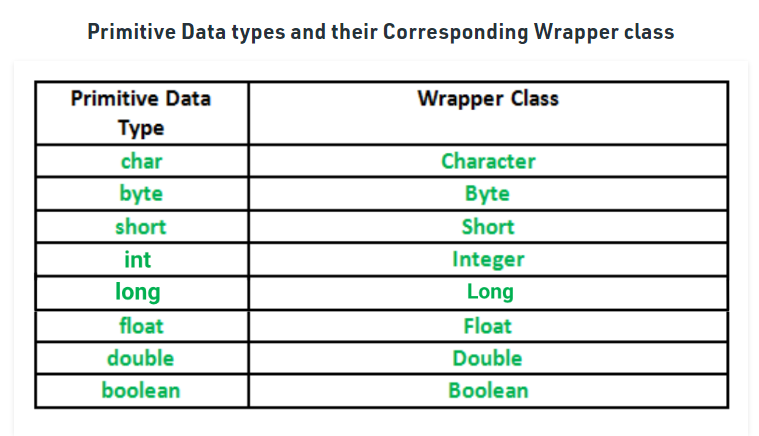
**Wrapper classes in Java**

A Wrapper class 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.

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

**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. An object is needed to support synchronization in multithreading.



**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.

**Wrapper class Example: Primitive to Wrapper**

**Integer I = 10; is auto boxing internally this statement is converted into**

**Integer I = Integer.valueOf(10);**

**Auto unboxing**

**Integer I = new Integer(10);**

**Int i = I; 🡺 int i = Integer.intValue(); // Auto unboxing**