**Wrapper classes in Java**

The wrapper class in Java provides the mechanism to convert primitive into object and object into primitive.

Since J2SE 5.0, autoboxing and unboxing feature convert primitives into objects and objects into primitives automatically. The automatic conversion of primitive into an object is known as autoboxing and vice-versa unboxing.

Use of Wrapper classes in Java

Java is an object-oriented programming language, so we need to deal with objects many times like in Collections, Serialization, Synchronization, etc. Let us see the different scenarios, where we need to use the wrapper classes.

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.
* 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 Type | Wrapper class |
| boolean | [Boolean](https://www.javatpoint.com/java-boolean) |
| char | [Character](https://www.javatpoint.com/post/java-character) |
| byte | [Byte](https://www.javatpoint.com/java-byte) |
| short | [Short](https://www.javatpoint.com/java-short) |
| int | [Integer](https://www.javatpoint.com/java-integer) |
| long | [Long](https://www.javatpoint.com/java-long) |
| float | [Float](https://www.javatpoint.com/java-float) |
| double | [Double](https://www.javatpoint.com/java-double) |

**Autoboxing**

The automatic conversion of primitive data type into its corresponding wrapper class is known as autoboxing, for example, byte to Byte, char to Character, int to Integer, long to Long, float to Float, boolean to Boolean, double to Double, and short to Short.

Since Java 5, we do not need to use the valueOf() method of wrapper classes to convert the primitive into objects.

**Wrapper class Example: Primitive to Wrapper**

//Java program to convert primitive into objects

//Autoboxing example of int to Integer

public class WrapperExample1{

public static void main(String args[]){

//Converting int 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);

}}

**Output:**

20 20 20

**Unboxing**

The automatic conversion of wrapper type into its corresponding primitive type is known as unboxing. It is the reverse process of autoboxing. Since Java 5, we do not need to use the intValue() method of wrapper classes to convert the wrapper type into primitives.

**Wrapper class Example: Wrapper to Primitive**

public class WrapperExample2{

public static void main(String args[]){

//Converting Integer to int

Integer a=new ~~Integer~~(3);

int i=a.intValue();//converting Integer to int explicitly

int j=a;//unboxing, now compiler will write a.intValue() internally

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

}

}

**Output:**

3 3 3

**Java Wrapper classes Example**

//Java Program to convert all primitives into its corresponding

//wrapper objects and vice-versa

public class WrapperExample3{

public static void main(String args[]){

byte b=10;

short s=20;

int i=30;

long l=40;

float f=50.0F;

double d=60.0D;

char c='a';

boolean b2=true;

//Autoboxing: Converting primitives into objects

Byte byteobj=b;

Short shortobj=s;

Integer intobj=i;

Long longobj=l;

Float floatobj=f;

Double doubleobj=d;

Character charobj=c;

Boolean boolobj=b2;

//Printing objects

System.***out***.println("---Printing object values---");

System.***out***.println("Byte object: "+byteobj);

System.***out***.println("Short object: "+shortobj);

System.***out***.println("Integer object: "+intobj);

System.***out***.println("Long object: "+longobj);

System.***out***.println("Float object: "+floatobj);

System.***out***.println("Double object: "+doubleobj);

System.***out***.println("Character object: "+charobj);

System.***out***.println("Boolean object: "+boolobj);

//Unboxing: Converting Objects to Primitives

byte bytevalue=byteobj;

short shortvalue=shortobj;

int intvalue=intobj;

long longvalue=longobj;

float floatvalue=floatobj;

double doublevalue=doubleobj;

char charvalue=charobj;

boolean boolvalue=boolobj;

//Printing primitives

System.***out***.println("---Printing primitive values---");

System.***out***.println("byte value: "+bytevalue);

System.***out***.println("short value: "+shortvalue);

System.***out***.println("int value: "+intvalue);

System.***out***.println("long value: "+longvalue);

System.***out***.println("float value: "+floatvalue);

System.***out***.println("double value: "+doublevalue);

System.***out***.println("char value: "+charvalue);

System.***out***.println("boolean value: "+boolvalue);

}}

**Output:**

---Printing object values---

Byte object: 10

Short object: 20

Integer object: 30

Long object: 40

Float object: 50.0

Double object: 60.0

Character object: a

Boolean object: true

---Printing primitive values---

byte value: 10

short value: 20

int value: 30

long value: 40

float value: 50.0

double value: 60.0

char value: a

boolean value: true

**Custom Wrapper class in Java**

Java Wrapper classes wrap the primitive data types, that is why it is known as wrapper classes. We can also create a class which wraps a primitive data type. So, we can create a custom wrapper class in Java.

//Creating the custom wrapper class

class Javatpoint{

private int i;

Javatpoint(){}

Javatpoint(int i){

this.i=i;

}

public int getValue(){

return i;

}

public void setValue(int i){

this.i=i;

}

*@Override*

public String toString() {

return Integer.*toString*(i);

}

}

//Testing the custom wrapper class

public class TestJavatpoint{

public static void main(String[] args){

Javatpoint j=new Javatpoint(10);

System.***out***.println(j);

}}

**Output:**

10

**Example:**

public class Demo {

public static void main(String[] args) {

// Autoboxing: Converting primitive types to their corresponding wrapper classes

int num = 42;

Integer wrappedNum = num; // Autoboxing

System.***out***.println("Autoboxing example:");

System.***out***.println("Primitive int: " + num);

System.***out***.println("Wrapped Integer: " + wrappedNum);

// Unboxing: Converting wrapper classes to their corresponding primitive types

Double doubleValue = 3.14;

double unwrappedValue = doubleValue; // Unboxing

System.***out***.println("\nUnboxing example:");

System.***out***.println("Wrapped Double: " + doubleValue);

System.***out***.println("Unwrapped double: " + unwrappedValue);

// Autoboxing and unboxing with method parameters

*calculateSum*(10, 20); // primitive int values are autoboxed to Integer objects

*calculateProduct*(2.5, 3); // primitive double and int values are autoboxed to Double and Integer objects

}

public static void calculateSum(Integer a, Integer b) {

int sum = a + b; // Integer objects are unboxed to primitive int for calculation

System.***out***.println("\nSum: " + sum);

}

public static void calculateProduct(Double a, Integer b) {

double product = a \* b; // Double and Integer objects are unboxed to primitive double and int for calculation

System.***out***.println("Product: " + product);

}

}

Output:

Autoboxing example:

Primitive int: 42

Wrapped Integer: 42

Unboxing example:

Wrapped Double: 3.14

Unwrapped double: 3.14

Sum: 30

Product: 7.5

**Problem Statement : Character Identifier**

Question Description: Create a program that identifies a character as a lower-case vowel, upper-case vowel, lower-case consonant, upper-case consonant, digit, or special character. The program should accept a character 'ch' as input.

**Boiler Plate Code:**

public class Main

{

public static void identifyCharacter(char ch)

{

// your code here

}

public static void main(String[] args)

{

}

}

Sample Input: 'a'

Sample Output: Lower-case vowel

**Solution:**

import java.util.Scanner;

public class Main {

public static void identifyCharacter(char ch) {

if (Character.*isLowerCase*(ch)) {

if (*isVowel*(ch)) {

System.***out***.println("Lower-case vowel");

} else {

System.***out***.println("Lower-case consonant");

}

} else if (Character.*isUpperCase*(ch)) {

if (*isVowel*(ch)) {

System.***out***.println("Upper-case vowel");

} else {

System.***out***.println("Upper-case consonant");

}

} else if (Character.*isDigit*(ch)) {

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

} else {

System.***out***.println("Special character");

}

}

public static boolean isVowel(char ch) {

ch = Character.*toLowerCase*(ch);

return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.***in***);

System.***out***.print("Enter a character: ");

char ch = scanner.next().charAt(0);

*identifyCharacter*(ch);

scanner.close(); }

}

**Output:**

Enter a character: P

Upper-case consonant

Enter a character: a

Lower-case vowel

Enter a character: O

Upper-case vowel

Enter a character: 3

Digit

Enter a character: =

Special character