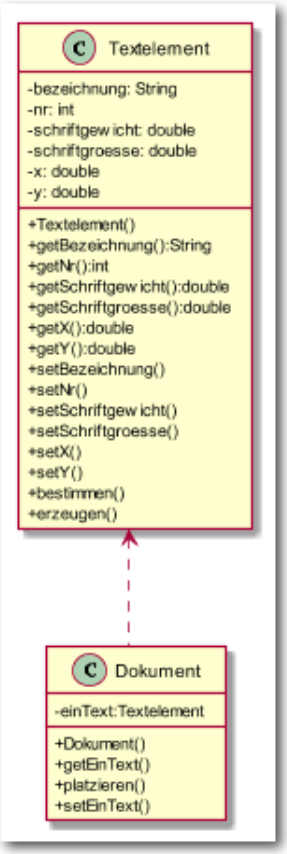
Tier → vorstellen() : String

```
public String vorstellen() {
    return "Ich bin ein Tier und heiße "+this.getName()+"\n\n"
            + this.getName() + " kann immer noch nichts besonderes!";
}
```

Abstrakte Klassen

Haben keinen Konstruktor. Es ist also nicht möglich ein Objekt dieser Klasse zu erzeugen. In UML werden die Klassennamen abstrakter Klassen *kursiv* geschrieben.

Thema:	Unified Modelling Language Urquelle: Christine Janischek Übung: Abstraktion und Vererbung (Textverarbeitungssoftware)



Situation

Ein Softwarehersteller möchte eine neue Testverarbeitungssoftware auf dem Markt etablieren. Die Software soll dem Nutzer die Möglichkeit bieten Textelemente zu erzeugen und anhand der Textelementkoordinaten x und y im Dokument zu platzieren. Jedes Textelement hat eine Nummer und eine Bezeichnung. Für alle Textelemente kann u.a. die Schriftgröße und das Schriftgewicht bestimmt werden. Der Nutzer kann nur spezielle Textelemente erzeugen! Im Speziellen kann der Nutzer dazu u.a. Verzeichniselemente und Standardtextelemente erzeugen. Speziell bei den Verzeichniselementen kann der Nutzer die Ebene selbst festlegen. Für ein Standardtextelement kann der Nutzer u.a. den Einzug vermindern oder erhöhen. Verzeichniselemente können in Inhaltsverzeichniselemente und Gliederungspunkte eingeteilt werden. Verzeichniselemente zeichnen sich dadurch aus, dass sie Bestandteil des Inhaltsverzeichnisses sind.

Arbeitsauftrag:

1. Leisten Sie im ersten Schritt die notwendige Textarbeit und ermitteln Sie dazu Substantive, Verben, Adjektive.
2. Vervollständigen Sie das angezeigte UML-Klassendiagramm mit Hilfe einer geeigneten Entwicklungsumgebung.
3. Klären Sie den den Begriff einer abstrakten Klasse mit Hilfe einer Internetrecherche und wenden Sie dieses Konzept auf das o.g. Beispiel an.
4. Dokumentieren Sie unbedingt die Realisierung der Vererbung im Quellcode der einzelnen Klassen.

Thema:	Unified Modelling Language Urquelle: Christine Janischek Übung: Spezialisierung und Generalisierung - Vererbung (Grafiksoftware)

Ein Softwarehersteller möchte eine ein neue Grafiksoftware auf dem Markt etablieren. Die Software soll dem Nutzer die Möglichkeit bieten Formen (shapes) zu erzeugen und auf einer Bühne zu platzieren. Jede Form hat eine Nummer und eine Bezeichnung. Im Speziellen kann der Nutzer u.a. Kreise und Vierecke erzeugen. Speziell bei den Kreisen kann der Nutzer den Radius für die Größe des Kreises selbst festlegen. Für ein Viereck kann der Nutzer u.a. die Länge und Breite für die Größe bestimmen. Vierecke können in Rechtecke und Quadrate eingeteilt werden. Ein Quadrat zeichnet sich dadurch aus, dass zur Bestimmung der Größe nur die Eingabe der Seitenlänge erforderlich ist.

Identifizieren Sie die *systemrelevanten* Subjekte, Verben und Adjektive. Erzeugen Sie das UML-Klassendiagramm für die Software (händisch - alle genannten Klassen, Attribute und Methoden). Klären Sie den Begriff der Spezialisierung und Generalisierung im Kontext der objektorientierten Programmierung.

Deklarieren Sie die Klasse Quadrat (Klasse, Konstruktor, Attribute,Methode(n)).

Thema:	Unified Modelling Language Urquelle: Christine Janischek Übung: Polymorphie und Vererbung (Allgäu-Car)

Die Firma Allgäu-Car GmbH vermietet folgende Fahrzeugarten. Andere Fahrzeugarten gibt es nicht.

Fahrzeugart	Motorräder	PKW	Nutzfahrzeuge
Attribute	mietdauer: int fixe_Mietkosten: double kilometerkosten: double gefahrene_kilometer: int mietkosten: double	mietdauer: int fixe_Mietkosten: double kilometerkosten: double gefahrene_kilometer: int mietkosten: double sitzplaetze:int	mietdauer: int fixe_Mietkosten: double kilometerkosten: double gefahrene_kilometer: int mietkosten: double sitzplaetze: int nutzlast: double
Methoden	• Die üblichen set- und get-Methoden • Methode mieten_pruefen(): Mindestmietdauer 3 Tage • Methode vorlaeufige_mietkosten_berechnen() Dabei gilt: mietkosten = fixe_Mietkosten * mietdauer + kilometer-kosten * gefahrene_kilometer • Methode hoehe_mietkosten_pruefen(). Dabei gilt, dass die Mindest-mietkosten 30,00 EUR betragen.		
Besonderheiten			Die Höhe der Mietkosten muss hier ebenfalls ge-prüft werden. Es gilt: Die Mindestmietkosten betragen für die Nutz-fahrzeuge 60,00 EUR

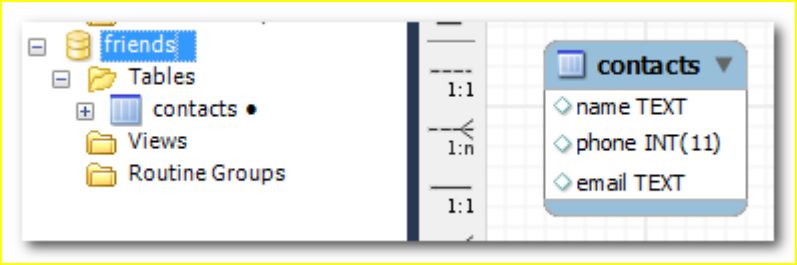
Arbeitsauftrag:

1. Leisten Sie im ersten Schritt die notwendige Textarbeit und ermitteln Sie dazu Substantive, Verben, Adjektive.
2. Vervollständigen Sie das angezeigte UML-Klassendiagramm mit Hilfe einer geeigneten Entwicklungsumgebung.
3. Wenden Sie das Konzept abstrakter Klassen an.
4. Klären Sie den den Begriff der Polymorphie mit Hilfe einer Internetrecherche und wenden Sie dieses Konzept auf das o.g. Beispiel an.
5. Dokumentieren Sie unbedingt die Realisierung der Vererbung im Quellcode der einzelnen Klassen.

10 Datenbankanbindung

Datenbankanbindung

Thema:	Datenbankanbildung in Java Urquelle: Christine Janischek Übung: Überblick Datenbankoperationen



Lokal oder Serverseitig?

CREATE DATABASE
Datenbank erzeugen
 	
PATH:
Umgebungs- und Systemvariable setzen
Systemsteuerung → System und Sicherheit → System → Erweiterte Systemeinstellungen
 	
C:\Programme\Android\sdk\tools
 	
C:\Programme\Android\sdk\tools\platform-tools
 	
Terminal öffnen
adb shell
 	
Verzeichnis wechseln
cd /data/data
ls
 	
cd com.example.chrissi.friends
ls
 	
Verzeichnis anlegen
mkdir databases
cd /databases
ls
 	
Datenbank erzeugen bzw. öffnen
sqlite3 friends.db
	MySQL:
	`CREATE DATABASE friends;`
	
	SQLite:
	`sqlite3 friends.db`

CREATE TABLE Datenbanktabellen erzeugen	MySQL: `CREATE TABLE friends.contacts (`

<table>
<tr>
<td>

```
//String zum erzeugen des SQL-Create-Table-Befehls
public static final String SQL_CREATE =
        "CREATE TABLE "
                + TABLE_CONTACT_LIST
                + "(" + COLUMN_ID
                + " INTEGER PRIMARY KEY AUTOINCREMENT, "
                + COLUMN_NAME + " TEXT NOT NULL, "
                + COLUMN_PHONE + " TEXT NOT NULL, "
                + COLUMN_EMAIL + " TEXT NOT NULL);";
```

</td>
<td>

```
        name TEXT,
        phone INTEGER,
        email TEXT);
```

SQLite:

```
CREATE TABLE contacts (
        name TEXT,
        phone INTEGER,
        email TEXT);
```

</td>
</tr>
<tr>
<td>SELECT
Daten auswählen</td>
<td>MySQL:

```
SELECT * FROM friends.contacts;
```

SQLite:

```
SELECT * FROM contacts;
```

</td>
</tr>
<tr>
<td>INSERT INTO
Daten einfügen</td>
<td>MySQL:

```
INSERT INTO friends.contacts
(name,phone,email) VALUES
('Chrissi',123789,'chrissi@mydomain.de'),
('Brian',1234,'brian@mydomain.de'),
('Roy',56789,'roy@mydomain.de');
```

SQLite:

```
INSERT INTO contacts (name,phone,email) VA-
LUES
('Chrissi',123789,'chrissi@mydomain.de');

INSERT INTO contacts VALUES
('Brian',1234,'brian@mydomain.de');

INSERT INTO contacts VALUES
('Roy',56789,'roy@mydomain.de');
```

</td>
</tr>
<tr>
<td>UPDATE
Daten ändern</td>
<td>MySQL:

```
UPDATE friends.contacts SET name='Christi-
ne' WHERE name='Chrissi';
```

SQLite:

```
UPDATE contacts SET name='Christine' WHERE
name='Chrissi';
```

</td>
</tr>
<tr>
<td>DELETE
Daten löschen</td>
<td>MySQL:

```
DELETE FROM friends.contacts WHERE
name='Roy';
```

SQLite:

```
DELETE FROM contacts WHERE name='Roy';
```

</td>
</tr>
</table>