**JAVA**

1. **Javac, java**

„Javac“ kompiliert Javadateien nach Bytecode; „java“ interpretiert Bytecode

1. **Data types**

Primitive Datentypen

Primitive Datentypen besitzen weder Eigenschaften (Variablen) noch Fähigkeiten (Methoden) besitzen



Nicht primitive Datentypen  
sind z.B. Felder, Klassen  
🡪 ermöglichen es, neue, komplexere Typen zu definieren Variable eines Referenztyps, enthält Verweis auf ein Objekt

🡪 Während primitive Typen lediglich deklariert werden, reicht dies bei Referenztypen nicht aus. Sie müssen mit Hilfe **des****[new](https://dbs.cs.uni-duesseldorf.de/lehre/docs/java/javabuch/html/index_n.html" \l "ixb100083)-Operators oder** - im Falle von Arrays und Strings - durch Zuweisung von Literalen zusätzlich noch explizit erzeugt werden

1. **Wrapper Classes**

Eine Wrapper-Klasse kapselt die jeweilige primitive Variable in einem sehr einfachen Objekt. Dabei stellt die Wrapper-Klasse einige Methoden für den Zugriff auf die primitive Variable und nützliche Funktionen zur Verfügung

* **Übergabe des Werts als primitiver Datentyp** – Dem Konstruktor wird der primitive Wert übergeben. Der Konstruktor erzeugt dann ein Objekt des korrespondierenden Datentyps.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* **Übergabe des Werts als *String*** – Dem Konstruktor wird der Wert als *String* übergeben. Anschließend wird der übergebene *String* konvertiert und ein Objekt des korrespondierenden Datentyps erzeugt.



* **Nutzen der *valueOf*-Fabrikmethode** – Mit Hilfe der *valueOf*-Fabrikmethode wird ein Objekt des korrespondierenden Datentyps erzeugt. Die *valueOf*-Fabrikmethode ist so implementiert, dass zu gleichen primitiven Werten identische Instanzen ausgegeben werden. Somit werden überflüssige Objekte vermieden



1. **String (Zeichenkette)**

String ist eine Klasse, deshalb wird String auch im Gegensatz zu int oder double großgeschrieben.

* ***s.length()*** ergibt die Anzahl der Zeichen von s.
* ***s.charAt(0****)* ergibt das erste Zeichen von String s. Position 0 entspricht dem ersten Zeichen.
* ***s.charAt(1)*** würde das zweite Zeichen ergeben usw.
* ***s.charAt(s.length()-1****)* ergibt das letzte Zeichen von s.
* ***System.out.println(s.equals("Hallo****"))* gibt ein true aus, wenn s dem String „Hallo“ entspricht, ansonsten wird false ausgegeben. Auf diese Weise lassen sich 2 String Variablen oder Eigenschaften miteinander vergleichen

**Stringbuffer**

Ein *StingBuffer*-Objekt kann durch vier überladene Konstruktoren erzeugt werden

* StringBuffer()
* StringBuffer(CharacterSequenz cs)
* StringBuffer(int capacity)
* StringBuffer(String s)

stellt aber selbst kein String-Objekt dar, sondern muss - wenn gewünscht - durch die Methode *toString()* in einen solchen gewandelt werden.  
Die wesentlichen Routinen zur Manipulation eines *StringBuffer*-Objektes sind die vielfach überladenen Methoden insert() und append(), mit deren Hilfe jeder beliebige Wert entweder an einem als Parameter übergebenen Index eingefügt oder ans Ende angehängt werden kann

**Ein Bild, das Text enthält.

Automatisch generierte BeschreibungEin Bild, das Text enthält.

Automatisch generierte Beschreibung**

**Stringbuilder**

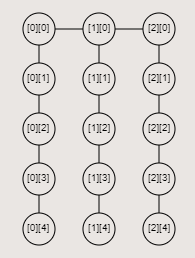
Das gleiche nur mit -builder

1. **Array**

Arrays sind Datentypen, die zur Speicherung mehrerer Werte eines einzigen Typs dienen. Ein Array ist in Java selbst ein Objekt und wird mit new erzeugt.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung oder 

Mehrdimensionales Array



1. **Operators**

Arithmetische Operatoren

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

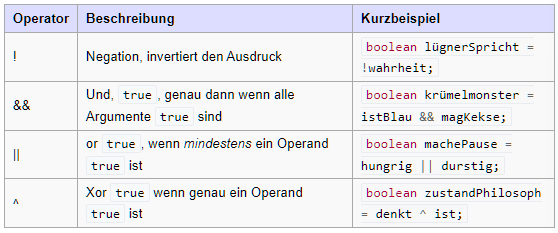
Automatisch generierte Beschreibung

Vergleichsoperatoren

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

Boolische Operatoren



Zuweisungsoperatoren

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

1. **Nutzung statischer Methoden**

Methoden

Ein Bild, das Text, Screenshot, Bildschirm enthält.

Automatisch generierte Beschreibung

„STATIC“

„Static“ zeigt an, das etwas direkt zu einer Klasse gehört: Ein statisches Feld gehört zu einer Klasse; eine statische Methode ebenso.

Wenn zum Beispiel das Feld count in der Klasse Counter statisch ist, dann kannst du die Variable mit dem folgenden Ausdruck referenzieren: Counter.count.

„VOID“

Keine Daten werden übergeben oder der Typ der Daten ist nicht angegeben. Syntaktisch wird void wie ein Datentyp behandelt, aber es kann nur an bestimmten Stellen stehen.

1. **Schleifen und Verzweigungen**

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

1. **Methoden aufrufen und Rekursion**

Methoden werden auf einem Objekt mit dem Punktoperator aufgerufen: obj.meineMethode();Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

**Reursion**

Unter Rekursion versteht man in der Programmierung ein Verfahren, bei dem sich eine Methode selbst aufruft, sodass, ähnlich einer Endlosschleife, ein potenziell unendlicher Programmablauf entsteht.

Z.B.:

Ein Bild, das Text enthält.

Automatisch generierte BeschreibungEin Bild, das Text enthält.

Automatisch generierte Beschreibung

1. **Parameterübergabe als Wertkopie oder Objektreferenz**

Wertkopie

**Ein Bild, das Text enthält.

Automatisch generierte Beschreibung**

Objektreferenz

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

1. **Overloading von Methoden**

Wenn wir eine Methode überladen, bedeutet das, dass wir mindestens zwei Methoden vom gleichen Namen innerhalb einer Klasse haben.

Somit verbleibt uns zur Unterscheidung lediglich die Parameterliste und der Rückgabewert. Zumeist wird die Anzahl der übergebbaren Parameter erweitert, um mehr Optionen zu erlauben, oder verringert, um Standardwerte zuzuweisen. Wenn die Anzahl der Parameter gleichbleibt, ermöglicht man durch Überladung die Nutzung einer Funktion mit anderen Datentypen

**Und um es kurz zu machen! Eine Unterscheidung über den Rückgabewert funktioniert nicht.**

1. **OOP**

Objekt: Ein Software Objekt ist ein Bündel aus Attributen und darauf bezogenen Methoden.

Klasse: Eine Klasse ist ein Bauplan, welcher die Attribute und Methoden definiert, die alle Objekte einer bestimmten Art besitzen

(Schlüsslewort class;

ihr Bezeichner wird mit großem Anfangsbuchstaben geschrieben und folgt dem sog. CamelCase-Aufbau;

der Klassenquelltext (Variablen und Methoden) ist in einem Block eingeschlossen, der durch geschweifte Klammern eingegrenzt wird)

Abstrakte Klasse: Abstrakte Klassen sind Klassen, die als Oberklassen dienen und selbst nicht zur Bildung von Objekten herangezogen werden können. (Schlüsselwort abstract)

Interface (Schnittstellen): Interfaces ermöglichen eine spezielle Form der Mehrfachvererbung, da es in Java nicht erlaubt ist, eine Klasse von mehreren Superklassen abzuleiten.

(Schlüsselwort: interface

Anfangsbuchstabe ist groß;

Ist immer abstract ohne, dass es angegeben werden muss)

Vererbung: Schlüsselwort „extends“ (ist eine Art von); private Eigenschaften werden nicht vererbt;

Dynamische Polymorphie: Bei der dynamischen Polymorphie werden in mehreren Klassen einer Vererbungslinie gleiche, d.h. mit identischen Signaturen versehene Methoden unterschiedlich implementiert, sodass zur Laufzeit entschieden werden muss, welche der Methoden ausgeführt wird.  **Mit der Annotation @Override kennzeichnen**

Sichtbarkeiten:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Zugriffsmethoden:

Getter: (get) + name des entsprechenden Feldes zB.: getWeight() (gibt den wert des Felds weight des Objekts zurück

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Setter: (set) + name

Kapselung: Integer i1 = new Integer(2); //Boxing

Int test = i1.intValue(); //Unboxing

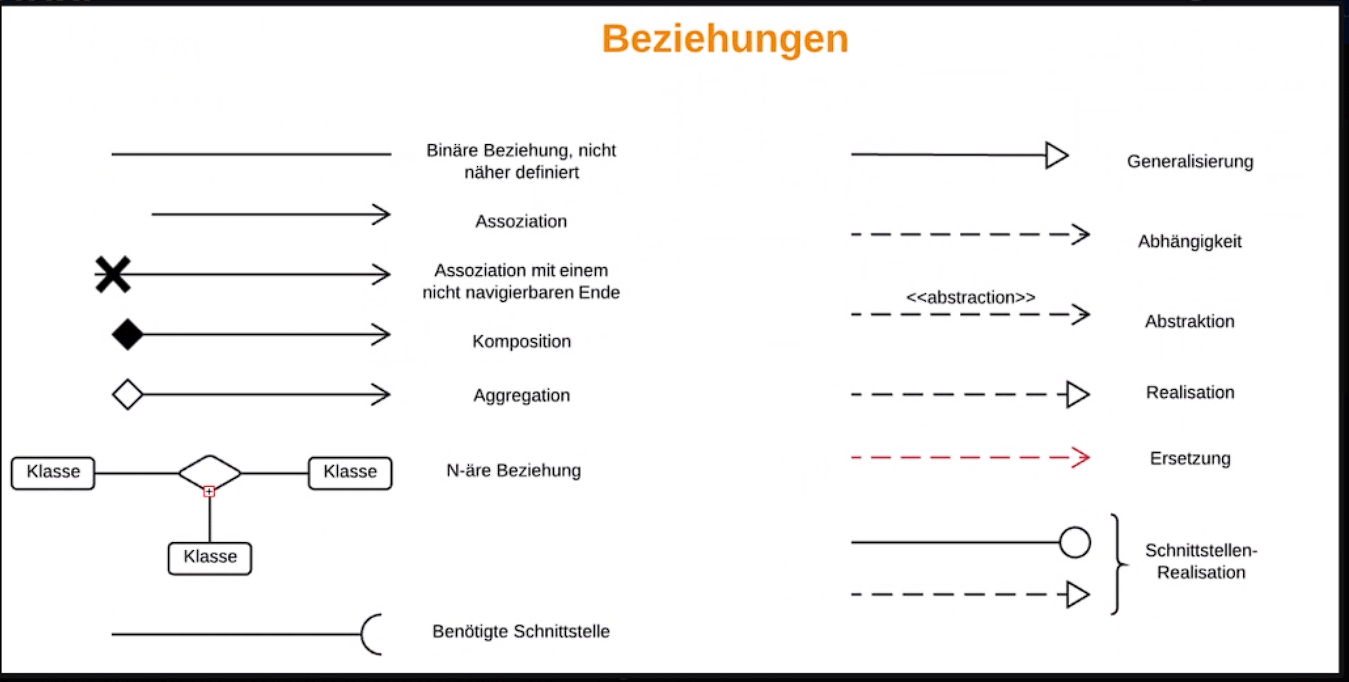
Integer i2 = 3;//Autoboxing

Int i3 = i2;

UML-Diagramm: public +, private -, protected #, package ~, kursiv geschrieben = abstrakt, Attribut unterstrichen == Klassenattribut; Methode unterstrichen = Klassenmethode; Pfeile „extends“ kommt an Pfeilanfang; Großschreibung bedeutet

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



Generalisierung = erben

Exception handling:

* Try-catch: (InputMismatchException)Ein Bild, das Text enthält.

  Automatisch generierte Beschreibung

+ finally (wird immer ausgeführt)

* Throw: eigene Ausnahmen werden behandelt

Nützliche Links:

[Objektorientiertes Programmieren in Java (kompf.de)](https://www.kompf.de/java/tutor.html#objekte)

[Javabeginners - Abstrakte Klassen](https://javabeginners.de/Klassen_und_Interfaces/Abstrakte_Klassen.php)