# Java Wrapper Classes

## **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 |

## **Need of Wrapper Classes**

There are certain needs for using the Wrapper class in Java as mentioned below:

* 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).
* 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](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.
* An object is needed to support synchronization in multithreading.
* **Change the value in Method:** Java supports only call by value. So, if we pass a primitive value, it will not change the original value. But, if we convert the primitive value in an object, it will change the original value.
* **Serialization:** We need to convert the objects into streams to perform the serialization. If we have a primitive value, we can convert it in objects through the wrapper classes.
* **Synchronization:** Java synchronization works with objects in Multithreading.
* **java.util package:** The java.util package provides the utility classes to deal with objects.
* **Collection Framework:** Java collection framework works with objects only. All classes of the collection framework (ArrayList, LinkedList, Vector, HashSet, LinkedHashSet, TreeSet, PriorityQueue, ArrayDeque, etc.) deal with objects only.

|  |
| --- |
| int i=100;  Integer iObject = I;  Primitive value assigned to Wrapper class, which is autoboxing. |
| Integer object = new Integer(10); //this is wrapper object  int value = object.intValue(); //wrapper to primitive conversion |
| String number = new String(“100”);  int value = Integer.parseInt(number); //string to int primate conversion |
| int value =100;  String number = String.valueOf(value); //primate to String conversion |
| Assignment: convert number to binary number |
| Assignment: convert number to octal number |
| Assignment: convert number to hexadecimal |
| Assignment: print max and min numbers |
| Write methods available in Wrapper classes and their usage. |

Java Doc API

[java.lang (Java SE 11 & JDK 11 ) (oracle.com)](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/package-summary.html)

|  |
| --- |
| public class WrapperClass {    public static void main(String[] args) {   *//Primitive to Object conversion* int amount=200;  Integer integerObject = amount; *//valid syntax, direct, compiler will do conversion* Integer intObject = new Integer(200);*// using Constructor, not recommended instead use static method as below* Integer intObj2= Integer.*valueOf*(200);*//parameter take int value* Integer intObj3= Integer.*valueOf*("200");*//parameter take String value   //Object to primitive Conversion* int value = integerObject.intValue();*// object to int primitive conversion* String intStringValue = intObj2.toString(); *// get String int value* String intStringValue2 = String.*valueOf*(29);   String number ="200";  int i = Integer.*parseInt*(number);*//String to primitive int conversion* System.*out*.println(Integer.*MAX\_VALUE*);  System.*out*.println(Integer.*MIN\_VALUE*);   System.*out*.println(Integer.*toBinaryString*(10));  System.*out*.println(Integer.*toHexString*(10));  System.*out*.println(Integer.*toOctalString*(10));   } } |
| public class WrapperClass {    public static void main(String[] args) {  *//double numbers show infinity and NAN* System.*out*.println(0.0D / 0.0D);  System.*out*.println(0.0D / 10.0D);  System.*out*.println(10.0D / 0.0D);  System.*out*.println(-10.0D / 0.0D);     *//float numbers show infinity and NAN* System.*out*.println(0.0F / 0.0F);  System.*out*.println(0.0F / 10.0F);  System.*out*.println(10.0F / 0.0F);  System.*out*.println(-10.0F / 0.0F);  } } |
| public class WrapperClass {    public static void main(String[] args) {   *//byte,int,short,long numbers throws java.lang.ArithmeticException: / by zero* System.*out*.println((byte) 0 / (byte) 0);  System.*out*.println((byte) 0 / (byte) 10);  System.*out*.println((byte) 10 / (byte) 0);  System.*out*.println(-(byte) 10 / (byte) 0);  } } |
| By default numbers are consider at int level.  Sout(10); here 10 is considered as int number,  Sout((byte)10) to specify byte number  Sout ((short)20) to specify short number  Sout (100L); L and lower case l is for long value type  Sout (10.0F); F or f for float number  Sout(10.10D); D or d for double number.  By default sout(10.0) is considered as double number. |