Informatik: Characters und Strings

Ziele

- Mit dem primitiven Datentyp `char` arbeiten können
- Die grundlegenden Zeichensätze kennen lernen
- Mit Sonderzeichen arbeiten können
- Mit Methoden der Klasse `Character` arbeiten können
- Den Unterschied zwischen Strings und Characters verstehen
- Strings mit den von Java bereitgestellten Methoden bearbeiten und vergleichen können

Characters

```
char c = 'a';
```

c | 'a'

```
if (c < 'b')
   System.out.println(10 + c);          // ergibt 10 + 97 = 107 'k'
```

implizite Typkonvertierung **char ➔ int** OK,

umgekehrt nicht möglich, aber:

```
char a = (char)97;                                  // ergibt 'a'
```

Zeichensätze

ASCII-Zeichensatz

128 Zeichen von 0 – 127 codiert

Ausgerichtet auf US-amerikanische Anwendungen

Keine Umlaute, kein €

ISO-Latin-1- oder ISO-8859-1-Zeichensatz

256 Zeichen, die ersten 128 identisch mit ASCII

Sammelt Sonderzeichen aus überwiegend westeuropäischen Sprachen

ISO-Latin15 identisch mit ISO-Latin-1 enthält €

Unicode

umfasst 65536 Zeichen, die ersten 256 identisch ISO-Latin-1

Enthält viele Alphabete aus der ganzen Welt

Java benutzt Unicode

Nicht alle Systeme können Unicodes korrekt wiedergeben. Deshalb…

Sonderzeichen

```
char a = '\u0061';                          // 0061 ist Hex-Wert von 97
char euro = '\u20AC';                                          // ergibt €
char hochkomma = '\'';          // Darstellung des Hochkommas ' selbst
char backslash = '\\';
char newline = '\n';
System.out.println("Hallo \"Leute\"");
```

Methoden der Klasse Character

aufgerufen über die Klasse selbst mit

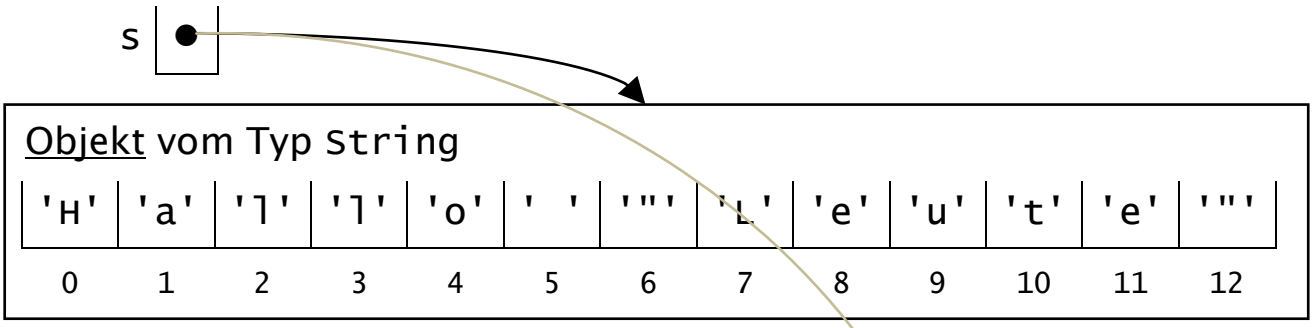
`Character.`Methodenname(Parameterliste)

```
if (Character.isLetter('\u0061')) …          // ergibt true
if (Character.isDigit('a')) …                // ergibt false
```

`public static boolean isLetter(char c)`	ist Buchstabe
`public static boolean isDigit(char c)`	ist Ziffer
`public static boolean isWhitespace(char c)`	ist Zwischenraum
`public static boolean isLowerCase(char c)` `public static boolean isUpperCase(char c)`	ist Kleinbuchstabe
`public static char toLowerCase(char c)` `public static char toUpperCase(char c)`	in Kleinbuchstabe

Strings

```
String s = "Hallo \"Leute\"";
```



- `String` kann beliebig viele **`char`**-Zeichen enthalten
- Einmal gefüllt, kann der Inhalt des Strings *nicht geändert* werden
- String leeren durch `s = null;`
- Strings *zusammenfügen* durch

```
int i = 3;
boolean b = true;
s = s +
   " wie gehts" + i + i + b;
```

```
Hallo "Leute" wie gehts33true
```

- *Elementzugriff* und *Länge*

```
char c = s.charAt(1);                                  // ergibt 'a'
c = s.charAt(-1);          // ergibt Fehler IndexOutOfBoundException
s = s.charAt(29);          // ergibt Fehler IndexOutOfBoundException
int l = s.length();                                    // ergibt 29
```

Methoden für Strings

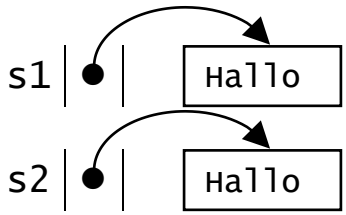
aufgerufen über das Objekt selbst mit

Objekt●Methodenname(Parameterliste)

```
String s = "Hallo Leute";
int anzahlGrossbuchstaben = 0;
for (int i = 0; i < s.length(); i = i + 1)
   if (Character.isUpperCase(s.charAt(i)))
      anzahlGrossbuchstaben = anzahlGrossbuchstaben + 1;

"  Hallo Leute   ".trim()           // ergibt "Hallo Leute"
"Hallo Leute".indexOf('e')         // ergibt 7
"Hallo Leute".indexOf('x')         // ergibt -1
"Hallo Leute".indexOf("Leu")       // ergibt 6
"Hallo Leute".indexOf("leu")       // ergibt -1
"Hallo Leute".substring(3, 7)      // ergibt "lo L"
"Hallo Leute".substring(3)         // ergibt "lo Leute"
```

Vergleichen von Strings




```
String s1 = "Hallo";
String s2 = "Hallo";
```

`if (s1 == s2) …`	`if (s1.equals(s2)) …`
vergleicht die Variablen	vergleicht Inhalt der Strings
↓	↓
false[1]	**true**

```
String s1 = "Hallo";
String s2 = "Hallo";
String s3 = "Jungs";
System.out.println(s1 == s2);          // ergibt false
System.out.println(s1.equals(s2));     // ergibt true
System.out.println(s1.compareTo(s2));  // ergibt 0
System.out.println(s1.compareTo(s3));  // ergibt -2
System.out.println(s3.compareTo(s1));  // ergibt 2
```

---

[1] Aus *„Programmieren mit Java“*, Pearson Studium: Die JVM optimiert den Platzbedarf und legt inhaltlich gleiche Strings bei Gelegenheit zusammen. Zulässig ist das wegen der Unveränderlichkeit von Stringobjekten. Der Vergleich gleich lautender Strings mit == liefert deshalb je nach Vorgeschichte der Operanden manchmal **true**, manchmal **false**.