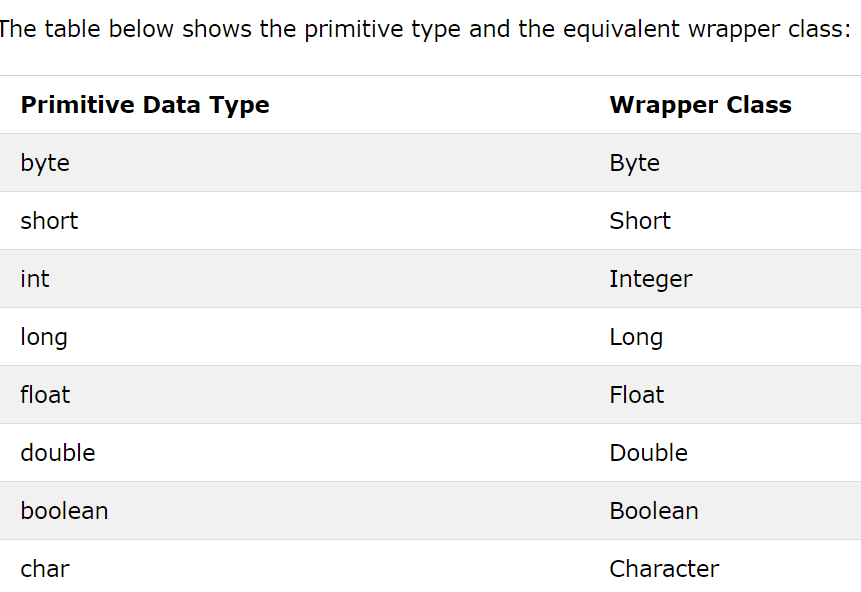
Wrapper class is used to convert Primitive data types into Objects.

**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](https://www.geeksforgeeks.org/arraylist-in-java/) and [Vector](https://www.geeksforgeeks.org/vector-vs-arraylist-java/), store only objects (reference types) and not primitive types.
4. An object is needed to support synchronization in multithreading.



**For example when working with Collection objects, such as ArrayList, where primitive types cannot be used (the list can only store objects):**

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

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

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:

public class MyClass {

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(), doubleValue(), charValue(), booleanValue().

**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":

Integer myInt = 100;

String myString = myInt.toString();

System.out.println(myString.length());

**Autoboxing and Unboxing**

**Autoboxing:** Automatic conversion of primitive types to the object of their corresponding wrapper classes is known as autoboxing. For example – conversion of int to Integer, long to Long, double to Double etc.

**char** ch = 'a';

//Autoboxing - Primitive to Character Object conversion

Character a = ch;

ArrayList<Integer> al=**new** ArrayList<Integer>();

// Autoboxing because ArrayList stores only objects

al.add(200);

//Printing the value from object.

System.***out***.println(al.get(0));

**Unboxing:** It is just the reverse process of autoboxing. Automatically converting an object of a wrapper class to its corresponding primitive type is known as unboxing. For example – conversion of Integer to int, Long to long, Double to double etc.

// Java program to demonstrate Unboxing

import java.util.ArrayList;

class Unboxing

{

    public static void main(String[] args)

    {

        Character ch = 'a';

        // unboxing - Character object to primitive conversion

        char a = ch;

        ArrayList<Integer> arrayList = new ArrayList<Integer>();

        arrayList.add(24);

        // unboxing because get method returns an Integer object

        int num = arrayList.get(0);

        // printing the values from primitive data types

        System.out.println(num);

    }

}