**Wrapper Class**

* The **wrapper class in Java** provides the mechanism to convert primitive into object and object into primitive.
* The automatic conversion of primitive into an object is known as autoboxing .
* Objects into primitives automatically is called unboxing .
* The eight classes of the java.lang package are known as wrapper classes in java .
* The list of eight wrapper classes are given below :

|  |  |
| --- | --- |
| **Primitive types** | **Wrapper Classes** |
| boolean | Boolean |
| char | Character |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| float | float |
| double | Double |

**Needs of Wrapper classes**

* Generic classes only work with objects and don’t support primitives.
* 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 .

**Importance of Wrapper classes**

* There are mainly two uses with wrapper classes .

1. To convert simple data types into objects , i.e. to give object form to a data type ; here constructors are used .
2. To convert strings into data types(known as parsing operations) , here methods of type parseYYY() are used . YYY -> datatypes

**Features of java Wrapper classes**

* Wrapper classes convert numeric strings into numeric values .
* The way to store primitive data is an object .
* The valueOf() method is available in all wrapper classes except character .
* All wrapper classes have datatypeValue() method .This method returns the value of the object as its primitive type .

**Creating Wrapper Objects**

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”) ;

} }

**Methods of wrapper classes**

* valueOf() method : To create wrapper object for given primitive data types or String .
* parseInt() , parseFloat() , parseDouble() method : to convert String class (Wrapper class) to primitive .

|  |  |
| --- | --- |
| **Method** | **Purpose** |
| parseInt(s) | Return a signed decimal integer value equivalent to string s |
| toString(i) | Returns a new string object representing the integer i |
| byteValue() | Returns the value of this integer as a byte |
| doubleValue() | Returns the value of this integer as a double |
| floatValue() | Returns the value of this integer as a float |
| intValue() | Returns the value of this integer as a int |
| shortValue() | Returns the value of this integer as a short |
| longValue() | Returs the value of this integer as a long |

**Converts wrapper class into primitive data types**

public class MyClass {

public static void main(String[] args) {

Integer myInt = 5;

Double myDouble = 5.99;

Character myChar = 'A';

System.***out***.println(myInt.intValue());

System.***out***.println(myDouble.doubleValue());

System.***out***.println(myChar.charValue());

}

}

**autoboxing**

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);

}

}

**Unboxing**

public class UnboxingTest {

public static void main(String[] args) {

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

//creating a wrapper class object

int num = obj.intValue();

//converting the wrapper object to primitive datatypes

System.***out***.println(num +" "+obj);

}

}

**Primitive data types to Wrapper class**

//Autoboxing example of Int to integer

class WrapperExample {

public static void main(String[] args) {

//Converting into into integer

int a = 20 ;

Integer i = Integer.*valueOf*(a);

//converting int into Integer explicitly

Integer j = a ;

//autoboxing , now compiler will write integer.valueOf(a) internally

System.***out***.println(a+" " +i+" " +j);

}

}

**toString()**

//Java program to illustrate toString()

class GFG {

public static void main(String[] args) {

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

String s = I.toString();

System.***out***.println("s");

}

}