Wrapper Classes

By
M. BABY ANUSHA,
ASST.PROF IN CSE DEPT.,
RGUKT,NUZVID

INTRODUCTION

Java uses primitive types, such as int, char, double to hold the basic data types supported by the language.

Sometimes it is required to create an object representation of these primitive types.

These are collection classes that deal only with such objects. One needs to wrap the primitive type in a class.

To satisfy this need, java provides classes that correspond to each of the primitive types. Basically, these classes encapsulate, or wrap, the primitive types within a class.

Thus, they are commonly referred to as type wrapper. Type wrapper are classes that encapsulate a primitive type within an object.

The wrapper types are **Byte, Short, Integer, Long, Character, Boolean, Double, Float.**

These classes offer a wide array of methods that allow to fully integrate the primitive types into Java's object hierarchy.

WRAPPER CLASS

- Wrapper classes are classes that allow primitive types to be accessed as objects.

- Wrapper class is wrapper around a primitive data type because they "wrap" the primitive data type into an object of that class.

WHAT IS WRAPPER CLASS?

- Each of Java's eight primitive data types has a class dedicated to it.
- They are one per primitive type: Boolean, Byte, Character, Double, Float, Integer, Long and Short.
- Wrapper classes make the primitive type data to act as objects.

PRIMITIVE DATA TYPES AND WRAPPER CLASSES

Data Type	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
char	Character
float	Float
double	Double
boolean	Boolean

Converting primitive numbers to Object numbers using constructor methods

Constructor calling	Conversion Action
Integer IntVal = new Integer(i);	Primitive integer to Integer object
Float FloatVal = new Float(f);	Primitive float to Float object
Double DoubleVal = new Double(d);	Primitive double to Double object
Long LongVal = new Long(l);	Primitive long to Long object

Converting Numeric Strings to Primitive numbers using Parsing method

Method calling	Conversion Action
int i = Integer.parseInt(str);	Converts String str into primitive integer i
long l = Long.parseLong(str);	Converts String str into primitive long l

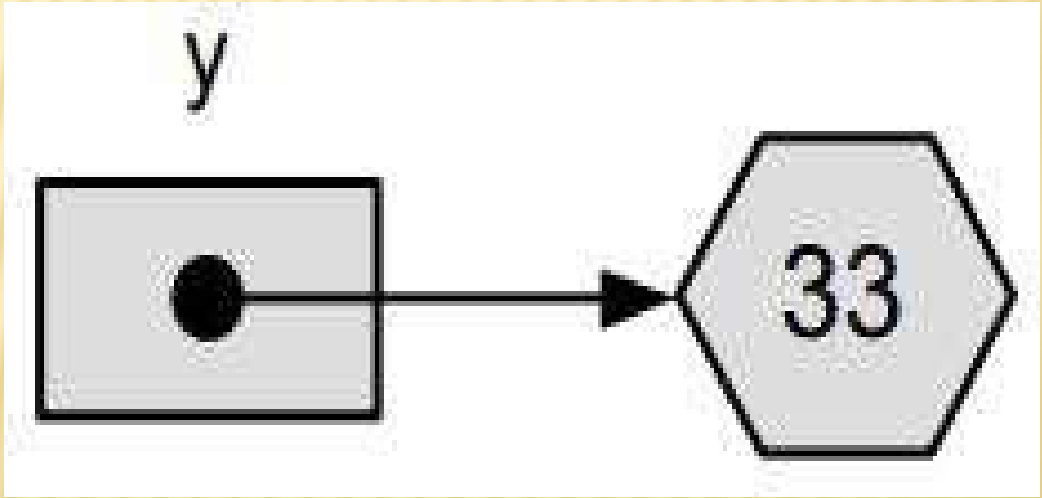
Difference b/w Primitive Data Type and Object of a Wrapper Class

- The following two statements illustrate the difference between a primitive data type and an object of a wrapper class:

```
int x = 25;

Integer y = new Integer(33);
```

- The second statement instantiates an Integer object.
- The object is initialized with the value 33 and a reference to the object is assigned to the object variable y.



- Clearly x and y differ by more than their values:
- x is a variable that holds a value.
- y is an object variable that holds a reference to an object.

WHAT IS THE NEED OF WRAPPER CLASSES

- Wrapper classes are used to be able to use the primitive data- types as objects.
- Many utility methods are provided by wrapper classes.

To get these advantages we need to use wrapper classes.

BOXING AND UNBOXING :

- The wrapping is done by the compiler.
- if we use a primitive where an object is expected, the compiler boxes the primitive in its wrapper class.
- Similarly, if we use a number object when a primitive is expected, the compiler un-boxes the object.

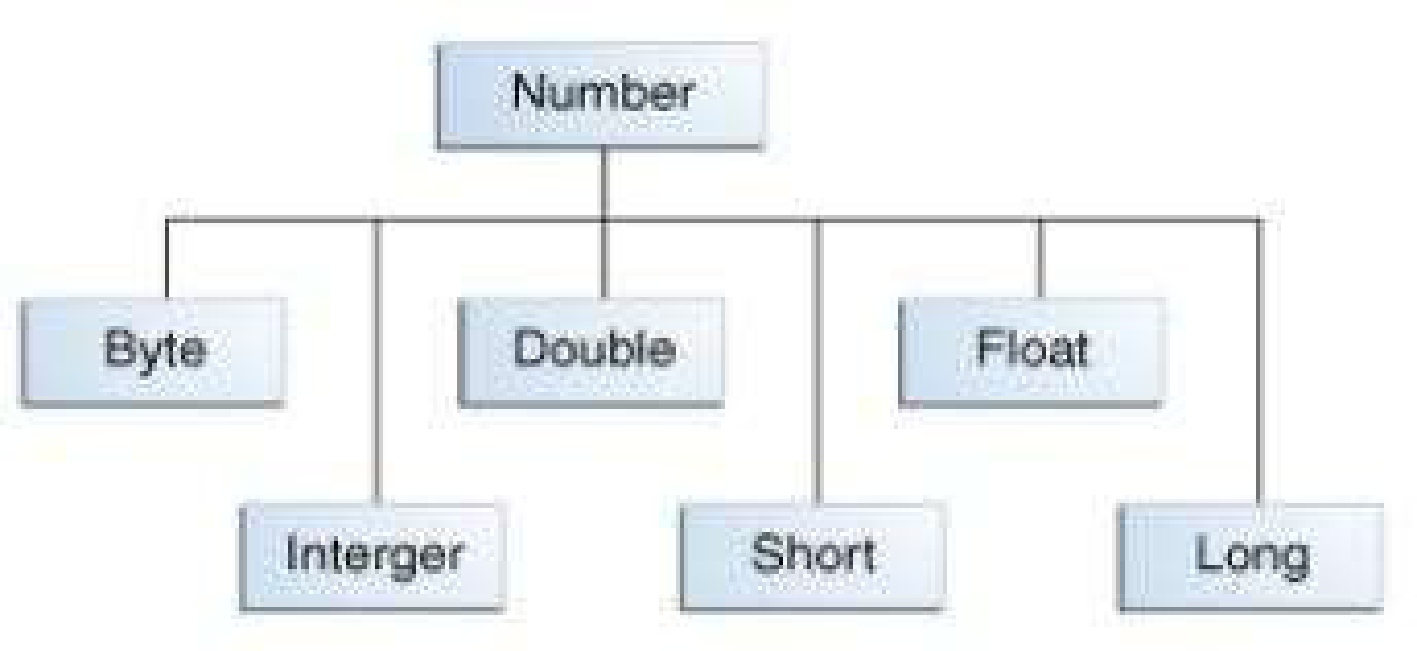
Example of boxing and unboxing:

- Integer x, y; x = 12; y = 15;System.out.println(x+y);
- When x and y are assigned integer values, the compiler boxes the integers
  because x and y are integer objects.
- In the println() statement, x and y are unboxed so that they can be added as integers.

NUMERIC WRAPPER CLASSES :

- All of the numeric wrapper classes are subclasses of the abstract class Number .

- Short, Integer, Double and Long implement Comparable interface.



The Byte class encapsulates a byte value. It defines the constants MAX_VALUE and MIN_VALUE and provides these constructors:

Byte(**byte b**)
Byte(**String str**)

Here, b is a byte value and str is the string equivalent of a byte value.

EX :

```
import java.util.*;
public class Byte_Demo
{
   public static void main(String args[])
   {
      Byte b1 = new Byte((byte)120);
      for(int i = 125; i<=135; i++)
      {
         Byte b2 = new Byte((byte)i);
         System.out.println("b2 = " + b2);
      }
      System.out.println("b1 Object = " + b1);
      System.out.println("Minimum Value of Byte = " + Byte.MIN_VALUE);
      System.out.println("Maximum Value of Byte = " + Byte.MAX_VALUE);
      System.out.println("b1* 2 = " + b1*2);
      System.out.println("b1* 2 = " + b1.byteValue()*2);
      Byte b3 = new Byte("120");
      System.out.println("b3 Object = " + b3);
      System.out.println("(b1==b3)? " + b1.equals(b3));
      System.out.println("(b1.compareTo(b3)? " + b1.compareTo(b3));
       /*Returns 0 if equal. Returns -1 if b1 is less than b3 and 1 if b1 is
      greater than 1*/
   }
}
```

Output :

```
b2 = 125
b2 = 126
b2 = 127
b2 = -128
b2 = -127
b2 = -126
b2 = -125
b2 = -124
b2 = -123
b2 = -122
b2 = -121
b1 Object = 120
Minimum Value of Byte = -128
Maximum Value of Byte = 127
b1* 2 = 240
b1* 2 = 240
b3 Object = 120
(b1==b3)? true
(b1.compareTo(b3)? 0
```

The Short class encapsulates a short value. It defines the constants **MAX_VALUE** and **MIN_VALUE** and provides the following constructors:

Short(**short s**)
Short(**String str**)

EX :

```
import java.util.*;
public class Short_Demo
{
   public static void main(String args[])
   {
      Short s1 = new Short((short)2345);
      for(int i = 32765; i<=32775; i++)
      {
         Short s2 = new Short((short)i);
         System.out.println("s2 = " + s2);
      }
      System.out.println("s1 Object = " + s1);
      System.out.println("Minimum Value of Short = " + Short.MIN_VALUE);
      System.out.println("Maximum Value of Short = " + Short.MAX_VALUE);
      System.out.println("s1* 2 = " + s1.shortValue()*2);
      Short s3 = new Short("2345");
      System.out.println("s3 Object = " + s3);
      System.out.println("(s1==s3)? " + s1.equals(s3));
      Short s4 = Short.valueOf("10", 16);
      System.out.println("s4 Object = " + s4);
   }
}
```

Output :

```
s2 = 32765
s2 = 32766
s2 = 32767
s2 = -32768
s2 = -32767
s2 = -32766
s2 = -32765
s2 = -32764
s2 = -32763
s2 = -32762
s2 = -32761
s1 Object = 2345
Minimum Value of Short = -32768
Maximum Value of Short = 32767
s1* 2 = 4690
s3 Object = 2345
(s1==s3)? true
s4 Object = 16
```

Integer class :

The Integer class encapsulates an integer value. This class provides following constructors:

Integer(**int i**)
Integer(**String str**)

Here, i is a simple int value and str is a String object.

EX :

```
import java.util.*;
public class Int_Demo
{
   public static void main(String args[])
   {
      Integer i1 = new Integer(12);
      System.out.println("I1 = " + i1);
      System.out.println("Binary Equivalent = " + Integer.toBinaryString(i1));
      System.out.println("Hexadecimal Equivalent = " + Integer.toHexString(i1));
      System.out.println("Minimum Value of Integer = " + Integer.MIN_VALUE);
      System.out.println("Maximum Value of Integer = " + Integer.MAX_VALUE);
      System.out.println("Byte Value of Integer = " + i1.byteValue());
      System.out.println("Double Value of Integer = " + i1.doubleValue());
      Integer i2 = new Integer(12);
      System.out.println("i1==i2 " + i1.equals(i2));
      System.out.println("i1.compareTo(i2) = " + i2.compareTo(i1));
      // Compareto - if it is less than it returns -1 else 1, if equal it return 0.
      Integer i3 = Integer.valueOf("11", 16);
      System.out.println("i3 = " + i3);
   }
}
```

Output :

```
I1 = 12
Binary Equivalent = 1100
Hexadecimal Equivalent = c
Minimum Value of Integer = -2147483648
Maximum Value of Integer = 2147483647
Byte Value of Integer = 12
Double Value of Integer = 12.0
i1==i2 true
i1.compareTo(i2) = 0
i3 = 17
```

Long :

The Long class encapsulates a long value. It defines the constants MAX_VALUE and MIN_VALUE and provides the following constructors:

Long(**long l**)
Long(**String str**)

EX :

```
import java.util.*;
 public class Long_Demo
{
   public static void main(String args[])
   {
      Long L1 = new Long(68764);
      Long L2 = new Long("686748");
      System.out.println("Object L1 = " + L1);
      System.out.println("Object L2 = " + L2);
      System.out.println("Minimum Value of Long = " + Long.MIN_VALUE);
      System.out.println("Maximum Value of Long = " + Long.MAX_VALUE);
      System.out.println("L1 * 2 = " + L1 * 2);
      System.out.println("L1.longValue() * 2 = " + L1.longValue() * 2);
      System.out.println("L1.compareTo(l2) = " + L1.compareTo(L2));
      System.out.println("L1==L2 ? = " + L1.equals(L2));
      Long L3 = Long.valueOf("10", 16);
      System.out.println("Object L3 = " + L3);
      System.out.println("Byte value of Long = " + L1.byteValue());
      System.out.println("int value of Long = " + L1.intValue());
      System.out.println("Double value of Long = " + L1.doubleValue());
      int i = 12;
      System.out.println("Binary equivalent of decimal " + i + "=" + Long.toBinaryString(i));
      System.out.println("Hexadecimal equivalent of decimal " + i + "=" + Long.toHexString(i));

   }
}
```

Output :

```
Object L1 = 68764
Object L2 = 686748
Minimum Value of Long = -9223372036854775808
Maximum Value of Long = 9223372036854775807
L1 * 2 = 137528
L1.longValue() * 2 = 137528
L1.compareTo(l2) = -1
L1==L2 ? = false
Object L3 = 16
Byte value of Long = -100
int value of Long = 68764
Double value of Long = 68764.0
Binary equivalent of decimal 12=1100
Hexadecimal equivalent of decimal 12=c
```

Double class :

The Double class encapsulates a double value. It defines several constants.

The largest and smallest values are saved in MAX_VALUE and MIN_VALUE.

The constant NaN (Not a Number) indicates that a value is not a number.

If you divide a double number by zero, the result is NaN. This class defines these constructors:

Double(**double d**)
Double(**String str**)

Here, d is a double value to be encapsulated in a Double object. In the last form, str is the string representation of a double value.

EX :

```
import java.util.*;
class Double_Demo
{
   public static void main(String args[])
   {
      Double d1 = new Double(687642365.4563);
      Double d2 = new Double("686748");
      System.out.println("Object d1 = " + d1);
      System.out.println("Object d2 = " + d2);
      System.out.println("Minimum Value of Double = " + Double.MIN_VALUE);
      System.out.println("Maximum Value of Double = " + Double.MAX_VALUE);
      System.out.println("Byte value of Double = " + d1.byteValue());
      System.out.println("int value of Double = " + d1.intValue());
      System.out.println("Float value of Double = " + d1.floatValue());
      Double d3 = Double.parseDouble("765.89");
      System.out.println("Double value from the string \"765.89\"="+d3);

   }
}
```

Output :

```
Object d1 = 6.876423654563E8
Object d2 = 686748.0
Minimum Value of Double = 4.9E-324
Maximum Value of Double = 1.79769313486231 57E308
Byte value of Double = -3
int value of Double = 687642365
Float value of Double = 6.8764237E8
Double value from the string "765.89"=765.89
```

Float class :

The float class encapsulates a float value. It defines several constants the largest and smallest values are stored in MAX_VALUE and MIN_VALUE.

The constant NaN indicates that a value is not a number. If you divide a floating – point number by zero, the result is NaN.

This class defines these constructors:

Float(**float f**)
Float(**double d**)
Float(**String str**)

Here, f and d are float and double types to be encapsulated in a Float object.

str is the string representation of a float value.

EX :

```
import java.util.*;

public class Float_Demo
{
   public static void main(String args[])
   {
      Float f1 = new Float(123.5626);
      Float f2 = new Float(854.32f);
      Float i = new Float(10);
      System.out.println("Object f1 = " + f1);
      System.out.println("Object f2 = " + f2);
      System.out.println("Minimum Value of Float = " + Float.MIN_VALUE);
      System.out.println("Maximum Value of Float = " + Float.MAX_VALUE);
      System.out.println("Byte value of Float = " + f1.byteValue());
      System.out.println("Short value of Float = " + f1.shortValue());
      System.out.println("Integer value of Float = " + f1.intValue());
      System.out.println("Double value of Float = " + f1.doubleValue());
      System.out.println("(f1==f2) ?= " + f1.equals(f2));
      System.out.println("f1.compareTo(f2) = " + f1.compareTo(f2));
      System.out.println("f1 is not a number = " + i.isNaN());

   }
}
```

Output :

```
Object f1 = 123.5626
Object f2 = 854.32
Minimum Value of Float = 1.4E-45
Maximum Value of Float = 3.4028235E38
Byte value of Float = 123
Short value of Float = 123
Integer value of Float = 123
Double value of Float = 123.5625991821289
(f1==f2) ?= false
f1.compareTo(f2) = -1
f1 is not a number = false
```

Character class :

The Character class encapsulates a char value. This class provides the following constructor.

Character(**char ch**)

Here, c is a char value. charValue() method returns the char value that is encapsulated by a Character object and has the following form:

char charValue()

EX :

```
import java.util.*;

public class Char_Demo
{
   public static void main(String args[])
   {
      Character c1 = new Character('m');
      char c2 = 'O';
      System.out.println("Object C1 = " + c1);
      System.out.println("char value of Character Object = " + c1.charValue());
      System.out.println(c2 + " is defined character set ? " + Character.isDefined(c2));
      System.out.println("c2 is digit = " + Character.isDigit(c2));
      System.out.println("c2 is lowercase character = " + Character.isLowerCase(c2));
      System.out.println("c2 is uppercase character = " + Character.isUpperCase(c2));


   }
}
```

Output :

Object C1 = m
char value of Character Object = m
O is defined character set ? true
c2 is digit = false
c2 is lowercase character = false
c2 is uppercase character = true

Boolean class :

The Boolean class encapsulates a Boolean value. It defines FALSE and TRUE constants.

This class provides following constructors:

Boolean(**Boolean b**)
Boolean(**String str**)

Here, b is a Boolean value and str is the string equivalent of a Boolean value.

The methods associated with Boolean Class are as follows:

1. **Boolean booleanValue()**
2. **Boolean equals(Boolean b)**
3. **String toString(Boolean b)**

EX :

```
import java.util.*;

public class Boolean_Demo
{
   public static void main(String args[])
   {
      Boolean b1 = new Boolean(true);
      Boolean b2 = new Boolean(false);
      System.out.println("Object B1 = " + b1);
      System.out.println("Object B2 = " + b2);
      Boolean b3 = new Boolean("true");
      Boolean b4 = new Boolean("false");
      System.out.println("Object B3 = " + b3);
      System.out.println("Object B4 = " + b4);
      System.out.println("Boolean Value = " + b1.booleanValue());
      System.out.println("(b1==b2)? " + b1.equals(b2));
      String S1 = b1.toString();
      System.out.println("String S1 " + S1);

   }
}
```

Output :

Object B1 = true
Object B2 = false
Object B3 = true
Object B4 = false
Boolean Value = true
(b1==b2)? false
String S1 true


Thank you!
shutterstock.com · 567687052