Objektorientiertes Programmieren:

Background Fragen:

Was ist ein Programm:

Eine Sammlung von Befehlen und Daten, die es einem Computer erlauben Berechnungen oder Kontrollfunktionen auszuführen.

Was ist objektorientiertes Programmieren:

OOP ist programmieren mit Objekten statt mit prozeduralen Elementen.

Was ist ein Objekt in OOP:

Ein Objekt ist eine gekapselte Menge an Daten und Methoden auf diesen Daten, belegen Speicherplatz .

Warum objektorientiertes Programmieren:

Gründe für objektorientiertes Programmieren sind unter anderem, dass die Denkweise beim Programmieren natürlichen Abläufen ähnlicher ist, es die Wiederverwendbarkeit von Programmteilen erleichtert und in gewisser weise auch einfach „modern“ ist.

Konzepte des objektorientierten Programmieren:

Was ist eine Klasse:

Eine Klasse ist ein Prototyp, der die Variablen und die Methoden gemeinsam für alle Objekte einer Bestimmten Art definiert (Instanzveriablen, Instanzmethoden).

Object test = new Object ();
Constructor: Eine Methode zum Initialisieren neuer Objekte die von einer Klasse kreert wurden, hat den selben Namen wie die Klasse, die Initialwerte können dem Constructor als Arguemente übergeben werden.

Objekte, die dieselben Botschaften verstehen, dieselben Merkmale aufweisen und sich nur in der Ausprägung der Merkmale unterscheiden, bilden eine Klasse von Objekten. Eine Klasse legt in gewisser Analogie zum Datentyp einer Variable den Typ eines Objekts fest. Daher wird synonym zum Begriff Klasse auch der Begriff Objekttyp gebraucht.

Datenkapselung:

Schutz von Daten innerhalb eines Objektes gegen direkten Zugriff von außen. Entsprechende Methoden sollen bereit gestellt werden um Änderungen in einem kontrollierbaren Rahmen zu halten. (Information Hiding) Mit diesem Mittel können sowohl Datenschutzaspekte erfüllt werden als auch Fehlermöglichkeiten eingegrenzt werden indem Änderungen nur in einem erlaubten Rahmen durchgeführt werden dürfen.

Was ist eine Message:

Softwareobjekte interagieren und kommunizieren miteinander, indem sie einander Messages senden.

Object_1 ------ Message ------ Object_2

Subklassen vs. Superklassen:

Eine Subklasse ist eine Klasse, die eine andere Klasse erweitert.
Eine Superklasse ist eine Klasse, von der eine bestimmte Klasse abgeleitet wird (auch indirekt möglich)

Was ist Vererbung:

Ist ein Mechanismus zum Halten gemeinsamer Information, eine Klasse erbt den Zustand und das Verhalten seiner Superklasse, dabei können Methoden auch hinzugefügt und überschrieben werden.

Unter Vererbung versteht man die Möglichkeit, ein neues Objekt von einem vorhandenen Objekt abzuleiten, wobei das neue Objekt alle Merkmale und Fähigkeiten des alten besitzt. Dem neuen Objekt können dann weitere charakteristische Merkmale hinzugefügt werden. So könnte man auf der Basis des Objekts »Fahrzeug« das Objekt »Auto« oder »Lokomotive« ableiten. »Auto« erbt dann die Merkmale von »Fahrzeug«, so zum Beispiel »besitzt Räder« oder »kann Personen aufnehmen«. Ferner könnte »Auto« das Objekt »Fahrzeug« um einige Fähigkeiten erweitern, beispielsweise »anlassen« und »ausschalten«.

Was ist Polymorphismus:

Polymorphismus ist die Vielgestaltigkeit.
Die Polymorphie macht es möglich, dass verschiedene Unterklassen dieselbe Botschaft verstehen, obwohl die technische Umsetzung der Reaktion auf diese Botschaft völlig unterschiedlich sein kann. Auf die Botschaft »anfahren« können sowohl Objekte vom Typ »Auto« als auch vom Typ »Lokomotive« reagieren, und das Resultat ist bei beiden vergleichbar, nämlich, dass sie sich in Bewegung setzen. Allerdings sind die Handgriffe, die hierzu erforderlich sind, bei beiden Objekten sehr unterschiedlich.

Was ist ein Interface:

Ein Interface ist eine benannte Sammlung von Methodendefinitionen (ohne Implementierung)

Klassen können ein oder mehrere Interfaces implementieren worin auch ein wesentlicher Unterschied liegt. Der Interface Name selber kann auch als Typ verwendet werden (zB. für Klassen die dieses Interface implementieren).

Abstract Classes vs. Interface:

Eine Klasse kann Methoden implementieren, ist Teil einer Klassenhierachie und kann nur eine Klasse erweitern.

Ein Interfache kann keine Methoden implementieren sondern nur deklarieren, ist selber nicht Teil einer Klassenhierachie und kann eine eine beliebige Anzahl von Schnittstellen erweitern.

Typen und Subtypen:

Was ist ein Typ:

Die Definition eines Typs hängt stark von der entsprechenden Sichtweise ab.
Aus Sicht des Systemprogrammierers: Filter für Rohdaten (Bits)
Aus Sicht des Implementierenden: Speicher Abbildung für Werte
Aus Sicht der Typenkontrolle: Kompatibilität von Operand und Operator
Objektorientierte Sichtweise: Verhaltensspezifikation

Typen in Java:

Primitive Typen: byte, short, int, float, double, char, boolean
Referenz: Arrays, Klassen und Schnittstellen

Subklasse vs. Subtyp:

Jede Klasse wird in Java auch als Typ angesehen. Nicht alle Subklassen können auch als Subtypen betrachtet werden (warum genau ? – irgendwas mit dem Verhalten ...)!

Ersetzbarkeit:

Jede Instanz eines Subtypes kann verwendet werden, wo auch immer eine Instanz des Supertyps erwartet wird. Dies ist besonders in Hinblick auf Wiederverwendbarkeit wichtig. Als Beispiel ist ein Polygon eine Superklasse von Dreieck und Quadrat, trotzdem bleiben die beiden aber ihrerseits Polygone.

Design Pattern:

Ein verallgemeinerter Lösungsansatz für immer wiederkehrende Probleme. Diese Ansätze können dann im gegebenen Kontext umgesetzt und verwendet werden.

Singleton-Pattern:

Das Singleton-Pattern wird im Kontext des Kreieren von Instanzen verwendet um sicherzustellen dass von einer Klasse nur ein einzige Instanz existiert. Bekannte Anwendungen sind Window-Manger und Spooler etc.

Das Pattern verwendet einen Privaten Constructor, um zu verhindern dass die Klasse mehr als einmal instanziert wird. Die Speicherung der Referenz auf die einzige Instanz dieser Klasse übernimmt eine Klassenvariable. Das Kreieren erfolgt innerhalb einer statischen Methode in der auch berücksichtigt wird ob bereits eine Instanz zuvor schon erzeugt wurde.

Observer Pattern:

Das Observer-Pattern wird im Kontext des Zustandswechsels oder Ereignisses – das für andere Objekte relevant sind – verwendet. Datenänderung an einer Stelle soll bekannt gemacht werden.

Ein Publisher registriert dynamisch einen oder mehrere Subscriber die sich für ein Ereignis interessieren und benachrichtigt sie im Falle einer Änderung.

Wird für Synchronisationszwecke verwendet.

Testen von objekt. Programmen:

Worum geht es beim Testen:

Beim Testen geht es darum Systeme bzw. Teilsysteme unter spezifizierten Bedingungen laufen zu lassen und die Ergebnisse zu Verfolgen (Aufzuzeichnen) um sie mit den erwarteten Werten vergleichen zu können.

Wozu testen:

Man Testet um Fehler zu finden, erwünschtes Verhalten verifizieren zu können und um die Qualität einstufen zu können.

Zutaten für das Testen:

Testfälle Menge an Test-Eingaben und Anfangsbedingungen samt zu erwartender Resultate
Testkriterien spezifizierte Kriterien die erfüllt werden müssen und zu bestehen sind
Testdokumentation Spezifikation des Testfalles und dem Verlauf als Testbericht
Tester Personen die diese Arbeit ausführen, Fehleingaben machen etc.

White-Box-Testen:

Testen einer Software unter Rücksichtnahme der inneren Struktur des Programms. Die Testdaten werden in Hinsicht der Wirkungsweise ausgewählt. Ein explizites wissen über den Source-Code ist Voraussetzung. Eingabewerte werden mit Ausgabewerten verifiziert um eine korrekte Wirkungsweise zu ermitteln.

Black-Box-Testen:

Bei diesem Testverfahren wird nicht auf die inneren Codeteile eingegangen sondern das Software Fragment, Funktion etc als Black-Box betrachtet. Es sind nur die Eingangsdaten und die Ausgangsdaten bekannt aufgrund dessen man feststellen kann ob die Spezifikation erfüllt wird. Dabei wird aber nicht auf die innere Struktur eingegangen.

Arten des Testen:

- Testen einer Einheit
- Integrationstesten
- System / Akzeptanztesten
- Leistungstesten
- Stresstesten
- Sicherheitstesten
- Regressionstesten

Testen einer Einheit:

Testen einer Einheit (Modul, Komponente) im Stand-Alone Betrieb erfordert eine Testumgebung wobei sowohl Black-Box als auch White-Box Testen möglich ist.

System- und Akzeptanztesten:

Testen eines vollständig integrierten Systems.
Evaluierung der Übereinstimmung mit spezifizierten Anforderungen, bzw. Erfüllung der festgelegten Abnahmebedingungen. Black-Box Testen

Was ist besonderes an OOP:

Testen einer Klasse als Einheit.
Systemtesten basierend auf auf Use-Cases / Szenarien.
Modellbasierendes Testen basierend auf UML
Spezialthemen über (Vererbung, Datenkapselung, Polymorphismus)

Datentypen:

- Short
- Int
- Long
- Float
- Double
- Boolean
- Char

Methoden:

Instanzmethoden:
- Haben Zugriff auf alle Variablen und Methoden der Klasse
- Gebunden an die Existenz eines Objekts der Klasse

Klassenmethoden:
- Schlüsselwort **`static`**
- Sind nicht an die Existenz eines Objekts gebunden
- Aufruf: **`Klassenname.Klassenmethode`**

Klasse:

Class EmptyClass {

}

Erzeugen on Objekten / Instanzen einer Klasse mittels new() Operator.

Variablen:

Instanzvariable ist an ein Objekt gebunden, sie werden automatisch initialisiert.

Klassenvariable
- Existieren genau einmal für die gesamte Klasse
- Definition mit dem Schlüsselwort static
- Sie können von außen (bei erlaubten Zugriff) angesprochen werden Klassenname.Klassenvariable
- Klassenvariable sind nicht an die Existenz eines Objects gebunden, ihre Lebensdauer höngt vom Ladezustand der Klasse ab.

Konstruktoren:

```
Public ExampleClass () {       // Leerer Konstruktor
}
```

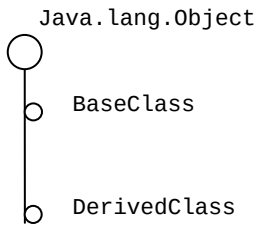
- Existiert kein Konstruktor, so wird ein default Konstruktor aufgerufen
- Ein Konstruktor kann einen anderen Konstruktor aufrufen
- Es darf auch Methoden geben die gleich lauten wie der Konstruktor, aber diese müssen einen Rückgabetyp haben

Vererben:

Schlüsselwort: **extends**

```
Class DerivedClass extends BaseClass {

}
```



Typenverträglichkeit:

Jedes Subklassenobjekt hat alle Eigenschaften der Basisklasse und ist daher ein spezielles Basisklassenobjekt. Darum ist es auch vom Typ der Basisklasse:

Typenverträglichkeitsregeln:

Eine Subklasse kann ohne weiteres über eine Basisklassenreferenz gespeichert werden
Ein Superklassenobjekt kann nach einem expliziten Typecast über eine Subklassenreferenz gespeichert werden. Typen die nicht in einer Vererbungslinie liegen können nie in Beziehung miteinander stehen.

Abstrakte Methoden:

- Enthalten n ur die Deklarationen des Methodenrumpfes, aber keine Implementierung
- Abstrakte Methoden können nicht aufgerufen werden
- Konkretisierung durch Ableiten der Klasse

```
public abstract void foo ();
```

Abstrakte Methoden:

Eine Klasse die mindestens eine abstrakte Methode besitzt ist selbst abstrakt. Schlüsselwort: **abstract**

- Können nicht instanziert werden.
- Eine Klasse kann auch **abstract** definiert werden, wenn sie keine abstrakten Methoden besitzt
- Eine konkrete Methode kann in der nächsten Vererbungsstufe wieder abstrakt definiert werden.

Interfaces:

Jede Klasse darf beliebig viele Interfaces implementieren. Ein Interface darf nur abstrakte Methoden und Konstanten enthalten.

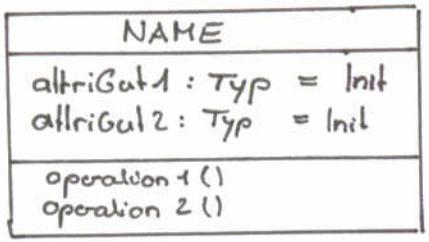
```
Interface I {
  // Interfacekörper
}
```

```
class A implements I {
  // Klassenkörper
}
```

Alle Methoden eines Interface sind standardmäßig ***abstract*** und ***public***. Nur wenn alle Methoden des Innterface konkretisiert sind => nicht abstrakt

UML

Klasse:



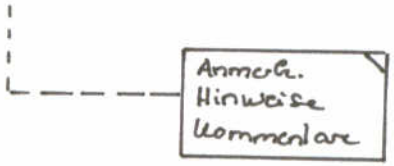
Bei abstrakten Klassen steht unter dem Namen noch „{abstract}"

Attribut:

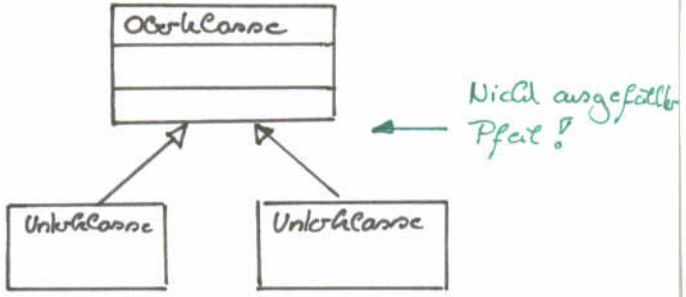
attribut : Klasse = Init-Wert (Merkmal) {Zusicherung}

Merkmale : Zum Beschreiben von Besonderheiten
Zusicherung: Einschränkung des Wertebereichs

Notiz



Vererbung



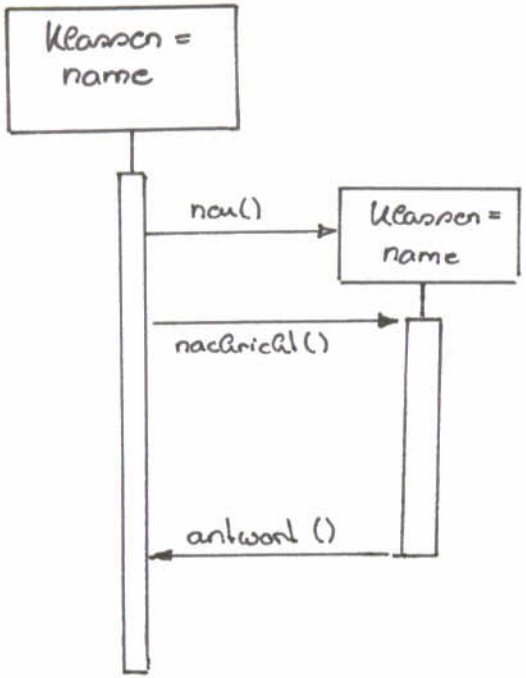
Assoziation

Notation durch eine Linie zwischen den beteiligten Klassen. An den Enden der Verbindungslinie kann die Multiplizität der Beziehung angegeben werden.

Multiplizität

- Angabe des Min. und des Max getrennt durch zwei Punkte
- Mit einem * wird der jeder Beschränkung = viele
- Mit einem Komma können mehrere Möglichkeiten aufgezählt werden.

Sequenzdiagramm



Zeitlinie verläuft senkrecht von oben nach unten, die Objekte werden durch senkrechte Lebenslinien beschrieben, die gesendeten Nachrichten wagrecht entsprechend ihres Auftretens eingetragen.

Einlesen von System-IN

```
private String readLine() {

     BufferedReader keyb = new BufferedReader(new
                    InputStreamReader(System.in));

     String name;

     try {
          name = keyb.readLine();
          return name;
     } catch (Exception e) {
          System.out.println("FEHLER beim einlesen");
          return null;
     }
}
```

Durchsuchen eines Vector

```
Vector liste = null;
Element temp = null;

liste = new Vector();

Iterator listenIterator = liste.iterator();

while ( listenIterator.hasNext() ) {
     temp = (Element) listeniterator.next();
}
```

String

```
STRING_1.CompareTo ( STRING_2 )

Liefert 0 wenn die beiden Strings gleich sind
Größer 0 wenn 1 größer als 2
Kleiner 0 wenn 1 kleiner als 2

CompareToIgnoreCase( STRING )

ValueOf ( int, float, long, double )

ToUpperCase()

ToLowerCase
```

Integer parseInt(STRING)