

(Transcript) Part 1-Data Type Conversions/Casting

Rex
Jones II

Part 1

Data Type Conversion & Casting

Convert Compatible Data Types

In this session, let's talk about Data Type Conversions and Data Type Casting. Both involve changing the Data Type from one data type to a different data type. However, the difference depends on their compatibility. A Data Type Conversion happens automatically with compatible data types. If the data types are then we must apply a Data Type Casting. We are not changing the value but only converting the value.

In Eclipse, let's start with a Data Type Conversion which can also be called // Widening Conversion. Widening Conversion is when a smaller data type is converted to a larger data type. It gets wide from small to large. For numeric data types, the precision goes from byte, short, int, long, float, double. Byte is the smallest numeric data type and double is the largest data type. If our data types are from left to right then the compiler automatically convert the types.

2 things must happen for an automatic conversion. Both data types must be compatible and the destination type must be larger than the source type. Look what happens when the destination is smaller. `int destination = .int` is a whole number and smaller than the source `34.56`. `34.56` is a double. Therefore, the error message states "Type mismatch: cannot convert from double to int". Let's add a comment `// Cannot Convert From double to int`

Now, look what happens for the automatic conversion. We write
`int source1 = 34 / double destination1 = source1.` `source1` is smaller than `destination1` because `int` has a lower precision than `double`. Let's print the values:

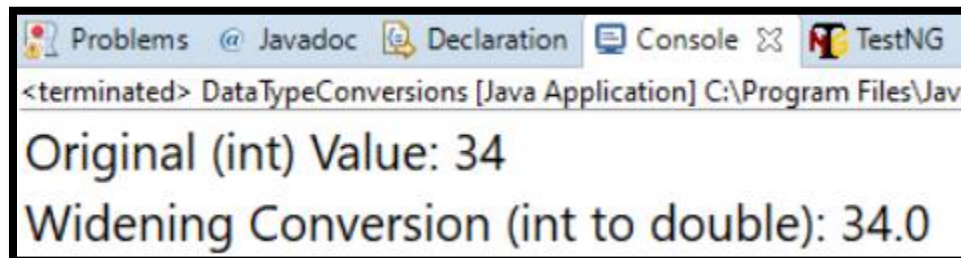
`sysout("Original (int) Value: " + source1)`

`sysout("Widening Conversion (int to double): " + destination1)` Let's Run

```
public static void main(String[] args) {
    // Widening Conversion (byte, short, int, long, float, double)

    // int destination = 34.56; Cannot Convert From double to int
    int source1 = 34;
    double destination1 = source1;
    System.out.println("Original (int) Value: " + source1);
    System.out.println("Widening Conversion (int to double): " + destination1);
}
```

Original (int) Value is 34 and Widening Conversion (int to double) is 34.0. The value on the right which is `source1` automatically converted to the `double` data type `destination1` on the left. So, we see 34.0



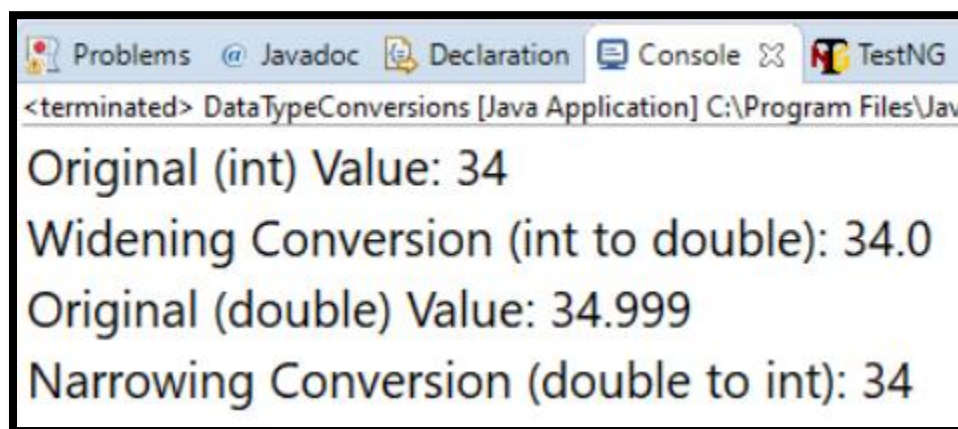
```
<terminated> DataTypeConversions [Java Application] C:\Program Files\Jav
Original (int) Value: 34
Widening Conversion (int to double): 34.0
```

Next, is Data Type Casting which can also be called // Narrowing Conversion. This type of conversion happens when the destination type is smaller than the source type. We write `double source2 = 34.999 / int destination2 = 34.999.` This is the same error from last time: "Type mismatch: cannot convert from `double` to `int`". We have to explicitly carry out the conversion by casting the data type. Casting takes place when we send an instruction to the compiler to convert the data type. The `cast ()` always happen before the value. Therefore, we place `int` for `int` inside the parenthesis.

A person might think the value 34.999 will round up to 35 but that's not true. It will convert to an `int` value. However, the decimal and 999 will cast off and not be shown. Let me change 34.999 to `source2` and add some print statements. `sysout("-----")`
`sysout("Original (double) Value: " + source2) /`
`sysout("Narrowing Conversion (double to int): " + destination2)`

```
// Narrowing Conversion
double source2 = 34.999;
int destination2 = (int) source2;
System.out.println("Original (double) Value: " + source2);
System.out.println("Narrowing Conversion (double to int): " + destination2);
```

Let's Run. Original (double) Value is 34.999. The Narrowing Conversion (double to int) cast off .999 and only show 34.



```
<terminated> DataTypeConversions [Java Application] C:\Program Files\Java
Original (int) Value: 34
Widening Conversion (int to double): 34.0
Original (double) Value: 34.999
Narrowing Conversion (double to int): 34
```

In the next video, I am going to convert an object to a variable and convert a String class to a numeric value. You can connect with me on LinkedIn, follow me on Twitter, and subscribe to my YouTube channel. See you next time.

Social Media Contact

- YouTube <https://www.youtube.com/c/RexJonesII/videos>
- Twitter <https://twitter.com/RexJonesII>
- LinkedIn <https://www.linkedin.com/in/rexjones34/>
- GitHub <https://github.com/RexJonesII/Free-Videos>
- Facebook <http://facebook.com/JonesRexII>