Wrapper classes,Autoboxing and Unboxing in Java

-K.L.Madhavi

Wrapper Classes in Java

A Wrapper class in Java is a class whose object wraps or contains primitive data types. When we create an object to a wrapper class, it contains a field and in this field, we can store primitive data types. In other words, we can wrap a primitive value into a wrapper class object.

Need of Wrapper Classes:

1. They convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method (because primitive types are passed by value).

2. The classes in java.util package handles only objects and hence wrapper classes help in this case also.

3. Data structures in the Collection framework, such as ArrayList and Vector , store only objects (reference types) and not primitive types.

4. object is needed to support synchronization in multithreading.

Primitive Data Types and their Corresponding Wrapper Class

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

Java Wrapper Classes Example

```
import java.io.*;

class Wrapper {
  public static void main(String[] args)
  {
    byte a = 1;
    // wrapping around Byte object
    Byte byteobj = new Byte(a);

    int b = 10;
    // wrapping around Integer object
    Integer intobj = new Integer(b);

    float c = 18.6f;
    // wrapping around Float object
    Float floatobj = new Float(c);

    double d = 250.5;
    // Wrapping around Double object
    Double doubleobj = new Double(d);

    char e = 'a';
    // wrapping around Character object
    Character charobj = e;
```

```
        // printing the values from objects
        System.out.println( "Values of Wrapper objects (printing as objects)");
        System.out.println("\nByte object byteobj: "+ byteobj);
        System.out.println("\nInteger object intobj: "+ intobj);
        System.out.println("\nFloat object floatobj: "+ floatobj);
        System.out.println("\nDouble object doubleobj: "+ doubleobj);
        System.out.println("\nCharacter object charobj: "+ charobj);

        // objects to data types (retrieving data types from objects) unwrapping objects to primitive data types
        byte bv = byteobj;
        int iv = intobj;
        float fv = floatobj;
        double dv = doubleobj;
        char cv = charobj;

        // printing the values from data types
        System.out.println("\nUnwrapped values (printing as data types)");
        System.out.println("\nbyte value, bv: " + bv);
        System.out.println("\nint value, iv: " + iv);
        System.out.println("\nfloat value, fv: " + fv);
        System.out.println("\ndouble value, dv: " + dv);
        System.out.println("\nchar value, cv: " + cv);
    }
}
```

Output:

```
Values of Wrapper objects (printing as objects)
Byte object byteobj: 1
Integer object intobj: 10
Float object floatobj: 18.6
Double object doubleobj: 250.5
Character object charobj: a
Unwrapped values (printing as data types)
byte value, bv: 1
 int value, iv: 10
float value, fv: 18.6
double value, dv: 250.5
char value, cv: a
```

1. Autoboxing

The automatic conversion of primitive types to the object of their corresponding wrapper classes is known as autoboxing. For example – conversion of int to Integer, long to Long, double to Double, etc.

2.Unboxing

It is just the reverse process of autoboxing. Automatically converting an object of a wrapper class to its corresponding primitive type is known as unboxing. For example – conversion of Integer to int, Long to long, Double to double, etc.

Program for Auto-Boxing and Un-Boxing

```
public class Demo
{
    public static void main(String args[])
     {
          int num=7;
          //here we are getting a value from primitive type to object type.
          //we are taking a primitive value and storing it in a wrapper object.
          Integer num1=new Integer(num);  //boxing
          Integer number1=num;   //autoboxing

         //here we are getting a value from object type to primitive type
         //now assigning again to a primitive value
         int num2=num1.intValue();  //Unboxing
         int number2=num1;  //Auto-Unboxing
         System.out.println(num1);
         System.out.println(number1);
         System.out.println(num2);
         System.out.println(number2);

       }
}
```