# Wrapper classes in Java

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

Since J2SE 5.0, **autoboxing** and **unboxing** feature convert primitives into objects and objects into primitives automatically. The automatic conversion of primitive into an object is known as autoboxing and vice-versa unboxing.

## Use of Wrapper classes in Java

Java is an object-oriented programming language, so we need to deal with objects many times like in Collections, Serialization, Synchronization, etc. Let us see the different scenarios, where we need to use the wrapper classes.

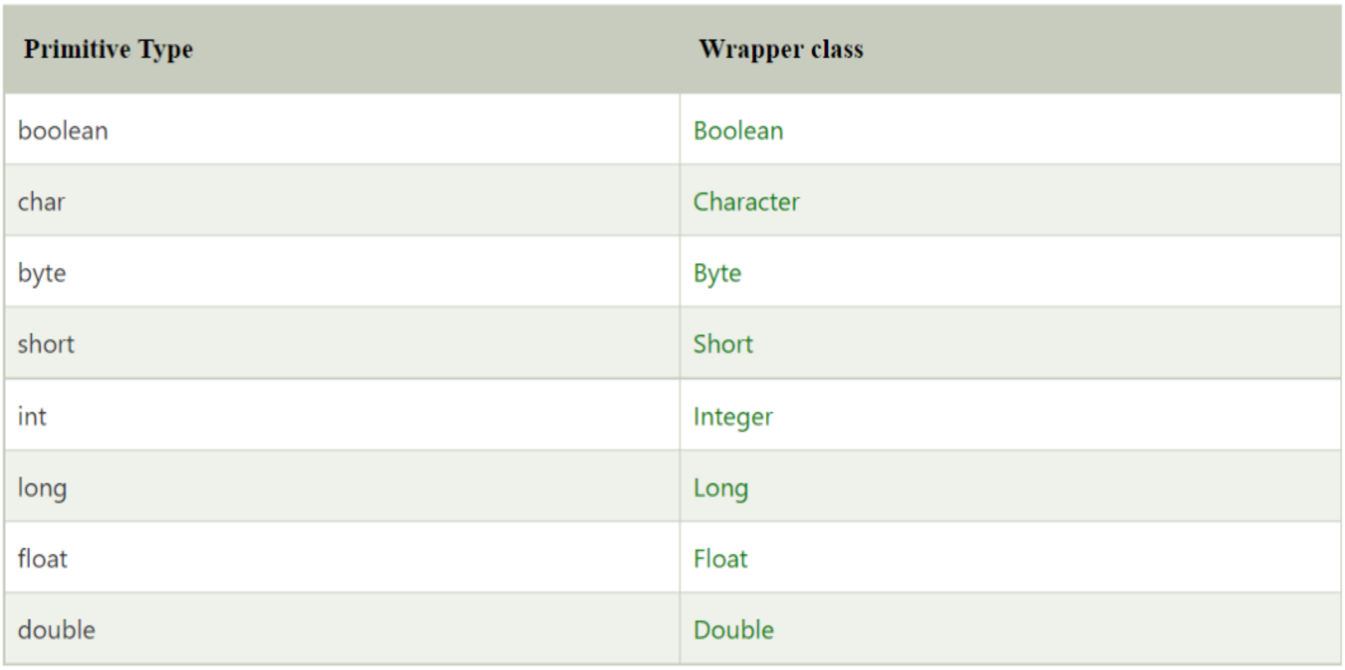
- **Change the value in Method:** Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value.
- **Serialization:** We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.
- **Synchronization:** Java synchronization works with objects in Multithreading.
- **java.util package:** The java.util package provides the utility classes to deal with objects.
- **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.

Sometimes you must use wrapper classes, for example when working with Collection objects, such as ArrayList, where primitive types cannot be used (the list can only store objects):

Example

```
ArrayList<int> myNumbers = new ArrayList<int>(); // Invalid


ArrayList<Integer> myNumbers = new ArrayList<Integer>(); // Valid
```

**The eight classes of the *java.lang* package ae known as wrapper classes in Java. The list of eight wrapper classes are given below:**



| Primitive Type | Wrapper class |
|---|---|
| boolean | Boolean |
| char | Character |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | Float |
| double | Double |

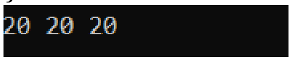
## Autoboxing

The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing, for example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short.

Since Java 5, we do not need to use the valueOf() method of wrapper classes to convert the primitive into objects.

## Example

```
//Java program to convert primitive into objects

//Autoboxing example of int to Integer

public class AutoBoxing
{
public static void main(String args[])
{

//Converting int into Integer
int a=20;
Integer i=Integer.valueOf(a);//converting int into Integer explicitly

Integer j=a;//autoboxing, now compiler will write Integer.valueOf(a) internally

System.out.println(a+" "+i+" "+j);

}
}
```



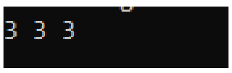
```
20 20 20
```

## Unboxing

The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing. It is the reverse process of autoboxing. Since Java 5, we do not need to use the intValue() method of wrapper classes to convert the wrapper type into primitives.

### Example

```
//Java program to convert object into primitives
//Unboxing example of Integer to int

public class UnBoxing
{
public static void main(String args[])
{

//Converting Integer to int

Integer a=new Integer(3);

int i=a.intValue();//converting Integer to int explicitly

int j=a;//unboxing, now compiler will write a.intValue() internally

System.out.println(a+" "+i+" "+j);

}
}
```



```
3 3 3
```

## Creating Wrapper Objects

***To** **create a wrapper object, use the wrapper class instead of the primitive type. To get the value, you can just print the object:***

Example

```
public class Main {

  public static void main(String[] args) {

    Integer myInt = 5;

    Double myDouble = 5.99;

    Character myChar = 'A';

    System.out.println(myInt);

    System.out.println(myDouble);

    System.out.println(myChar);

  }

}
```

Since you're now working with objects, you can use certain methods to get information about the specific object.

For example, the following methods are used to get the value associated with the corresponding wrapper
object: intValue(), byteValue(), shortValue(), longValue(), floatValue(), double Value(), charValue(), booleanValue().

**This example will output the same result as the example above:**

Example

```
public class Main {

  public static void main(String[] args) {

    Integer myInt = 5;

    Double myDouble = 5.99;

    Character myChar = 'A';

    System.out.println(myInt.intValue());

    System.out.println(myDouble.doubleValue());

    System.out.println(myChar.charValue());}}
```

Another useful method is the toString() method, which is used to convert wrapper objects to strings.

In the following example, we convert an Integer to a String, and use the length() method of the String class to output the length of the "string":

Example

```java
public class Main {
  public static void main(String[] args) {
    Integer myInt = 100;
    String myString = myInt.toString();
    System.out.println(myString.length());
  }
}
```