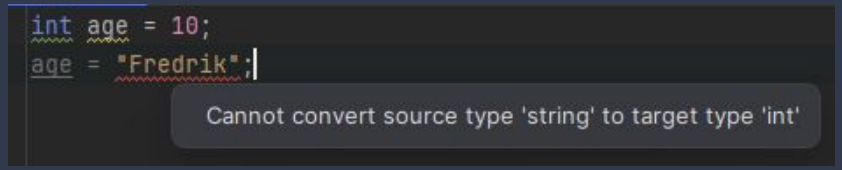
Converting from one to another

Type safety



Earlier we learned that C# is a type safe language meaning that variables have a certain type which cannot change thus making it type safe.

- **int age = 10;**
- **age = “Fredrik”; // ERROR**

Though this would cause an error, there are other conversions that are allowed, through **Implicit/Explicit Casting**, or **Conversion** and **Parsing**.

Implicit Casting

```
int number = 5 ;
float fractionalNumber = number; // value: 5.0f
```

Some types can be **Implicitly** casted into another type because they are closely related to each other.

For example an integer can be a fractional number:

- **int number = 5;**
- **float fractionalNumber = number; // value: 5.0f**

This work because conversion can happen without losing any information in the process.

Explicit Casting

```
float fractionalNumber = 9.78f;
int number = (int) fractionalNumber; // new value: 9
```

Some can only be casted **Explicitly**, due to the act coming with a loss of precision:

- **float decimalNumber = 9.78;**
- **int number = (int) decimalNumber; // value: 9**

The reason for this is that for example **converting** a fractional number into a whole number comes with the loss of precision, 9.78 suddenly becomes 9 and you've just lost 0.78 of the value.

Sometimes that's okay, when you only care about a whole number!

Conversion

```
1 {}  string input = "Fredrik";
2     int i = Convert.ToInt32(input);
```

```
Unhandled exception. System.FormatException:
The input string 'Fredrik' was not in a corre
ct format.
```

Other times we need to explicitly **convert** things, where conversions can be safely validated and throw an exception if they are invalid.

For example if the is string “Cat”, it’s not a number.

Some conversions:

- **string input = “2”;**
- **int i = Convert.ToInt32(input); // new value: 2**

- **string input = “2.3”;**
- **double d = Convert.ToDouble(input); // value 2.3**

- **string input = “Fredrik”;**
- **input i = Convert.ToInt32(input); // EXCEPTION (Error), cannot convert letters into numbers**

Later we will learn how to safely handle exceptions.

Parsing

```
// Instead of this..
string input = "22";
int number = Convert.ToInt32(input);

// I can use this?
int number2 = int.Parse(input);
```

```
// But remember:
string input = "2,2";
int number = int.Parse(input, CultureInfo.InvariantCulture);
// Culture info matters when parsing decimals.
```

Parsing describes a special conversion in which the input is a string.

Instead of writing **Convert.ToInt32();** you can write:

- **int number = int.Parse(input);**

Be weary though as parsing numbers is dependent on your systems regional settings.

In Sweden and Germany for example, decimal numbers are write **2,3** instead of **2.3** and if you try converting it in the wrong format you will see the error: “Input string was not in the correct format”.

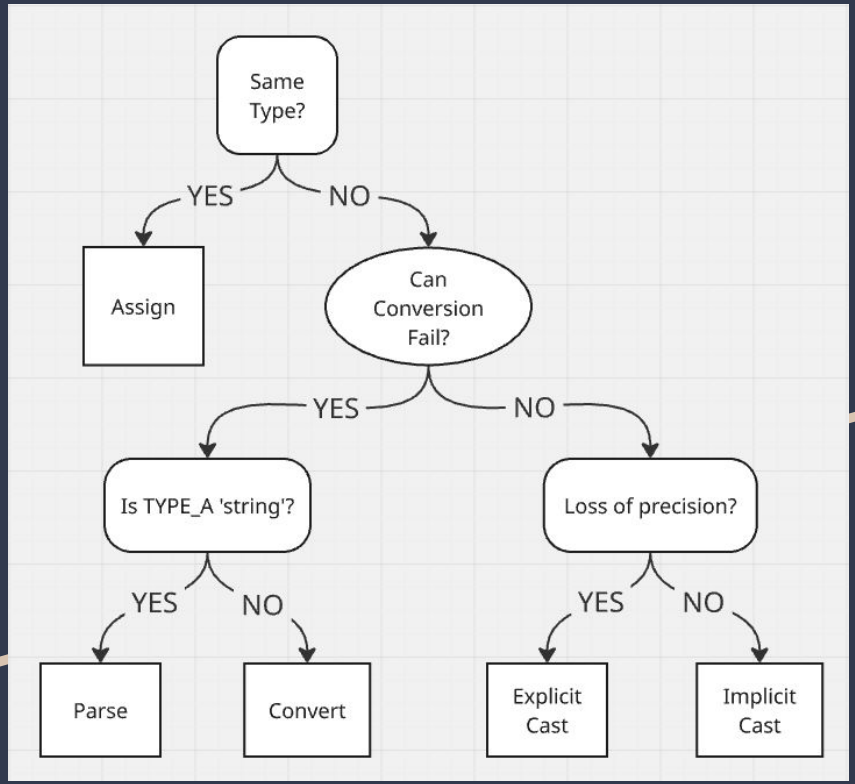
To always convert numbers using the internal standard notation use this instead:

- **int number = int.Parse(input, CultureInfo.InvariantCulture);**

Or you configure it at the start of your program for all upcoming parsing:

- **Thread.CurrentThread.CurrentCulture = CultureInfo.InvariantCulture;**

Summary



Try for yourself:

- **TYPE_A a;**
- **TYPE_B b = a;**

Follow the flow chart or you can brute force it:

- Try **assigning** it, if it gives an error, then…
- Try **casting** it explicitly, if it gives an error, then…
- Try **parsing** it, if it still gives and error, then…
- Try **converting** it.

Conversions are often necessary while computing, for example:

- When you ask the user how many times he wants to purchase something…
  - But **Console.ReadLine();** always returns a string
  - You can use **int.Parse(userInput);** to convert the text into a number
- There's a variety of ways of doing that, you should always choose the easiest path that's available for your conversion!

Goal

```
Output: Give me a number.
Input: 5,6
Output: 5,6
Output: 5
```

- Create a console project named **E8Conversion**
- Ask the user for a Number and assign the result to a variable.
- Convert the variable to type **float**
- Print the **float** to the console
- Convert the double number to type **int**
- Print the **int** to the console
- Convert the original **string** to type **int**
- What happens? (hint: you will see an exception)