**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.  
A picture containing text

Description automatically generated  
Example:

class Test  
{  
public static void main(String[] args)  
{  
int i = 200;

// 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 200  
Long value 200  
Float value 200.0

**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.  
  

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 = 98;  
ch = num;  
}  
}  
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 = 200.04;

//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);  
}  
}  
**Output:**

Double value 200.04  
Long value 200  
Int value 200  
While assigning value to byte type the fractional part is lost and is reduced to modulo 256(range of byte).