Wrapper-Klassen in Java

Warum braucht man Wrapper-Klassen?

Primitive Datentypen wie int, double etc. sind keine Objekte. Viele Java-APIs wie Collections (z.B. List, Map) arbeiten nur mit Objekten.

Deshalb braucht man Wrapper-Klassen, um primitive Typen als Objekte verwenden zu koennen.

Gruende fuer Wrapper-Klassen

- Collections: List<int> geht nicht, aber List<Integer> geht.
- Methoden: Viele Methoden erwarten Objekte.
- Generics: Funktionieren nur mit Klassen, nicht mit primitiven Typen.
- Zusatzmethoden: z.B. Integer.parseInt(), Double.valueOf()

Uebersicht: Primitive Typen und Wrapper-Klassen

Primitive -> Wrapper-Klasse

int -> Integer

double -> Double

boolean -> Boolean

char -> Character

byte -> Byte

short -> Short

long -> Long

float -> Float

String in Wrapper oder Primitive parsen

```
String s = "42";
int i = Integer.parseInt(s);        // ergibt 42

String s2 = "3.14";
double d = Double.parseDouble(s2);  // ergibt 3.14
```

Wrapper-Klassen in Java

Boxing und Unboxing

```
Integer i = Integer.valueOf(42);  // Boxing (manuell)
int x = i.intValue();             // Unboxing (manuell)

List<Integer> list = new ArrayList<>();
list.add(5);                      // Autoboxing
int zahl = list.get(0);           // Autounboxing
```

Merksatz

Wrapper = die Bruecke zwischen primitiven Typen und objektbasierten APIs