Java Wrapper Classes

Java Wrapper Classes

Wrapper classes provide a way to use primitive data types (`int` , `boolean` , etc..) as objects.

The table below shows the primitive type and the equivalent wrapper class:

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

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
```

Try it Yourself »

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);
  }
}
```

Try it Yourself »

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()`, `doubleValue()`, `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());
  }
}
```

Try it Yourself »

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

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