Java Spickzettel von Panjutorials.de

Hallo Welt

```
public class HalloWelt{
	public static void main(String[] args){
		// Gibt "Hallo Welt" auf die Konsole aus
		System.out.print("Hallo Welt");
	}

}
```

Eine eigenständige Applikation in Java:

```
class <klassenName>
{
	public static void main(String args[]){
	unsere Statements;
	_____;
	_____;
	}
}
```

Primitive Variablen in Java

Folgend sind die primitiven Variablen in Java. String, arrays und Objekte sind nicht dabei, da es sich dabei nicht um primitive Variablen handelt.

Primitive Variablen in Java

Typ	Bits	Digits	Niedrigster Wert	Höchster Wert
Boolean	1	1	false	true
char	16	4:5	‚\u0000‘ [0] erreichbar via Character. MIN_VALUE	‚\uffff‘ [216-1] erreichbar via Character. MAX_VALUE
byte	8	2:3	-128 [-27] erreichbar via Byte. MIN_VALUE	+127 [27-1] erreichbar via Byte. MAX_VALUE
short	16	4:5	-32,768 [-215] erreichbar via Short. MIN_VALUE	+32,767 [215-1] erreichbar via Short. MAX_VALUE
int	32	9:10	-2,147,483,648 [-231] erreichbar via Integer.MIN_VALUE also -2 gig, etwa -2 milliarden	+2,147,483,647 [231-1] erreichbar via Integer. MAX_VALUE. Auch 2 gig, etwa 2 milliarden
long	64	1:19	-9,223,372,036,854,775,807[-263] erreichbar via Long. MIN_VALUE etwa -9×1018	9,223,372,036,854,775,808[+263-1] erreichbar via Long. MAX_VALUE etwa 9×1018
float	32	7	±1.4012984643248170 7e-45 erreichbar via Float. MIN_VALUE	±3.40282346638528860e+38 erreichbar via Float. MAX_VALUE etwa ±2127 hat 7 signifikante Genauigkeitsstellen Ein Float kann auch Integer repräsentieren: für folgenden Bereich -224 to +224.
double	64	16	±4.94065645841246544e-324 erreichbar via Double. MIN_VALUE	±1.79769313486231570e+308 erreichbar via Double. MAX_VALUE etwa ±21023 hat 15 signifikante Genauigkeitsstellen, etwa 16 mit 15.95 signifikante Ziffern. Ein Double kann auch Integer repräsentieren: für folgenden Bereich-253 to +253.

Variablen

```
Anlegen/Deklarieren von Variablen:
<datentyp> <variablen name>;
Beispiel: int zahl;
```

Kommentare

```
Um eine einzelne Zeile zu kommentieren: //
Um einen Block von Code zu kommentieren: /* ---------- */)
Kommentierte Inhalte werden nicht vom Programm ausgeführt.
```

Kontrollstrukturen

If / Else If / Else

```
if ( a > b ){
      out.println( a );
}
else if (a < b){
      out.println( b );
}
else{
      out.println( "beide gleich groß" );
}
```

If / Else um Code auszuführen wenn der Wert Zwischen 2 Werten ist

```
if ( niedrig <= x && x <= hoch )
{
      out.println( "dazwischen" );
}
else
{
      out.println( "außerhalb" );
}
```

Switch Case:

```
switch ( n )
{
case 1: out.println( "eins" );
      break;
case 2: out.println( "zwei" );
      break;
default: out.println( "für alle nicht behandelten Fälle");
```

Schleifen

For Schleife

```
// F O R wird verwendet wenn man vorher weiß wie oft man
// den code wiederholt durchführen möchte
for ( int i=0; i<n; i++ ){
      out.println( i );
}
```

For Schleife für's durcharbeiten eines Strings

```
// F O R um durch einen String durchzuarbeiten
// Dadurch müssen wir die Länge des Strings nicht neu berechnen
// Dadurch müssen wir n nicht außerhalb anlegen.
// Beachte, man muss ein Komma vor n verwenden, kein Semikolon.
for ( int i=0,n=s.length(); i<n; i++ ){
      out.println( s.charAt( i ) );
}
```

Rückwärtszähler mit For

```
for ( int i=n-1; i>=0; i-- ){
      out.println( i );
}
```

Auslesen eines Arrays mit For (funktioniert auch mit ArrayList und Collections)

```
String[] inhalt = new String[ 10 ];
for ( String s : inhalt )  // Beachte, wir verwenden hier String nicht int
{
      out.println( s );
      s = etwasAnderes; // Achtung, dies hat keinen Einfluss auf den Inhalt // des Array
}
```

Iterator:

```
// beachte das Semikolon nach hasNext()
for ( Iterator iter = list.iterator(); iter.hasNext(); ){
      String key = (String)iter.next();
      out.println( key );
}
```

Foreach Schleife

```
void meineMethode( String... tiere )
{
      for ( String tier : tiere )
      {
            out.println( tier );
      }
}
...
// aufruf der meineMethode Methode
meineMethode( "Hund", "Katze", "Maus" );
meineMethode( "Hund" );
meineMethode( );

String[] einStringArray = getValues();
meineMethode( einStringArray );
```

While Schleife

```
// W H I L E (Schleife die die jedoch auch 0 mal ausgeführt werden kann)
// Wenn man nicht weiß, wie häufig die Schleife durchlaufen werden soll
while ( mehrDaten() )
{
     leseDies();
}
```

Do While Schleife:

```
// D O / W H I L E (wird mindestens einmal durchgeführt)
// Wenn man nicht weiß, wie häufig die Schleife durchlaufen werden soll, sie jedoch
// mindestens einmal durchlaufen werden soll
do{
      leseDies();

      if ( erledigt() ){
      break;
      }

      if ( lasseDieseAus() ){
            continue;
      }

      führeDiesAus();

}
while ( mehrDaten() );
```

Try, Catch & Throw

Wird verwendet um Fehler abzufangen.

```
public class Test extends StandardTest
{
      public static void main (String [] args)
      {
            try
            {
                  unsichereMethode();
            }
            catch ( StrangeException e ) // in JDK 1.7+ can man mehrere Ausnahmen gleichzeitig abfangen z.B.(IOException|
SQLException ex)
            {
                  out.println( "Fehler: " + e.getMessage() );
            }
      } // beende main

       void unsichereMethode() throws unsereException
      {
            if ( unerwartet() ) throw new unsereException ( "verdammt, etwas ist schiefgegangen" );
      } // beende unsichereMethode
} // beende die Klasse Test
```

Die standardmäßig eingebauten Ausnahmen/Exceptions:

1	**ArithmeticException** Arithmetischer Fehler, z.B. teilen durch 0.
2	**ArrayIndexOutOfBoundsException** Array index ist außerhalb der Reichweite des Array.
3	**ArrayStoreException** Zuweisung eines nicht kompatiblen Typs zu einem Array.

4	**ClassCastException** Invalider Ausdruck.
5	**IllegalArgumentException** Verbotenes Argument beim Aufruf einer Methode verwendet.
6	**IllegalMonitorStateException** Verbotene Monitoroperation, wie z.B. das warten auf einem offenen Thread.
7	**IllegalStateException** Umgebung oder Programm sind in einem inkorrekten Zustand.
8	**IllegalThreadStateException** Abgefragte Operation ist nicht kompatibel mit dem aktuellen Threadzustand.
9	**IndexOutOfBoundsException** Der Index ist außerhalb der Reichweite.
10	**NegativeArraySizeException** Array wurde mit negative Größe angelegt.
11	**NullPointerException** Verbotene Verwendung der Null Referenz.
12	**NumberFormatException** Verbotene Umwandlung eines Strings in ein numerisches Format.
13	**SecurityException** Versuch die Sicherheit zu verletzen.
14	**StringIndexOutOfBounds** Versuch einen String außerhalb seiner Reichweite zu indizieren.
15	**UnsupportedOperationException** Eine nicht unterstützte Operation ist aufgetreten.

Arrays

```
// Deklarieren und Initialisieren eines Arrays
String[] namen = {"Denis", "Michi", "Frank"};
// Zuweisen eines Wertes zum Array
namen[3] = "Claudia";
// deklarieren eines Arrays mit vorgegebener Größe
Int[10] werte = {0,1,2,3,4,5,6,7,8,9}

// Zweidimensionaler Array
double [][] = {
    {99.3, 23.3},
    {15.3, 125.3},
};
```

Zugriffsmodifikatoren

```
private
Wenn eine Variable oder Methode private deklariert ist,
dann ist sie nur in der Klasse in der sie erstellt wurde sichtbar,
also nicht von anderen Klassen.

public
Wenn eine Variable oder Methode public deklariert ist,
dann ist sie von jeder Klasse aus sichtbar und bearbeitbar,
die eine Referenz(ein Objekt) zu der Klasse der Variable/Methode besitzt.

default
Wenn eine Variable oder Methode default deklariert ist,
also keinen Zugriffsmodifikator hat,
dann wird sie vom Paket in dem sie befindlich ist
gesehen und kann via Referenz auf das Objekt der Klasse verändert/gesehen werden.

protected
Wenn eine Variable oder Methode protected deklariert ist,
dann kann nur vom gleichen Paket,
oder von Klassen die von der Klasse der Variable/Methode erben, a
us darauf zugegriffen werden.
```

Klassen und Objekte

```
//Anlegen einer Klasse

public class Internet {

	private String ipAdresse;
	private double geschwindigkeit;

	// Anlegen des Konstruktors
	public Internet(ipAdresse, geschwindigkeit){

	}

	// Methode der Klasse Internet
	public void verbinden(){
		System.out.println("Willkommen im World Wide Web! Deine Ip ist " +
		ipAdresse + " und deine Geschwindigkeit ist " + geschwindigkeit);
	}

}


//Anlegen von Objekten in der Main Klasse

public class Laptop {

	public static void main(String[] args) {
		Internet internetObjekt = new Internet("123.35.325.63", 67000000);
		internetObjekt.verbinden();
	}

}
```