Java Schnelleinstieg

Programmstart

Jedes Java-Programm benötigt einen Einstiegspunkt wie folgend:

```
class Main
{
  public static void main(String[] args)
  {

  }
}
```

Hier kann auch ein anderer Name benutzt werden

Hier ist der Einstiegspunkt des Programms.
Auf folgenden Folien wird die Struktur drumherum weggelassen.

Hello World

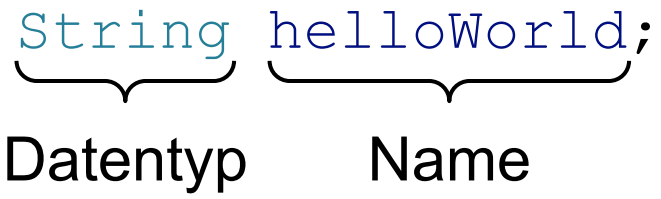
```
class Main
{
  public static void main(String[] args)
  {
    System.out.println("Hello World");
  }
}
```

Der Auszugebende Text

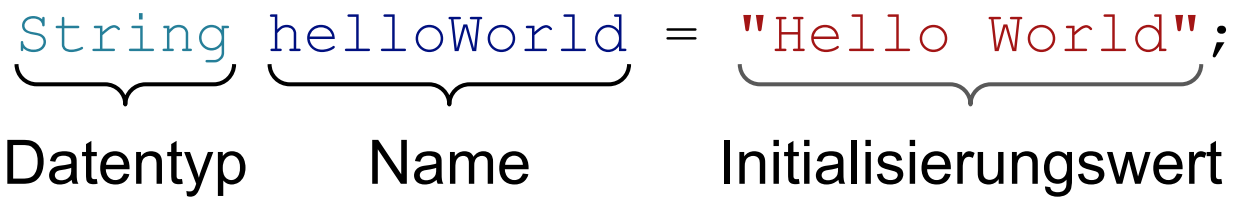
Damit werden Ausgaben auf der Konsole getätigt

Variablendeklaration

Bevor eine Variable benutzt werden kann muss sie zunächst deklariert werden.



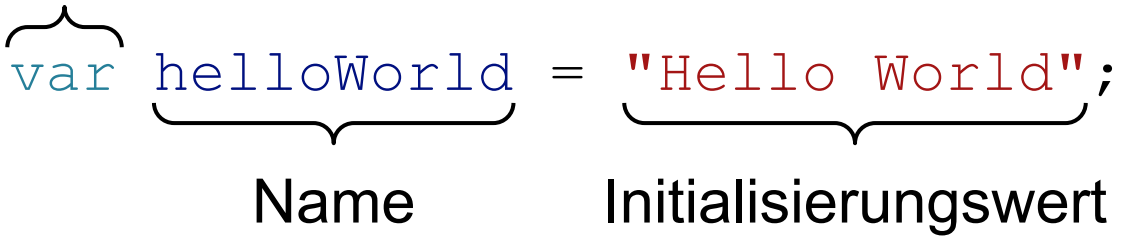
Dabei wird auch meistens direkt ein Wert zugewiesen:



Variablendeklaration

Bei direkter Initialisierung kann der Datentyp auch implizit abgeleitet werden:

String wird automatisch als Datentyp Erkannt und muss nicht angegeben werden



Primitive Datentypen

Datentyp	**Größe**	**Wertebereich**
boolean	1 bit	true oder false
byte	1 byte	-128 bis 127
short	2 bytes	-32.768 bis 32.767
int	4 bytes	-2^{31} bis $2^{31} - 1$
long	8 byte	-2^{63} bis $2^{63} - 1$
float	4 byte	$3.4*10^{-38}$ bis $3.4*10^{+38}$
double	8 byte	~~$1.8*10^{308}$ bis $4.9*10^{-324}$~~ groß!
char	2 byte	0 bis 65.535

Objekt Datentypen: String

```
String name = "Hello World";
```

Überprüfung der Gleichheit zweier Strings	boolean	`name.equals("Fischer")`
Abruf des Zeichens an Position 0	char	`name.charAt(0)`
Länge des Strings	int	`name.length()`

Verwendung von Variablen

Explizite Typdeklaration

```
String helloWorld = "Hello World";
System.out.println(helloWorld);
```

Die Variable kann als ersatz für den Wert eingesetzt werden

Implizit abgeleitete Typdeklaration

```
var helloWorld = "Hello World";
System.out.println(helloWorld);
```

```
var helloWorld = 42;
System.out.println(helloWorld);
```

Initialisierung versch. Variablentypen

```
boolean b1 = true;
byte b2 = 127;
short s1 = 32767;
int i = 42;
long l = 92233720368547758O7L;
float f = 3.14f;
double d = 3.14;
char c = 'A';
String s2 = "Hello World";
int[] a1 = {1, 2, 3};
int[] a2 = new int[3];
```

Nachträgliche Initialisierung

Falls eine Variable bei der Deklaration nicht direkt initialisiert wird muss sie durch Zuweisung initialisiert werden bevor sie benutzt werden kann

```
int x;
System.out.println(x);
x = 42;
```

The local variable x may not have been initialized

```
int x;
x = 42;
System.out.println(x);
```

Scope

Wenn eine Variable innerhalb eines Code-Blocks deklariert wird kann nur innerhalb dieses Blocks darauf zugegriffen wird:

```
{
  int x = 10;
  System.out.println(x);
}
System.out.println(x);
```

x cannot be resolved to a variable

Scope

Wenn eine Variable schon deklariert wurde, auch wenn es in einem umgebenden Code-Block ist, darf keine Variable mit demselben Namen nochmal deklariert werden.

```
int x = 42;
{
  int x = 10;
}
```

Duplicate local variable x

Ein- und Ausgabe von Strings

Deklaration und Initialisierung eines Scanners

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter your name: ");
String name = sc.nextLine();
System.out.println("Hello " + name);
```

Liest eine Zeile von Zeichen in Variable “name” ein

Der String "Hello " wird mit dem inhalt von “name” kombiniert

If-Abfragen

```
int x = 42;
```

Bedingung welche zu einem boolean Wert ausgewertet wird

```
if (x == 42)
{
  System.out.println("x ist 42");
}
else
{
  System.out.println("x ist nicht 42");
}
```

Code-Block falls die Bedingung zutrifft

Code-Block falls die Bedingung nicht zutrifft

If-Abfragen

```java
int x = 24;
if (x == 42)
{
  System.out.println("x ist 42");
}
else if (x == 24)
{
  System.out.println("x ist 24");
}
else
{
  System.out.println("x ist nicht 42 oder 24");
}
```

Code-Block falls die erste Bedingung nicht zutrifft, aber die zweite

Code-Block falls keine der Bedingungen zutrifft

If-Abfragen

```
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
if (n % 3 == 0)
{
  System.out.println("Fizz");
}
else if (n % 5 == 0)
{
  System.out.println("Buzz");
}
else if (n % 3 == 0 && n % 5 == 0)
{
  System.out.println("FizzBuzz");
}
else
{
  System.out.println(n);
}
```

Nachträgliche Initialisierung

```
int x;
int y = 42;
if (y < 24)
{
  x = 5;
}
System.out.println(x);
```

The local variable x may not have been initialized

Variable “x” muss in allen Code Pfaden initialisiert werden damit die Variable gelesen werden kann

```
int x;
int y = 42;
if (y < 24)
{
  x = 5;
}
else
{
  x = 24;
}
System.out.println(x);
```

Schleifen

```
int i = 1;
```

Solange die Bedingung zutrifft wird ein weiterer durchlauf vorgenommen

```
while (i <= 10)
{
  System.out.println(i);
  i++;
}
```

Code-Block der wiederholt ausgeführt wird

Schleifen

Für die while-Schleife muss eine Zählvariable außerhalb des Blocks definiert werden.

```
int i = 1;
while (i <= 10)
{
  System.out.println(i);
  i++;
}
```

Bei der do-while Schleife wird die bedingung beim ersten durchlauf nicht überprüft, somit wird die Schleife auf jeden fall einmal ausgeführt.

```
int i = 1;
do
{
  System.out.println(i);
  i++;
} while (i <= 10);
```

Schleifen

Es ist außerdem möglich, eine Zählvariable, die Schleifenbedingung und einen Ausdruck im Kopf der for-Schleife zu definieren und sie wie eine while-Schleife zu verwenden.

```
for (int i = 1; i <= 10; i++)
{
  System.out.println(i);
}
```

Mit der for-each-Schleife lässt sich durch die Elemente eines Array iterieren.

```
int[] array = {
  1, 2, 3, 4, 5,
  6, 7, 8, 9, 10
};
for (int element : array)
{
  System.out.println(element);
}
```

Vielen Dank