A **Wrapper class** is a class which contains the **primitive data types** (**int, char, short, byte, etc)**. In other words, wrapper classes provide a way to use **primitive data types (int, char, short, byte, etc) as objects**. These wrapper classes come under **java.util package**.

Why we need Wrapper Class

- Wrapper Class will **convert primitive data types into objects**. The objects are necessary if we wish to modify the arguments passed into the method (because primitive types are **passed by value**).
- The classes in **java.util package** handles only objects and hence **wrapper classes** help in this case also.
- **Data structures** in the Collection framework such as **ArrayList and Vector** store only the objects (reference types) and not the **primitive types.**
- The object is needed to support **synchronization** in **multithreading**.

Firstly the question that hits the programmers is when we have primitive data types then why does there arise a need for the concept of wrapper classes in java. It is because of the additional features being there in the Wrapper class over the primitive data types when it comes to usage. These methods include primarily methods like *valueOf()*, *parseInt()*, *toString()*, and many more.

A wrapper class wraps (encloses) around a data type and gives it an object appearance. Wrapper classes are final and immutable. Two concepts are there in the wrapper classes namely autoboxing and unboxing.

Autoboxing is a procedure of converting a primitive value into an object of the corresponding wrapper class. For example, converting int to Integer class. The Java compiler applies autoboxing when a primitive value is:

- Passed as a parameter to a method that **expects an object** of the corresponding wrapper class.
- Assigned to a variable of the corresponding **wrapper class**.

Unboxing is a procedure of converting an object of a wrapper type to its corresponding primitive value. For example conversion of Integer to int. The Java compiler applies to unbox when an object of a wrapper class is

- Passed as a parameter to a method that **expects a value** of the corresponding primitive type.
- Assigned to a variable of the corresponding **primitive type**.

Implementation of the wrapper class in Java

Autoboxing in Wrapper Class

Autoboxing is used to **convert primitive data types into corresponding objects**.

```
public class AutoBoxingTest {
  public static void main(String args[]) {
    int num = 10; // int primitive
    Integer obj = Integer.valueOf(num); // creating a wrapper class object
    System.out.println(num + " " + obj);
  }
}
```

Output

```
10 10
```

Unboxing in Wrapper Class

Unboxing is used to **convert the Wrapper class object into corresponding primitive data types**.

```
public class UnboxingTest {
  public static void main(String args[]) {
    Integer obj = new Integer(10); // Creating Wrapper class object
    int num = obj.intValue(); // Converting the wrapper object to primitive datatype
    System.out.println(num + " " + obj);
  }
}
```

Output

```
10 10
```