

[Write an Article](#)[Login](#)

## Type conversion in Java with Examples

When you assign value of one data type to another, the two types might not be compatible with each other. If the data types are compatible, then Java will perform the conversion automatically known as Automatic Type Conversion and if not then they need to be casted or converted explicitly. For example, assigning an int value to a long variable.

### Widening or Automatic Type Conversion

Widening conversion takes place when two data types are automatically converted. This happens when:

- The two data types are compatible.
- When we assign value of a smaller data type to a bigger data type.

For Example, in java the numeric data types are compatible with each other but no automatic conversion is supported from numeric type to char or boolean. Also, char and boolean are not compatible with each other.

**Byte → Short → Int → Long → Float → Double**

### Widening or Automatic Conversion

Example:

```
class Test
{
    public static void main(String[] args)
    {
        int i = 100;

        //automatic type conversion
        long l = i;

        //automatic type conversion
        float f = l;
        System.out.println("Int value "+i);
        System.out.println("Long value "+l);
        System.out.println("Float value "+f);
    }
}
```

Output:



```
Int value 100
Long value 100
Float value 100.0
```

### Narrowing or Explicit Conversion

If we want to assign a value of larger data type to a smaller data type we perform explicit type casting or narrowing.

- This is useful for incompatible data types where automatic conversion cannot be done.
- Here, target-type specifies the desired type to convert the specified value to.

**Double → Float → Long → Int → Short → Byte**

### Narrowing or Explicit Conversion

char and number are not compatible with each other. Let's see when we try to convert one into other.

```
//Java program to illustrate incompatible data
// type for explicit type conversion
public class Test
{
    public static void main(String[] argv)
    {
        char ch = 'c';
        int num = 88;
        ch = num;
    }
}
```

Run on IDE

Error:

```
7: error: incompatible types: possible lossy conversion from int to char
    ch = num;
      ^
1 error
```

### How to do Explicit Conversion?

Example:

```
//Java program to illustrate explicit type conversion
class Test
{
    public static void main(String[] args)
```

```
{
    double d = 100.04;

    //explicit type casting
    long l = (long)d;

    //explicit type casting
    int i = (int)l;
    System.out.println("Double value "+d);

    //fractional part lost
    System.out.println("Long value "+l);

    //fractional part lost
    System.out.println("Int value "+i);
}
```

[Run on IDE](#)

Output:

```
Double value 100.04
Long value 100
Int value 100
```

While assigning value to byte type the fractional part is lost and is reduced to modulo 256(range of byte).

Example:

```
//Java program to illustrate Conversion of int and double to byte
class Test
{
    public static void main(String args[])
    {
        byte b;
        int i = 257;
        double d = 323.142;
        System.out.println("Conversion of int to byte.");

        //i%256
        b = (byte) i;
        System.out.println("i = " + i + " b = " + b);
        System.out.println("\nConversion of double to byte.");

        //d%256
        b = (byte) d;
        System.out.println("d = " + d + " b= " + b);
    }
}
```

[Run on IDE](#)

Output:

```
Conversion of int to byte.
i = 257 b = 1

Conversion of double to byte.
d = 323.142 b = 67
```

## Type promotion in Expressions

While evaluating expressions, the intermediate value may exceed the range of operands and hence the expression value will be promoted. Some conditions for type promotion are:

1. Java automatically promotes each byte, short, or char operand to int when evaluating an expression.
2. If one operand is a long, float or double the whole expression is promoted to long, float or double respectively.

Example:

```
//Java program to illustrate Type promotion in Expressions
class Test
{
    public static void main(String args[])
    {
        byte b = 42;
        char c = 'a';
        short s = 1024;
        int i = 50000;
        float f = 5.67f;
        double d = .1234;

        // The Expression
        double result = (f * b) + (i / c) - (d * s);

        //Result after all the promotions are done
        System.out.println("result = " + result);
    }
}
```

[Run on IDE](#)

Output:

```
Result = 626.7784146484375
```

### Explicit type casting in Expressions

While evaluating expressions, the result is automatically updated to larger data type of the operand. But if we store that result in any smaller data type it generates compile time error, due to which we need to type cast the result.

Example:

```
//Java program to illustrate type casting int to byte
class Test
{
    public static void main(String args[])
    {
        byte b = 50;

        //type casting int to byte
        b = (byte)(b * 2);
        System.out.println(b);
    }
}
```

[Run on IDE](#)

Output

100

NOTE- In case of single operands the result gets converted to int and then it is type casted accordingly.  
Example:

```
//Java program to illustrate type casting int to byte
class Test
{
    public static void main(String args[])
    {
        byte b = 50;

        //type casting int to byte
        b = (byte)(b * 2);
        System.out.println(b);
    }
}
```

[Run on IDE](#)

### Output

100

This article is contributed by **Apoorva singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using [contribute.geeksforgeeks.org](https://contribute.geeksforgeeks.org) or mail your article to [contribute@geeksforgeeks.org](mailto:contribute@geeksforgeeks.org). See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Java

[Login to Improve this Article](#)

Please write to us at [contribute@geeksforgeeks.org](mailto:contribute@geeksforgeeks.org) to report any issue with the above content.

## Recommended Posts:

[Comments in Java](#)[Does Java support goto?](#)[Interesting facts about null in Java](#)[Bitwise right shift operators in Java](#)[Widening Primitive Conversion in Java](#)[Java | Converting an Image into Grayscale using cvtColor\(\)](#)[Java | Removing whitespaces using Regex](#)[Java System.nanoTime\(\) vs System.currentTimeMillis](#)[Java toDegrees\(\) method with Example](#)

## Java toRadians() method with Example

(Login to Rate)

1.7

Average Difficulty : **1.7/5.0**  
Based on **31** vote(s)

☐

Add to TODO List

☐

Mark as DONE

Basic

Easy

Medium

Hard

Expert

Writing code in comment? Please use [ide.geeksforgeeks.org](https://ide.geeksforgeeks.org), generate link and share the link here.


Share this post!

9 Comments

GeeksforGeeks

 Login ▾

 Recommend 1

 Share

Sort by Newest ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

**manish pamnani** • 20 days ago

Does System.out.println("Int value " + i); prints on next line or the same line?

^ | ▾ • Reply • Share ›

**manish pamnani** • 20 days ago

What is the range of long here?

^ | ▾ • Reply • Share ›

**manish pamnani** • 20 days ago

Example 1: line 8:

If it is an automatic type conversion then why do we have to write "long l = i" ?

^ | ▾ • Reply • Share ›

**manish pamnani** • 20 days ago

Can static variable be converted to non-static variable?

^ | ▾ • Reply • Share ›

**manish pamnani** • 20 days ago

What is to convert explicitly ?

^ | ▾ • Reply • Share ›

**Himanshu Prakash** • 4 months ago

I will like to highlight one important point regarding Assignment operators. Assignment Operators like +=, -=, \*=, /= are smart assignment operators.

For example Generally It seems that

variable \*= constant will have same effect as variable = variable \* constant.

But actually, this is not the case when always same effect take place. consider below code for example. here un-commenting line 2 will give a compile time error where as Line 1 gives a successful assignment.

```
public class operators
{
    public static void main(String[] args)
    {
        byte b = 50;

        //Line 1 explicit type casting not required
        b *= 2;

        //Line 2 explicit type casting required so uncommenting it will produce error
        // b = b*2;

        //line 3
        b=(byte) (b*2);

        System.out.println(b);
    }
}
```

^ | v • Reply • Share ›



**Nihal** • 8 months ago

```
public class Test
{
    public static void main(String[] argv)
    {
        char ch = 'c';
        int num = 88;
        num=ch;
    }
}

will work
```

1 ^ | v • Reply • Share ›

**Pallavi Manan** • a year ago

why is the conversion from long to float compatible? long size is 8 bytes and float size 4 bytes

1 ^ | v • Reply • Share ›

**Bharat Chauhan** ➔ Pallavi Manan • 7 months ago

That is what is explicit type conversion. Here you convert data types having more size to data types having less size

^ | v • Reply • Share ›

## A computer science portal for geeks

710-B, Advant Navis Business Park,  
Sector-142, Noida, Uttar Pradesh - 201305  
[feedback@geeksforgeeks.org](mailto:feedback@geeksforgeeks.org)

### COMPANY

[About Us](#)  
[Careers](#)  
[Privacy Policy](#)  
[Contact Us](#)

### PRACTICE

[Company-wise](#)  
[Topic-wise](#)  
[Contests](#)  
[Subjective Questions](#)

### LEARN

[Algorithms](#)  
[Data Structures](#)  
[Languages](#)  
[CS Subjects](#)  
[Video Tutorials](#)

### CONTRIBUTE

[Write an Article](#)  
[GBlog](#)  
[Videos](#)

@geeksforgeeks, Some rights reserved