**Wrapper Classes in Java**

A Wrapper class in Java 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

**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.

**Advantages of Wrapper Classes**

1. Collections allowed only object data.
2. On object data we can call multiple methods compareTo(), equals(), toString()
3. Cloning process only objects
4. Object data allowed null values.
5. Serialization can allow only object data.

Below are given examples of wrapper classes in Java with their corresponding Primitive data types in Java.

**Primitive Data types and their Corresponding Wrapper Class**

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

**Autoboxing and Unboxing**

**1. Autoboxing**

**The automatic conversion of primitive types to the object of their corresponding wrapper classes is known as autoboxin**g. For example – conversion of int to Integer, long to Long, double to Double, etc.

**Example:**

Java

|  |
| --- |
| // Java program to demonstrate Autoboxing    import java.util.ArrayList;  class Autoboxing {      public static void main(String[] args)      {          char ch = 'a';            // Autoboxing- primitive to Character object          // conversion          Character a = ch;            ArrayList<Integer> arrayList              = new ArrayList<Integer>();            // Autoboxing because ArrayList stores only objects          arrayList.add(25);            // printing the values from object          System.out.println(arrayList.get(0));      }  } |

**Output**

25

**2. 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.

**Example:**

Java

|  |
| --- |
| // 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);      }  } |

**Output**

24

## FAQs in Wrapper Class

### 1. Which are the wrapper classes in Java?

A Wrapper class in Java is a class whose object wraps or contains primitive data types.

### 2. Why use the wrapper class in Java?

The wrapper class in Java is used to convert primitive data types into objects. Objects are needed if we wish to modify the arguments passed into a method.

### 3. What are the 8 wrapper classes in Java?

There are 8 Wrapper classes in Java these are char, byte, short, int, long, float, double and boolean.