Grundlagen der Programmierung

VL15: Hilfs- / Hüllklassen

Prof. Dr. Samuel Kounev
Jóakim von Kistowski
Norbert Schmitt

Inhalt

Java

- Die Klassen **`String`** und **`StringBuffer`**
- Hüllklassen

Die Klasse `String`

Java

- Die Klasse **`String`** dient zur Darstellung von Zeichenketten
- Alle Zeichenketten-Literale in Java Programmen (z.B. **`"abc"`**) sind als Instanzen dieser Klasse realisiert
- Werte vom Typ String sind konstant, ihre Werte können nach ihrer Erzeugung nicht mehr verändert werden
- Es gibt verschiedene äquivalente Varianten zur Erzeugung von Strings:

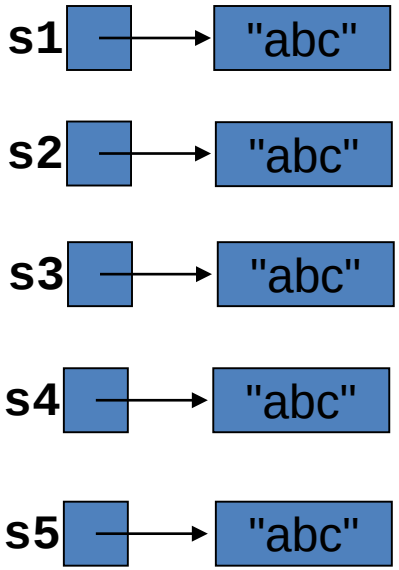
```
String s1 = "abc";                  // Variante 1

String s2 = new String("abc");      // Variante 2

char[] data = {'a', 'b', 'c'};      // Variante 3
String s3 = new String(data);

byte[] b = {97, 98, 99};            // Variante 4
String s4 = new String(b);

String s5 = new String(s4);         // Variante 5
```



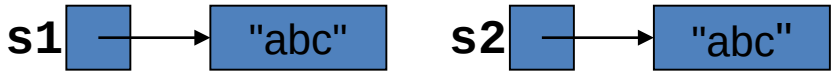
Die Klasse `String` (2)

Java

- Die Klasse **`String`** beinhaltet Methoden zum
  - Zugriff auf einzelne Zeichen der Zeichenkette
  - Vergleich von Zeichenketten
  - Suchen von Teil-Zeichenketten
  - Herausgreifen von Teil-Zeichenketten
  - Wandeln von Groß- in Kleinbuchstaben und umgekehrt
- Strings kann man mit dem Operator + konkatenieren (aneinanderhängen)
- Andere Objekte können in Strings umgewandelt werden
- Alle "Veränderungen" an einem String laufen so ab, dass jeweils ein **neues** String-Objekt geliefert wird!

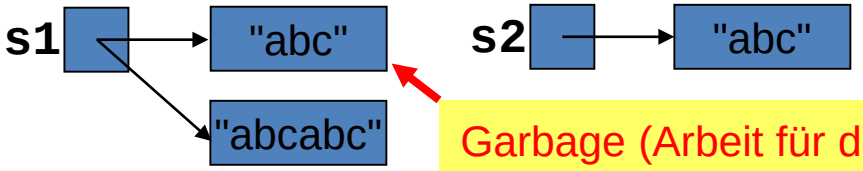
Beispiel

vorher:



Anweisung: **`s1 = s1 + s2`**

nachher:



Beispiele für String-Methoden

Java

```
public class StringTest {
 public static void main (String[] args) {
  String s1 = "Weihnachten";
  String s2 = "Veihnachten";
  String s3 = "Xeihnachten";
  String s4 = "WEIHNACHTEN";
  System.out.println(s1);
  System.out.println(s1.charAt(4));
  System.out.println(s1.compareTo(s1));
  System.out.println(s1.compareTo(s2));
  System.out.println(s1.compareTo(s3));
  System.out.println(s1.endsWith("ten"));
  System.out.println(s1.equals(s2));
  System.out.println(s1.equalsIgnoreCase(s4));
  System.out.println(s1.indexOf("n"));
  System.out.println(s1.indexOf("ach"));
  System.out.println(s1.length());
  System.out.println(s1.replace('e','E'));
  System.out.println(s1.startsWith("Weih"));
  System.out.println(s1.substring(3));
  System.out.println(s1.substring(3,7));
  System.out.println(s1.toLowerCase());
  System.out.println(s1.toUpperCase());
  System.out.println(String.valueOf(1.5e2));
 }
}
```

Ausgaben

```
Weihnachten
n
0
1
-1
true
false
true
4
5
11
WEihnachtEn
true
hnachten
hnac
weihnachten
WEIHNACHTEN
150.0
```

Beispiel: Löschen und Einfügen in Strings

Java

```
public static String insert(String old, int pos, String ins) {
  return old.substring(0,pos) + ins + old.substring(pos);
}

public static String delete(String old, int pos, int anzahl){
  return old.substring(0,pos) + old.substring(pos+anzahl);
}
```

Verwendung z.B. in der Form

```
s1 = insert(s1,4,"98"); // liefert s1 = "Weih98nachten"
s4 = delete(s4,4,5);    // liefert s4 = "WEIHEN„
```

Mit **`String`**-Klasse ineffizient, da bei jedem Aufruf neuer String erzeugt wird

Abhilfe: Klasse **`StringBuffer`**

Die Klasse `StringBuffer`...

Java

- ...stellt *veränderbare* Zeichenketten dar
- Beispiel

```
StringBuffer x = new StringBuffer();
x.append("A").append(4).append("-Seite");
```

- Im wesentlichen stellt die Klasse **`StringBuffer`** die Methoden **`append`** und **`insert`** zur Verfügung, die auf dem **`StringBuffer`**-Objekt selbst arbeiten (es muss also kein neuer String erzeugt werden). Die Methoden sind für fast alle elementaren Datentypen überladen.
- Die Anweisung

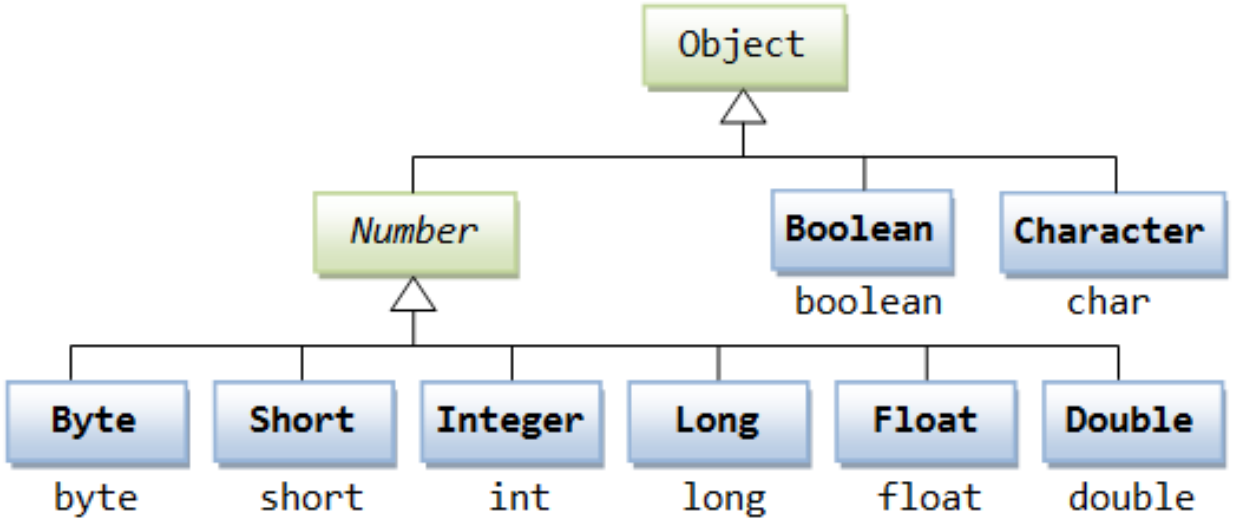
```
x = new StringBuffer("start").append("s").insert(4,"le");
```

sorgt also dafür, dass **`x`** den Wert **`"starlets"`** enthält.

Inhalt

Java

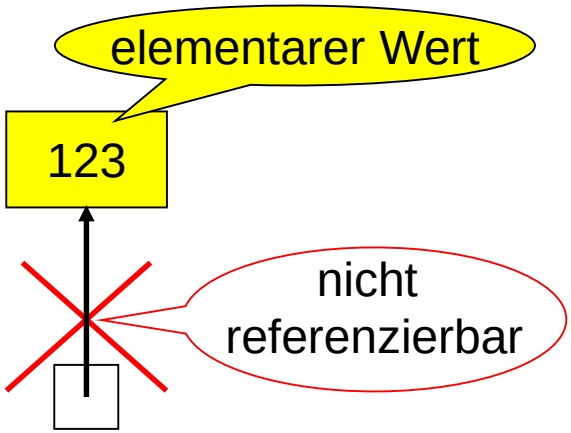
- Die Klassen **`String`** und **`StringBuffer`**
- **Hüllklassen**

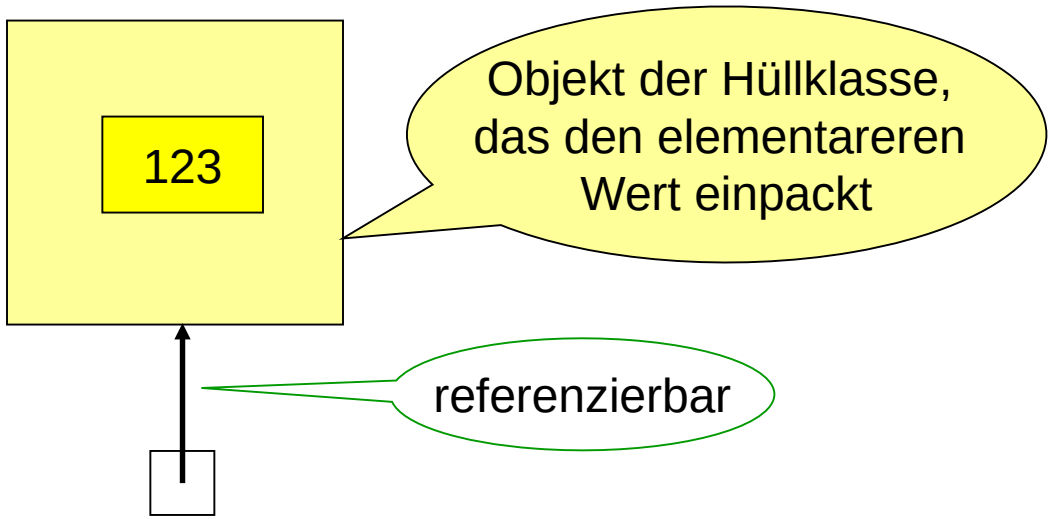


Hüllklassen (wrapper classes)

Java

- **Frage**: Können wir ein Feld anlegen, das in seinen Komponenten Werte von unterschiedlichen elementaren Datentypen speichern kann? → **Nein!**
- Andererseits könnten in einem Feld vom Typ **`Object[]`** Referenzen auf Objekte beliebiger Klassen gespeichert werden.
- Um Werte der elementaren Datentypen genauso handhaben zu können, muss man diese in Objekte "*einpacken*".
- Dazu stellt Java so genannte **Hüllklassen** **(wrapper classes)** zur Verfügung.





Hüllklassen für elementare Datentypen

Java

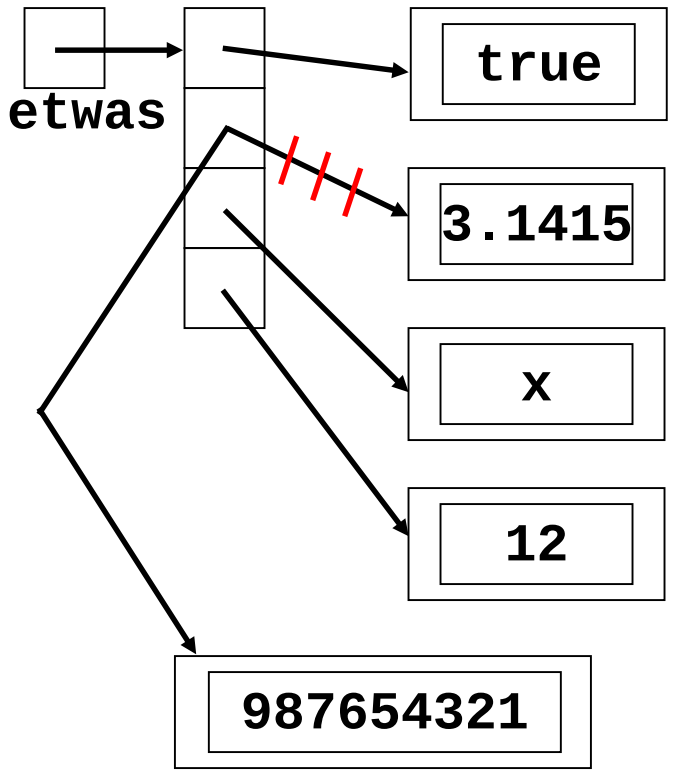
- Für elementare Datentypen existieren Hüllklassen mit ähnlichem Aufbau
  - **`Byte, Short, Integer, Long, Float, Double`**
  - **`Boolean`**
  - **`Character`**

elementarer Datentyp	Wrapper-Klasse	Konstruktoren
`byte`	`Byte`	`Byte(byte b)` `Byte(String s)`
`short`	`Short`	`Short(short s)` `Short(String s)`
`int`	`Integer`	`Integer(int i)` `Integer(String s)`
`long`	`Long`	`Long(long l)` `Long(String s)`
`float`	`Float`	`Float(float f)` `Float(String s)`
`double`	`Double`	`Double(double d)` `Double(String s)`
`boolean`	`Boolean`	`Boolean(boolean b)` `Boolean(String s)`
`char`	`Character`	`Character(char c)`

Hüllklassen - Beispiel

Java

```
public class WrapperBeispiel {
  public static void main (String[] args) {
    Object[] etwas = new Object[4];
    etwas[0] = new Boolean(true);
    etwas[1] = new Double(3.1415);
    etwas[2] = new Character('x');
    etwas[3] = new Integer(12);
    for (int i=0; i<4; i++)
      System.out.println(etwas[i]);
    etwas[1] = new Long(987654321);
    for (int i=0; i<4; i++)
      System.out.println(etwas[i]);
  }
}
```



Hüllklassen - Standard-Methoden

Java

- Alle Hüllklassen stellen die Klassenmethode **`valueOf(String s)`** bereit
- Außerdem gibt es die **`parseXxx-`** und die **`xxxValue`**-Methoden

Hüllklasse	parse-Methode	value-Methoden
`Byte`	`parseByte(String s)`	`byteValue(), shortValue(), intValue(), longValue(), floatValue(), doubleValue()`
`Short`	`parseShort(String s)`	
`Integer`	`parseInt(String s)`	
`Long`	`parseLong(String s)`	
`Float`	`parseFloat(String s)`	
`Double`	`parseDouble(String s)`	
`Boolean`		`booleanValue()`
`Character`		`charValue()`

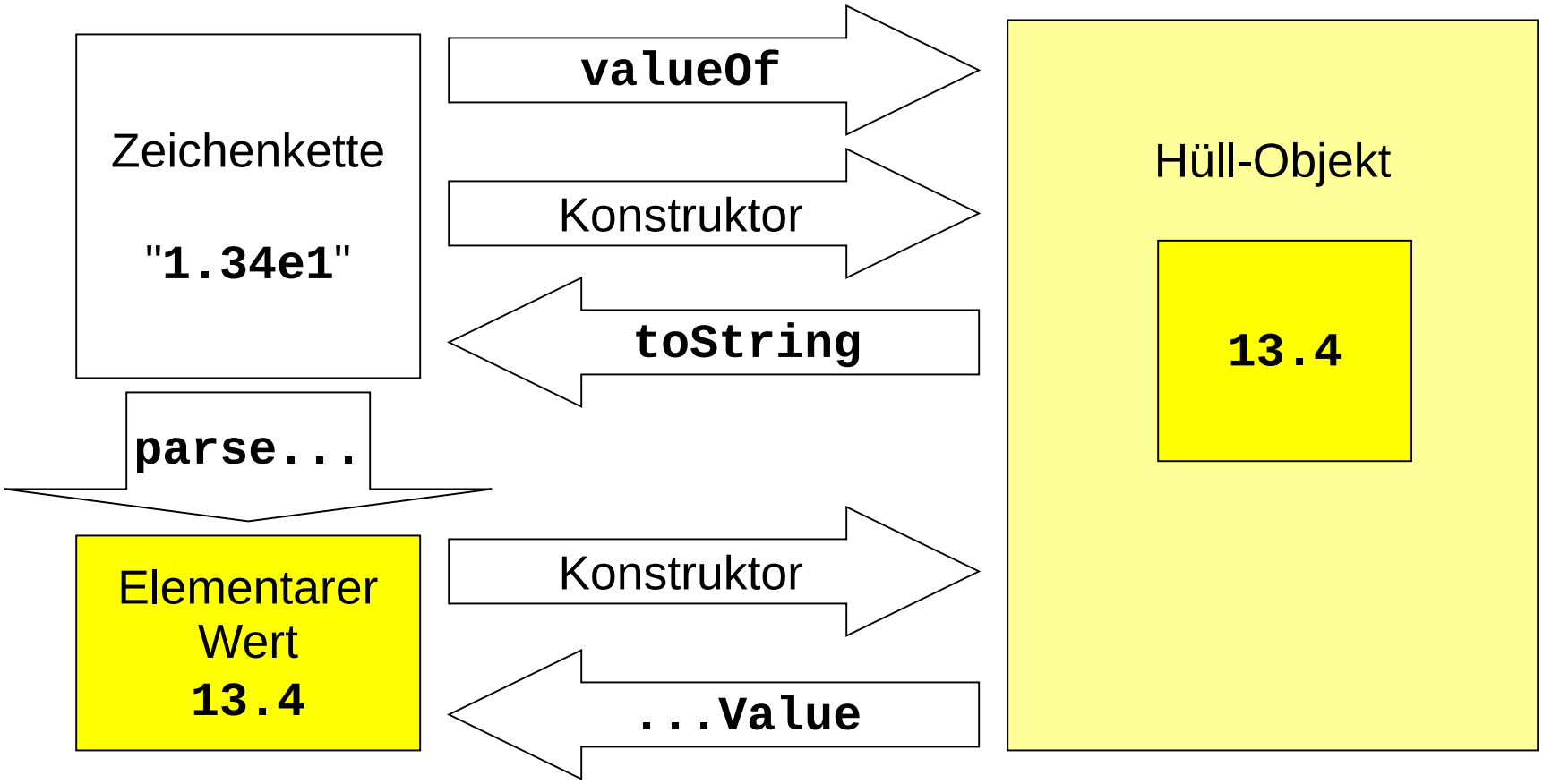
Hüllklassen - Beispiel

Java

- Aufbau am Beispiel **`Double`** als Hüllklasse zu **`double`**
  - je zwei Konstruktoren
    - `Double (double d)`
    - `Double (String s)`
  - Klassenmethode `Double valueOf(String s) {...}` wandelt einen String in ein `Double`-Objekt
  - Instanzmethode `double doubleValue() {...}` wandelt ein **`Double`**-Objekt in einen **`double`**-Wert
  - Klassenmethode `double parseDouble(String s) {...}` wandelt einen String in einen **`double`**-Wert
  - `double x = Double.valueOf("12.345").doubleValue();`
    entspricht also
    `double x = Double.parseDouble("12.345");`

Hüllklassen - Standard-Methoden

Java



Hüllklassen - Beispiel

Java

```
public class Summiere {
  public static void main(String[] summand) {
    int i = 0;
    double ergebnis = 0;
    for (i=0; i < summand.length; i++)
      ergebnis = ergebnis + Double.parseDouble(summand[i]);
    System.out.println("Ergebnis: " + ergebnis);
  }
}
```

```
java Summiere 1 2 3 4 5 6 7 8 9
```

```
Ergebnis: 45.0
```

```
java Summiere 1 2 x 4
```

```
Exception in thread "main"
java.lang.NumberFormatException: x
  at java.lang.FloatingDecimal...
  at java.lang.Double.parseDouble...
  at Summiere.main(Summiere.java:6)
```

unschön

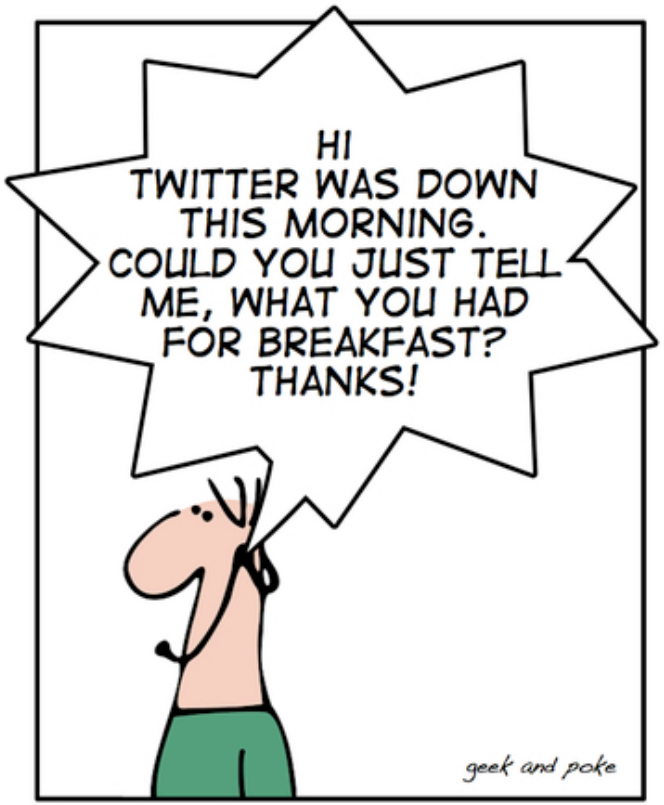
schöner wäre etwa

```
java Summiere 1 2 3 4 5 6 7 x 9
```

```
8. Summand unzulaessig!
```

Dazu ist eine Ausnahmebehandlung (siehe Teil Exceptions) notwendig!

Fragen?


HI
TWITTER WAS DOWN THIS MORNING. COULD YOU JUST TELL ME, WHAT YOU HAD FOR BREAKFAST? THANKS!
geek and poke
THE REAL FOLLOWERS

Danksagung

- Vorlesungsmaterialien von Prof. Dr. Detlef Seese wurden als Basis verwendet
- Unterstützung bei der technischen und inhaltlichen Gestaltung des Vorlesungsmaterials leisteten:

  Jóakim v. Kistowski

  Dietmar Ratz, Joachim Melcher, Roland Küstermann, Jana Weiner, Hagen Buchwald, Matthes Elstermann, Oliver Schöll, Niklas Kühl, Tobias Diederich