Wrapper-Klassen & Autoboxing

Java Foundation Track by Carsten Bokeloh

Wrapper & Autoboxing

- Wrapperklassen
- Wrapper erzeugen
- Autoboxing
- Probleme beim Autoboxing

Wrapper Klassen

- Was sind Wrapper Klassen?
  - zwei wichtige Aufgaben:
    - Verschiedene Datenstrukturen in Java können nur können nur Objekte aufnehmen. Somit stellt sich das Problem, wie primitive Datentypen diesen Containern zugefügt werden können. Die Klassenbibliothek bietet deswegen für jeden primitiven Datentyp eine entsprechende Wrapper Klasse. Exemplare dieser Klassen kapseln je einen Wert des zugehörigen primitiven Typs.
    - Zusätzlich werden einige nützliche Methoden angeboten.
  - zu allen primitiven Datentypen existieren Wrapper-Klassen

Wrapperklassen

Ordinaltyp	Wrapperklasse
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Wrapper erzeugen

- drei Arten Sie zu erzeugen
  - Über Konstruktor der Wrapper-Klasse
  - Über statische valueOf-Methoden, denen ein primitiver Wert oder ein String übergeben wird.
  - Über Boxing: Aus einem primitiven Wert erstellt der Compiler mithilfe der valueOf-Methoden das Wrapper-Objekt.

```
Integer i1 = new Integer(29);
Integer i2 = Integer.valueOf(30);
Double db = new Double(12.3);
Boolean b1 = true;
```

Wrapperklassen

- Wrapper sind immutable
  - einmal erzeugt, kann der Wert nicht verändert werden
- Wrapper sind nur als Ummantelung und nicht als vollständige Datentypen gedacht. Wollen wir den Inhalt eines Integer-Objekts io um eins erhöhen, so müssen wir Folgendes schreiben:

```
int i = 12;
Integer io = new Integer(i);
io = new Integer(io.intValue()+1);
i = io.intValue();
```

Autoboxing

- seit Java 1.5 enthalten.
- das bedeutet, dass primitive Datentypen und Wrapperklassen bei Bedarf ineinander umgewandelt werden:

```
int i = 4711;
Integer j = i;//steht für j= Integer.valueOf(i)
int k = j; //steht für k=j.intValue();
```

Boxing

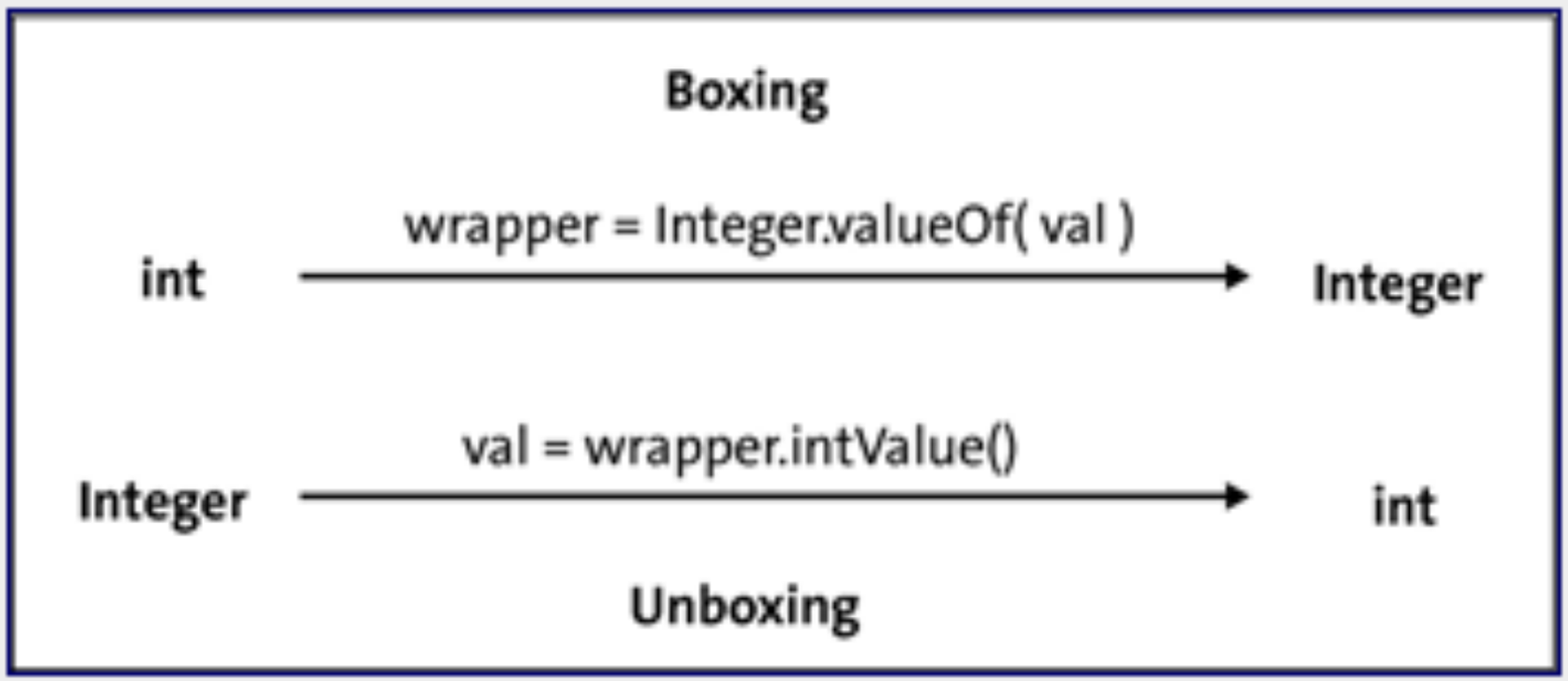
Unboxing

Autoboxing

- Compiler konvertiert nach festen Regeln und auch die Operatoren ++, - sind erlaubt.

```
Integer in = 12;
in = in + 1;
in++;
System.out.println(in);
```

Boxing / Unboxing



Arrays nicht konvertierbar

```
Character[] chars = { 'S', 'h', 'a' };
```

es gilt

```
char first = chars[ 0 ];
```

aber

```
char[] sha = chars;    // ☠ Compilerfehler!
```

Probleme beim Autoboxing

```
Integer i1 = 100;
Integer i2 = 100;

System.out.println(i1 >= i2); //true
System.out.println(i2 <= i1); //true
System.out.println(i1 == i2); //true
```

```
Integer i1 = 500;
Integer i2 = 500;

System.out.println(i1 >= i2); //true
System.out.println(i2 <= i1); //true
System.out.println(i1 == i2); //false
```

Probleme beim Autoboxing

- == weiterhin Referenzvergleich
  - kein Unboxing

Übung Autoboxing

```
public class AutoboxingM {
    public void schreiben(byte b) {
        System.out.println(b);
    }

    public static void main(String[] args) {
        AutoboxingM a = new AutoboxingM();
        a.schreiben(9);
    }
}
```

a: 9 **b:** Kompilierfehler **c:** Es wird eine Exception geworfen

Übung Wrapper

- Schreiben Sie ein Programm, das den größten gemeinsamen Teiler (ggT) von zwei positiven ganzen Zahlen berechnet. Die beiden Zahlen sollen als Kommandozeilenargumente übergeben werden. Zur Berechnung des ggt können Sie in einer Schleife jeweils die kleinere Zahl von der größeren abziehen, bis beide Zahlen gleich sind. Diese Zahl entspricht dann dem ggT.