



# Java Einführung Objekt-Datentypen und Strings

Kapitel 8 und 9

# Inhalt

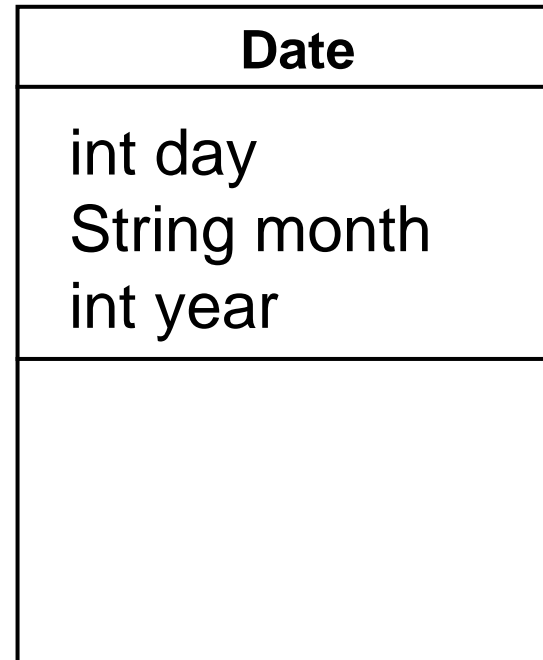
- Was sind Objekt-Datentypen
  - Besonderheiten bei Zuweisung und Vergleich
- Zeichenketten (Strings)
  - Zeichencodes
  - Char, Char-Arrays und Strings
  - String-Operationen
  - String-Konversion

# Objekt-Datentyp

- Jede Klasse in Java stellt einen sogenannten Objektdaten-Typ (Benutzerdefinierter Datentyp) dar.
- Die Instanzen verhalten sich meist wie primitive Datentypen, es gibt jedoch **wichtige Unterschiede!**

# Bsp: Date

```
class Date {  
    int day;  
    String month;  
    int year;  
}
```



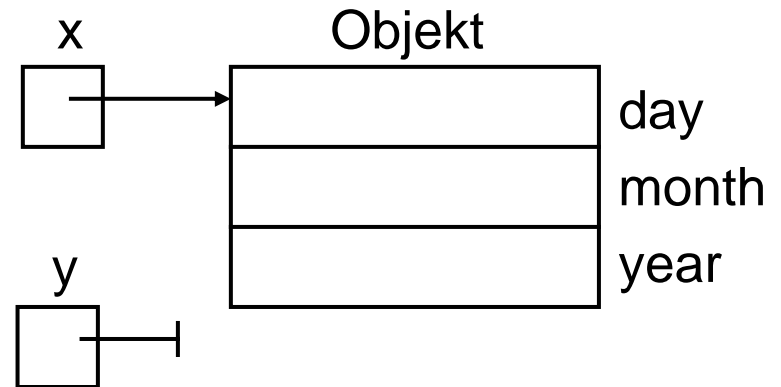
Date kann nach der Deklaration wie ein Datentyp verwendet werden.

# Erzeugung und Freigabe

- Deklaration und Erzeugung von Instanzen

```
Date x, y;
```

```
x = new Date();
```



- Freigabe von Objekten

**Garbage Collection:** Sobald kein Zeiger mehr auf das Objekt zeigt (Reference Count)

```
x = null; // x zeigt jetzt auf nichts!
```

# Punkt-Operator

- Verwendung der Klassenvariablen (Punkt-Operator)

```
Date x = new Date();
```

```
x.day = 13;
```

```
x.month = "November";
```

```
x.year = 2001;
```



Achtung: `month` ist vom Typ `String` und daher eigentlich ein `Array`.

# Zuweisung

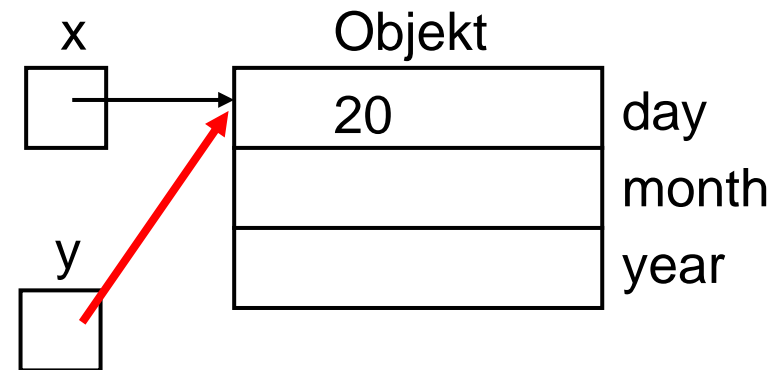
- Zuweisung von Objektnamen

```
Date x, y;
```

```
x = new Date();
```

```
y = x;
```

```
y.day = 20;
```



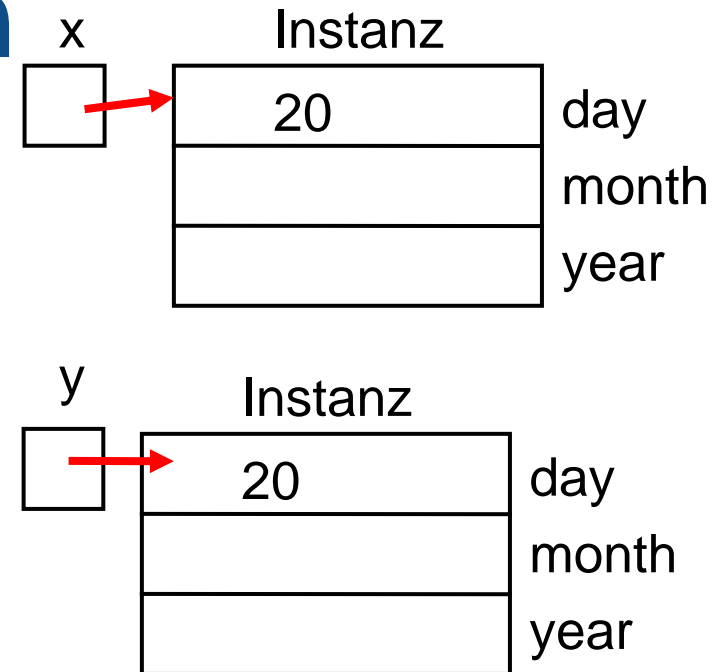
- Es wird nur die Referenz kopiert, nicht der Wert!
- Zuweisung ist nur bei gleichem Datentyp (Klassennamen) erlaubt!

# Vergleich

```
Date x, y;  
x = new Date();  
y = new Date();  
y.day = 20; x.day = 20;
```

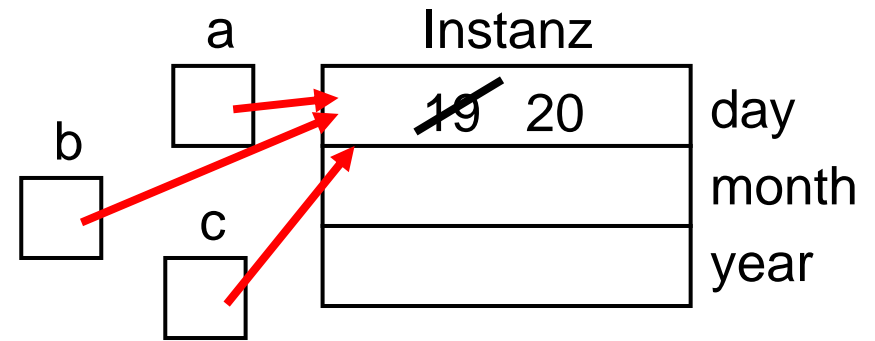
```
if ( x==y ) {  
    System.out.println("x ist gleich y");  
}
```

- Vergleich ergibt immer falsch, weil die Zeiger verglichen werden!



# Übergabe als Parameter

```
Date addOneDay (Date b) {  
    b.day++;  
    return b;  
}
```



...

```
Date a = new Date(); a.day=19;
```

```
Date c = addOneDay(a); // a hat jetzt auch 20!
```

# Zeichenketten in Java (Strings)

- Zeichenketten bestehen aus einzelnen Zeichen.
- Diese Zeichen sind vom primitiven Datentyp `char`
- Zeichenketten können als Array von `char` erzeugt werden, oder
- **als eigener Objekt-Datentyp `String` (ist in den Bibliotheken von Java vorhanden unter `java.lang.String`)**

# Zeichenkonstanten und -codes

'a' , 'A' , '?' ...

Zeichen werden in Java codiert als:

- **ASCII** (American Standard Code of Information Interchange): 128 Zeichen, benutzt 7 Bit, keine Umlaute...  
Darstellung: Zahl (Hex oder Dezimal)
- **Unicode**: benutzt 2 Byte, 65536 Zeichen, ersten 128 sind identisch mit ASCII (siehe: [www.unicode.org](http://www.unicode.org) )  
Darstellung: `\unnnn` ... `nnnn` ist der Code in Hex

# Zeichenvariable (Primitiver Datentyp)

- Deklaration

```
char ch;
```

- Zuweisung

```
ch = 'a';
```

- Kompatibilität mit `int`

```
char ch2 = 'x';
```

```
int i = ch2; // ok, Code in int-Variable
```

```
ch2 = (char) (i+1);
```

# Standardfunktionen für Zeichen

- Auswahl von Java-Bibliothek Standardfunktionen:

```
if (Character.isLetter(ch)) ... // Unicode?
```

```
if (Character.isDigit(ch)) ... // Digit?
```

```
ch=Character.toUpperCase(ch); // Groß
```

```
ch=Character.toLowerCase(ch); // Klein
```

siehe: `java.lang.Character`

# char-Arrays

- Eine Kette von Zeichen. Deklaration:

```
char[] s1 = new char[20];
```

```
char[] s2 = { 'a', 'b', 'c' };
```

- Zugriff auf einzelne Zeichen:

```
s2[1] = 'x'; // ersetzt 'b' durch 'x'
```

# Strings - Zeichenketten (Objekt-Datentyp)

- Char-Arrays sind relativ umständlich zu verwenden.
- Da Zeichenketten häufig vorkommen, stellt Java einen eigenen Bibliothekstyp mit dem Namen `String` (in `java.lang`) zur Verfügung.

# String-Konstanten

- Zeichenfolge zwischen doppelten Hochkommas

```
"Hello, I am a String!"
```

```
"Alice\t2000\nBob\t1500"
```

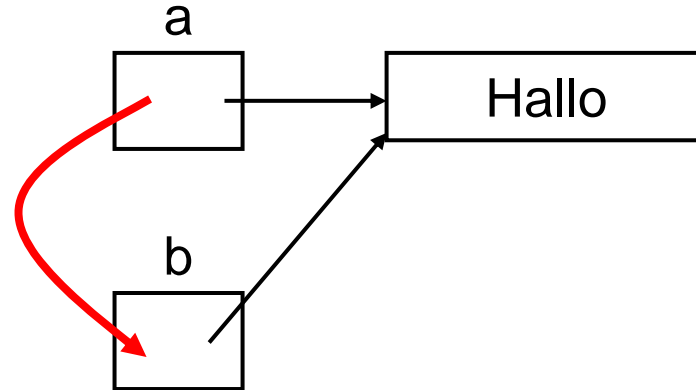
```
"Das \"-Zeichen "
```

```
"Griechische Symbol (pi): \u03c0"
```

- **Achtung:** Unterschied zwischen 'x' und "x"

# Datentyp String

- Deklaration  
`String a, b;`
- Zuweisung  
`a = "Hallo";`  
`b = a;`



- **Achtung:** Strings verhalten sich wie Arrays. Der Name ist nur ein Zeiger auf das "String-Objekt"

# Verkettung von Strings

- Verkettung

```
String a = "Hallo";
```

```
a = a + " Welt!"; // einfache Verkettung
```

- Andere Datentypen (z.Bsp: int) werden zu String konvertiert

```
int i = 12;
```

```
String b = "Es ist " + i;
```

```
System.out.println("i hat den Wert " + i);
```

# Vergleich von Strings

```
String s = new String("Hallo");
```

```
if (s=="Hallo") { ...
```

```
    // Fehler: Referenzen werden verglichen!
```

```
if (s.equals("Hello")) { ...
```

```
    /* die richtige Funktion (siehe:  
    java.lang.String) */
```

# Stringoperationen

- die Java-Bibliothek definiert eine Reihe nützlicher Funktionen für Strings (siehe: `java.lang.String`)

```
String s = "a long string"; String s2;  
int i; char ch;
```

```
i = s.length();           // bei Arrays kein ()!
```

```
ch = s.charAt(3);
```

```
i = s.indexOf("ng");
```

```
s2 = s.substring(2, 6)
```

```
if (s.startsWith("abc")) { ... }
```

# java.lang.StringBuffer

- Strings haben nach der Erzeugung eine fixe Länge. Zum zeichenweisen Aufbau eines Strings ist dies ungeeignet.

```
int i; // x kann sein char,int,float...
StringBuffer b = new StringBuffer("Hallo");
b.append("rld");
i= b.length();
b.insert(4,"o W"); // an Stelle 4 einfügen
b.delete(0,4); // Zeichen von 0-4 löschen
b.replace(1,3,"abc"); /* Zeichen 1-3 durch "abc"
                        ersetzen */
```

# Stringkonversionen

Oft müssen Strings in andere Datentypen konvertiert werden oder umgekehrt:

```
int i = Integer.parseInt( "123" );
```

```
float f = Float.parseFloat( "3.14" );
```

```
String s = String.valueOf( x );
```

```
char[] a = s.toCharArray();
```

# Nach dieser Einheit sollten Sie wissen, ...

- was der Unterschied zwischen primitiven und Objekt-Datentypen ist
- wie Objekt-Datentypen erzeugt, verwendet und als Parameter übergeben werden
- wie Strings in Java funktionieren