Wrapper class in Java

BY CHAITANYA SINGH | FILED UNDER: LEARN JAVA

In the OOPs concepts guide, we learned that object oriented programming is all about objects. The eight primitive data types byte, short, int, long, float, double, char and boolean are not objects, **Wrapper classes are used for converting primitive data types into objects**, like int to Integer etc. Lets take a simple example to understand why we need wrapper class in java.

For example: While working with collections in Java, we use generics for type safety like this: ArrayList<Integer> instead of this ArrayList<int>. The Integer is a wrapper class of int primitive type. We use wrapper class in this case because generics needs objects not primitives. There are several other reasons you would prefer a wrapper class instead of primitive type, we will discuss them as well in this article.

Primitive	Wrapper class
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer

long	Long
float	Float
double	Double

Why we need wrapper class in Java

1. As I mentioned above, one of the reason why we need wrapper is to use them in collections API. On the other hand the wrapper objects hold much more memory compared to primitive types. So use primitive types when you need efficiency and use wrapper class when you need objects instead of primitive types.

The primitive data types are not objects so they do not belong to any class. While storing in data structures which support only objects, it is required to convert the primitive type to object first which we can do by using wrapper classes.

Example:

```
HashMap<Integer, String> hm = new HashMap<Integer, String>();
```

So for type safety we use wrapper classes. This way we are ensuring that this HashMap keys would be of integer type and values would be of string type.
2. Wrapper class objects allow null values while primitive data type doesn't allow it.

Lets take few examples to understand how the conversion works:

Wrapper Class Example 1: Converting a primitive type to Wrapper object

```
public class JavaExample{
   public static void main(String args[]){
        //Converting int primitive into Integer object
        int num=100;
```

```
        Integer obj=Integer.valueOf(num);

        System.out.println(num+ " "+ obj);
   }
}
```

Output:

```
100 100
```

As you can see both primitive data type and object have same values. You can use obj in place of num wherever you need to pass the value of num as an object.

Wrapper Class Example 2: Converting Wrapper class object to Primitive

```
public class JavaExample{
   public static void main(String args[]){
        //Creating Wrapper class object
        Integer obj = new Integer(100);

        //Converting the wrapper object to primitive
        int num = obj.intValue();

        System.out.println(num+ " "+ obj);
   }
}
```

Output:

```
100 100
```