II.2.5 Vordefinierte Klassen

Dienstag, 1. Dezember 2020 16:02

Hüllklassen + Strings

Hüllklassen

- Sind vordefiniert
- hüllen einen Wert eines primitiven Typs in einem Objekt ein
- Java-API beschreibt die nach außen sichtbaren Methoden und Attribute dieser Klassen.

Bei Integer:

- MIN_VALUE: ist -2^{31}
  MAX_VALUE: ist $2^{31}-1$
- Methoden, um Integer-Objekte zu erzeugen

```
Integer i = Integer.valueOf(5);
Integer j = Integer.valueOf("5");
Integer u = Integer.valueOf(300);
Integer v = Integer.valueOf("300");
```

```
i == j    ist  true
u == v    ist  false
```

← Integer-Objekte mit "kleinen" eingehüllten Werten werden wiederverwendet (zwischen -128 und 127).

=> inhaltliche Gleichheit von Objekten sollte man mit "equals" überprüfen

```
i.equals(j)  } ist true
u.equals(v)  }
```

- Methoden zur Wandlung zwischen int und String

- Methoden zur Wandlung zwischen int und String

```
int x = Integer.parseInt("123");
String s = Integer.toString(123);
```

- toString existiert auch als nicht-statische Methode für Integer-Objekte:

```
i.toString()
```

- get-Methode, um den eingehüllten int-Wert zu lesen:

```
i.intValue()   ist 5 als int-Wert
```

String

- vordefinierte Klasse
- Man kann neues Objekt erzeugen durch

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

- Da Strings so oft verwendet werden, gibt es Kurzschreibweise:

```
String s = "Wort";
```

- Kleiner Unterschied:
  Java "merkt sich", welche Strings in Kurzschreibweise erzeugt wurden und versucht, diese wiederzuverwenden.

```
String s = "Wort";
String t = "Wort";
String u = new String("Wort");
```

⇒ `s == t` ist true

`s == u` ist false

Strings sind "immutable", d.h. man kann String-Objekte nicht verändern.

Strings sind [illegible], d.h. [illegible] Objekte nicht
verändern.
D.h.: Es existieren keine Seiteneffekte, die problematisch sein
könnten, wenn s und t auf das gleiche Objekt zeigen.

```
String s = "Wort";
String t = "Wort";
s = s + "e";
```

← erzeugt ein neues String-Objekt "Worte"

=> t bleibt unverändert