Numbers

(Wrapper Classes in Java)

By Dr. Subrat Kumar Nayak

Associate Professor

Department of CSE

ITER, SOADU

Wrapper Classes in Java

Wrapper class:

- The **wrapper class in Java** provides the mechanism *to convert primitive into object and object into primitive*.

Why Wrapper class?

- Java is an object-oriented programming language, so we need to deal with objects many times like in Collections, Serialization, Synchronization, etc.

1. Change the value in Method:

Java supports only call by value. So, if we pass a primitive value, it will not change the original value.

2. Serialization:

We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.

3. Synchronization:

Java synchronization works with objects in Multithreading.

4. java.util package:

The java.util package provides the utility classes to deal with objects.

5. Collection Framework:

Java collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, etc.) deal with objects only.

Wrapper Classes in Java

- The list of primitive types and their corresponding Wrapper classes.

Built-in type	Object wrapper	Size of built-in (bits)	Contents
byte	Byte	8	Signed integer
short	Short	16	Signed integer
int	Integer	32	Signed integer
long	Long	64	Signed integer
float	Float	32	IEEE-754 floating point
double	Double	64	IEEE-754 floating point
char	Character	16	Unsigned Unicode character
n/a	BigInteger	unlimited	Arbitrary-size immutable integer value
n/a	BigDecimal	unlimited	Arbitrary-size-and-precision immutable floating-point value

- Along with these, there is another wrapper class for boolean primitive data type, i.e. Boolean wrapper class
- The **java.lang.Number** class is the superclass of classes BigDecimal, BigInteger, Byte, Double, Float, Integer, Long, and Short.The Subclasses of Number must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

Wrapper Classes in Java

Constructors:

- Every wrapper class in java has two constructors,

(a) First constructor takes corresponding primitive data as an argument

(b) Second constructor takes string as an argument.

Notes:

- The string passed to second constructor should be parse-able to number , otherwise you will get run time **NumberFormatException**.
- Wrapper Class Character has only one constructor which takes char type as an argument. It doesn't have a constructor which takes String as an argument. Because, String can not be converted into Character.
- Wrapper class Float has three constructors. The third constructor takes double type as an argument.

Examples:

```
Integer I1 = new  Integer(30);
//Constructor which takes int value as an argument

Integer I2 = new  Integer("30");
//Constructor which takes String as an argument
```

Wrapper Classes in Java

Examples…

```
Short S1 = new Short((short) 20);   //Constructor which takes short value as an argument
Short S2 = new Short("10");         //Constructor which takes String as an argument


Float F1 = new Float(12.2f); //Constructor which takes float value as an argument
Float F2 = new Float("15.6");//Constructor which takes String as an argument
Float F3 = new Float(15.6d); //Constructor which takes double value as an argument


Character C1 = new Character('D'); //Constructor which takes char value as an argument
Character C2 = new Character("a");
//Compile time error : String a can not be converted to character
```

- Converting primitive data types into object is called **boxing**, and this is taken care by the compiler.
- Similarly, the Wrapper object can also be converted back to a primitive data type, and this process is called **unboxing**.

- The **automatic conversion** of primitive data type into its corresponding wrapper class is known as **auto-boxing**.
- The **automatic conversion** of wrapper type into its corresponding primitive type is known as **auto-unboxing**.

Wrapper Classes in Java

Examples for Auto-boxing & Auto-unboxing:

```
public class WrapperExample1{
public static void main(String args[]){
int a=20;
Integer j=a;//auto-boxing, now compiler will call Integer.valueOf(a) internally
System.out.println(j);// will print 20
}}
public class WrapperExample2{
public static void main(String args[]){
//Converting Integer to int
Integer a=new Integer(10);
int j=a;//auto-unboxing, now compiler will call a.intValue() internally
System.out.println(j);// will print 10
}}
```

Wrapper Classes in Java

Number Methods:

Following is the list of the instance methods that **all the subclasses** of the Number class implements.

xxxValue()

Converts the value of *this* Number object to the **xxx** data type and returns it.

Syntax:

xxx xxxValue()

- xxx stands for different primitive data type.

Example:

```
public class Test {
public static void main(String args[]) {
     Integer x = 5;
     // Returns byte primitive data type
     System.out.println(x.byteValue());
     // Returns double primitive data type
     System.out.println(x.doubleValue());
     // Returns long primitive data type
     System.out.println(x.longValue());
     //will not work x.charValue()for Integer.
} }
```

```
Output:
5
5.0
5
```

Wrapper Classes in Java

compareTo()

- The method compares the Number object that invoked the method to the argument. It is possible to compare Byte, Long, Integer, etc.
- However, two different types cannot be compared, both the argument and the Number object invoking the method should be of the same type.

Syntax:

```
public int compareTo(NumberSubClass referenceName)
```

Return Value:

- If this object equal to the argument then 0 is returned.
- If this object less than the argument then -1 is returned.
- If this object greater than the argument then 1 is returned.

Example:

```
public class Test {
public static void main(String args[]) {
     Integer x = 5;
     System.out.println(x.compareTo(3));
     System.out.println(x.compareTo(5));
     System.out.println(x.compareTo(8));
} }
```

```
Output:
1
0
-1
```

Wrapper Classes in Java

valueOf()

- The valueOf method returns the relevant Number Object holding the value of the argument passed. The argument can be a primitive data type, String, etc.
- This method is a **static** method. The method can take two arguments, where one is a String and the other is a radix.

Syntax:

```
static Xxx valueOf(xxx i)// works for all Wrapper type

//xxx: primitive type of corresponding wrapper type

static Xxx valueOf(String s)// does not work for Character

static Xxx valueOf(String s, int radix)// not applicable for Character,
Double,Float.

//Xxx represents any of the subclass of Number class(Integer, Float etc.)
```

Return Values:

- **valueOf(int i)** – This returns an Xxx object holding the value of the specified primitive.
- **valueOf(String s)** – This returns an Xxx object holding the value of the specified string representation.
- **valueOf(String s, int radix)** – This returns an Xxx object holding the integer value of the specified string representation, parsed with the value of radix.

Wrapper Classes in Java

Example:

```
public class Test {
public static void main(String args[]) {
    Integer x =Integer.valueOf(9);
    Double c = Double.valueOf(5);
    Float a = Float.valueOf("80");
    Integer b = Integer.valueOf(“0111",2);
    System.out.println(x);
    System.out.println(c);
    System.out.println(a);
    System.out.println(b);
} }
```

```
Output:
9
5.0
80.0
7
```

End of Session