Class Wrapper

Yuliana Setiowati

Tujuan dari Class Wrapper

- Class wrapper pada Java API mempunyai dua tujuan:
  - Menyediakan mekanisme untuk membungkus (wrap) nilai dengan tipe data primitif menjadi sebuah object, sehingga nilai primitif tadi bisa digunakan dalam kegiatan yang berhubungan dengan object, seperti menambahkan ke Collection, return type dari method berupa object. Di JDK 1.5 operasi wrapping dilakukan secara otomatis.
  - Untuk menyediakan fungsi-fungsi untuk mendukung tipe data primitif. Sebagian besar fungsi-fungsi ini berkaitan dengan berbagai konversi: konversi primitif ke dan dari objek String, dan mengkonversi primitif dan obyek String ke dan dari basis yang berbeda (atau radix), seperti biner, oktal, dan heksadesimal.

Class Wrapper

Primitive	Wrapper Class	Constructor Arguments
boolean	Boolean	boolean or String
byte	Byte	byte or String
char	Character	char
double	Double	double or String
float	Float	float, double, or String
int	Integer	int or String
long	Long	long or String
short	Short	short or String

Membuat Object Wrapper

- Semua class wrapper kecuali Character menyediakan dua constructor. Argument berupa String dan nilai primitif

```
Integer i1 = new Integer(42);
Integer i2 = new Integer("42");

or

Float f1 = new Float(3.14f);
Float f2 = new Float("3.14f");
```

```
Character c1 = new Character('c');
```

- Pada Java 5 object Boolean dapat digunakan pada kondisi, karena kompiler secara otomatis “un-box” (membuka) Boolean menjadi boolean.

```
Boolean b = new Boolean("false");
if (b)      // won't compile, using Java 1.4 or earlier
```

Setiap object wrapper mempunyai konstanta MAX_VALUE

```
byteObj = new Byte(Byte.MAX_VALUE);
shortObj = new Short(Short.MAX_VALUE);
intObj = new Integer(Integer.MAX_VALUE);
longObj = new Long(Long.MAX_VALUE);
floatObj = new Float(Float.MAX_VALUE);
doubleObj = new Double(Double.MAX_VALUE);

printNumValues("MAXIMUM NUMBER VALUES:");
```

Setiap object wrapper mempunyai konstanta MAX_VALUE

=>

Byte:127

Short:32767

Integer:2147483647

Long:9223372036854775807

Float:3.4028235E38

Double:1.79769313486231570E308

Class Wrapper
Method valueOf()

- **Method valueOf() mengubah suatu nilai menjadi object dari class tersebut.**
- Class Long, Integer, Short dan Byte mempunyai tiga method valueOf()

static Integer	**valueOf**(int i) Returns a Integer instance representing the specified int value.
static Integer	**valueOf**(String s) Returns an Integer object holding the value of the specified String.
static Integer	**valueOf**(String s, int radix) Returns an Integer object holding the value extracted from the specified String when parsed with the radix given by the second argument.

- Method 1 menerima argument berupa nilai.
- Method 2 menerima argument nilai dalam bentuk String.
- Method 3 argument kedua berupa int radix yang menyatakan base dari argument pertama (binary, octal, atau hexadecimal)

Class Wrapper
Method valueOf()

- Class Boolean, Float dan Double mempunyai dua method valueOf()

static Double	**valueOf**(double d) Returns a Double instance representing the specified double value.
static Double	**valueOf**(String s) Returns a Double object holding the double value represented by the argument string s.

- Sedangkan untuk Character hanya mempunyai satu method valueOf()

static Character	**valueOf**(char c) Returns a Character instance representing the specified char value.

Class Wrapper
Method valueOf()

```
Integer i2 = Integer.valueOf("101011", 2);  // converts 101011
                                            // to 43 and
                                            // assigns the value
                                            // 43 to the
                                            // Integer object i2
```

Or

```
Float f2 = Float.valueOf("3.14f");  // assigns 3.14 to the
                                    // Float object f2
```

Integer i1 = Integer.valueOf(42);
Integer i2 = Integer.valueOf(“42”);

Boolean b1 = Boolean .valueOf(true);
Boolean b2 = Boolean .valueOf(“true”);

Long n1 = Long.valueOf(42000000L);
Long n1 = Long.valueOf(“42000000L”);

Class Wrapper
xxxValue()

- Digunakan untuk mengubah object dari class wrapper (object ini mempunyai nilai) menjadi nilai numerik

Class Wrapper - xxxValue()

```
Integer i2 = new Integer(42);  // make a new wrapper object
byte b = i2.byteValue();       // convert i2's value to a byte
                               // primitive
short s = i2.shortValue();     // another of Integer's xxxValue
                               // methods
double d = i2.doubleValue();   // yet another of Integer's
                               // xxxValue methods
```

or

```
Float f2 = new Float(3.14f);   // make a new wrapper object
short s = f2.shortValue();     // convert f2's value to a short
                               // primitive
System.out.println(s);         // result is 3  (truncated, not
                               // rounded)
```

Class Wrapper

parseXxx() and valueOf() ?

- Fungsi parseXxx() dan valueOf(), argument berupa String dan melempar NumberFormatException(NFE) jika argument tidak sesuai.

static int	**parseInt**(String s) Parses the string argument as a signed decimal integer.
static int	**parseInt**(String s, int radix) Parses the string argument as a signed integer in the radix specified by the second argument.

- Perbedaan:
- parseXxx() mengembalikan nilai primitif.
- valueOf() mengembalikan object dari class wrapper

Class Wrapper

Here are some examples of these methods in action:

```
double d4 = Double.parseDouble("3.14");    // convert a String
                                           // to a primitive
System.out.println("d4 = " + d4);          // result is d4 = 3.14

Double d5 = Double.valueOf("3.14");        // create a Double obj
System.out.println(d5 instanceof Double); // result is "true"
```

The next examples involve using the radix argument (in this case binary):

```
long L2 = Long.parseLong("101010", 2);     // binary String to a
                                           // primitive
System.out.println("L2 = " + L2);          // result is: L2 = 42

Long L3 = Long.valueOf("101010", 2);       // binary String to
                                           // Long object
System.out.println("L3 value = " + L3);    // result is:
                                           // L3 value = 42
```

Class Wrapper
toString()

- untuk memungkinkan mendapatkan representasi yang bermakna dari object tertentu.

```
Double d = new Double("3.14");
System.out.println("d = "+ d.toString() ); // result is d = 3.14

String d = Double.toString(3.14);     // d = "3.14"

String s = "hex = "+ Long.toString(254,16); // s = "hex = fe"
```

Class Wrapper

toXxxString() (Binary, Hexadecimal, Octal)

- Fungsi tsb untuk mengubah bilangan dengan basis 10 menjadi basis lainnya

static String	**toBinaryString**(int i) Returns a string representation of the integer argument as an unsigned integer in base 2.
static String	**toHexString**(int i) Returns a string representation of the integer argument as an unsigned integer in base 16.
static String	**toOctalString**(int i) Returns a string representation of the integer argument as an unsigned integer in base 8.

- Contoh:

```
String s3 = Integer.toHexString(254);  // convert 254 to hex System.out.println("254 is " + s3);

String s4 = Long.toOctalString(254); // convert 254 to octal
System.out.print("254(oct) ="+ s4);  // result: "254(oct) =376"
```

Class Wrapper Autoboxing/Auto-unboxing

- Pada Java 5, dikenal istilah autoboxing
- Autoboxing adalah konversi secara otomatis oleh kompiler Java dari tipe data primitif ke tipe data sesuai dengan tipe wrappernya (misalnya, int dan Integer, double dan Double, dll)..
- Sedangkan mengubah object dari class wrapper menjadi nilai primitifnya disebut auto-unboxing
- Contoh

With Autoboxing	Without Autoboxing

`int i;`
`Integer j;`
`i = 1;`
`j = 2;`
`i = j;`
`j = i;`
	`int i;`
	`Integer j;`
	`i = 1;`
	`j = new Integer(2);`
	`i = j.intValue();`
	`j = new Integer(i);`

Contoh

```
Integer y = new Integer(567);      // make it
int x = y.intValue();              // unwrap it
x++;                               // use it
y = new Integer(x);                // re-wrap it
System.out.println("y = " + i);    // print it
```

Now, with new and improved Java 5 you can say

```
Integer y = new Integer(567);      // make it
y++;                               // unwrap it, increment it,
                                   // rewrap it
System.out.println("y = " + i);    // print it
```

Both examples produce the output:

```
y = 568
```

Class Wrapper Autoboxing

Let's try this:

```
Integer y = 567;                    // make a wrapper
Integer x = y;                      // assign a second ref
                                    // var to THE wrapper

System.out.println(y==x);           // verify that they refer
                                    // to the same object

y++;                                // unwrap, use, "rewrap"
System.out.println(x + " " + y);    // print values

System.out.println(y==x);           // verify that they refer
                                    // to different objects
```

Which produces the output:

```
true
567 568
false
```

So, under the covers, when the compiler got to the line `i++;` it had to substitute something like this:

```
int x2 = y.intValue();          // unwrap it
x2++;                           // use it
y = new Integer(x2);            // re-wrap it
```

Just as we suspected, there's gotta be a call to `new` in there somewhere.

Class Wrapper Autoboxing

- Mengapa object x dan y tidak mengacu pada alamat yang sama ?
  - Karena object dari class Wrapper bersifat immutable artinya kekal. Sekali object dicreate maka isi objek tidak bisa diubah.
  - Jika isi objek diubah maka objek harus mengacu ke alamat yang berbeda.

```
int x2 = y.intValue();          // unwrap it
x2++;                           // use it
y = new Integer(x2);            // re-wrap it
```

Class Wrapper
Boxing, ==, and Equals()

- Mengapa i1 and i2 dikatakan object yang berbeda, sedangkan i3 and i4 dikatakan object yang sama ?
  - Untuk menghemat memory, dua object dari class wrapper selalu == pada saat nilai primitif:
    - Boolean
    - Byte
    - Character from \u0000 to \u007f (7f is 127 in decimal)
    - Short and Integer from -128 to 127

```
Integer i1 = 1000;
Integer i2 = 1000;
if(i1 != i2) System.out.println("different objects");
if(i1.equals(i2)) System.out.println("meaningfully equal");
```

Produces the output:

```
different objects
meaningfully equal
```

```
Integer i3 = 10;
Integer i4 = 10;
if(i3 == i4) System.out.println("same object");
if(i3.equals(i4)) System.out.println("meaningfully equal"
```

This example produces the output:

```
same object
meaningfully equal
```

Class Wrapper
Boxing

```
class UseBoxing {
  public static void main(String [] args) {
    UseBoxing u = new UseBoxing();
    u.go(5) ;
  }

  boolean go(Integer i) {          // boxes the int it was passed
    Boolean ifSo = true;           // boxes the literal
    Short s = 300;                 // boxes the primitive

    if(ifSo) {                     // unboxing
      System.out.println(++s);     // unboxes, increments, reboxes
    }
    return !ifSo;                  // unboxes, returns the inverse
  }
}
```