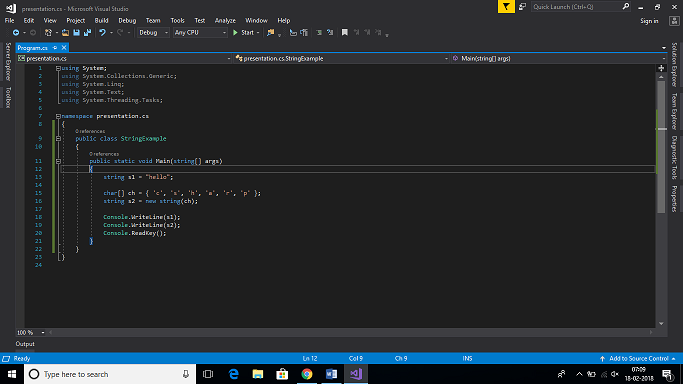
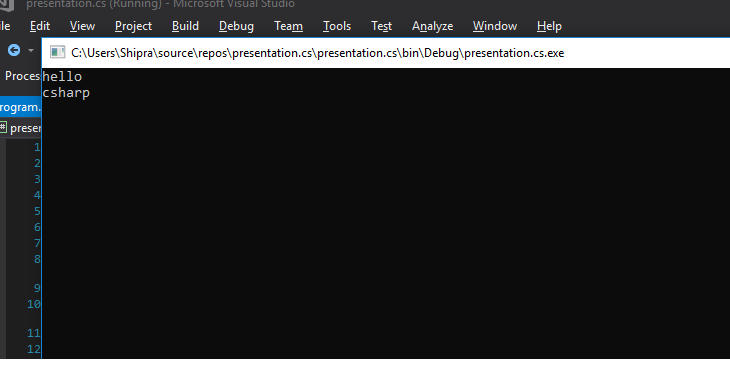
# **Strings, Properties & Type Conversion**

**STRINGS:**

Definition : .

In computer science a **string** is any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks). An important characteristic of each **string** is its length, which is the number of characters in it. The length can be any natural number (i.e., zero or any positive integer).

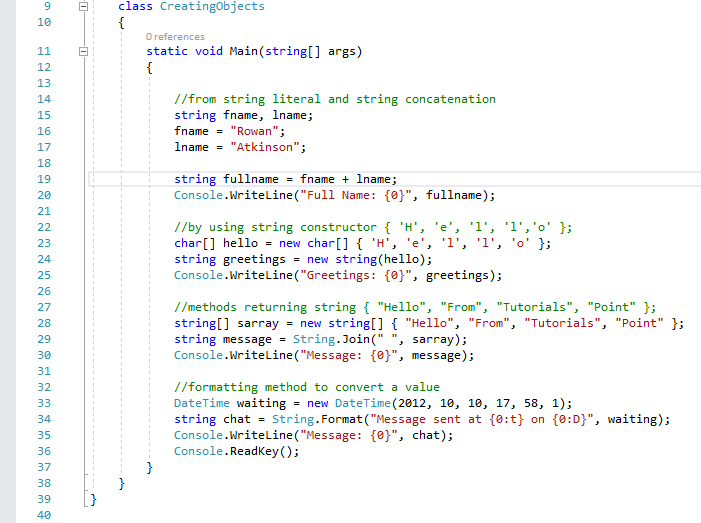


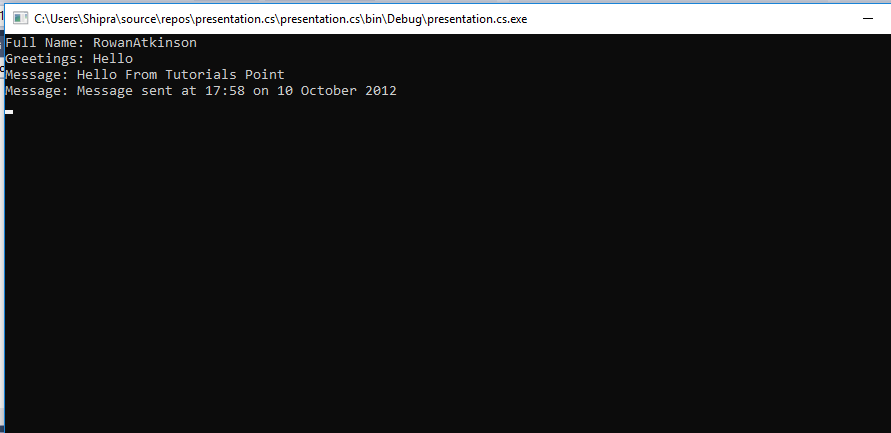


Real World Example:

You type a message on WhatsApp or other messaging app on your Android phone. Input text is a Java **String** and is encrypted. Received on other person’s phone messenger, decrypted and displayed as **String** which you see as text.

There are several methods for creating string objects:





Properties: .

**CHAR** Gets the *Char* object at a specified position in the current *String* object.

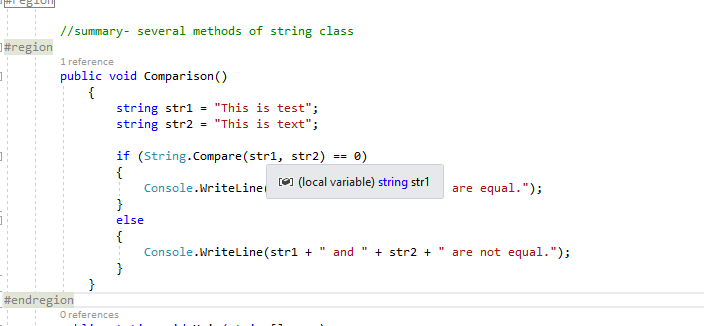
**LENGTH** Gets the number of characters in the current String object.

Methods:

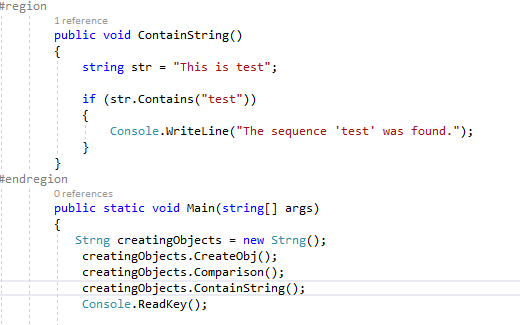
We can perform various string operations on strings like comparison, concatenation, getting substring, search, trim, replacement, etc.

Given below is the list of methods of the String class.

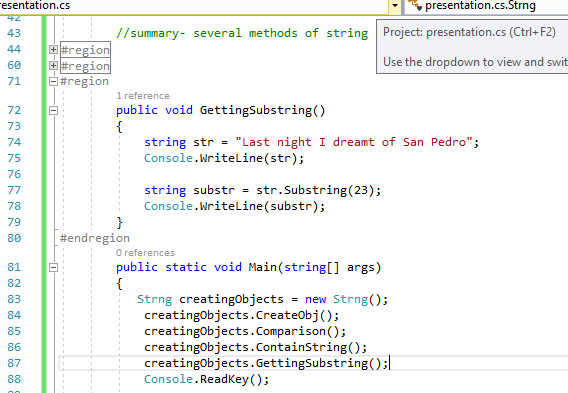
1.Comparing String



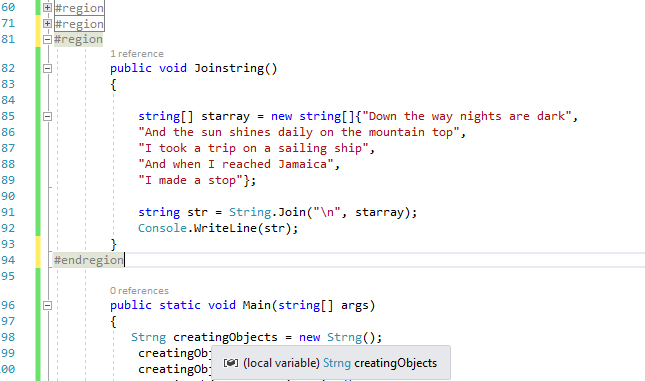
2. String contains string



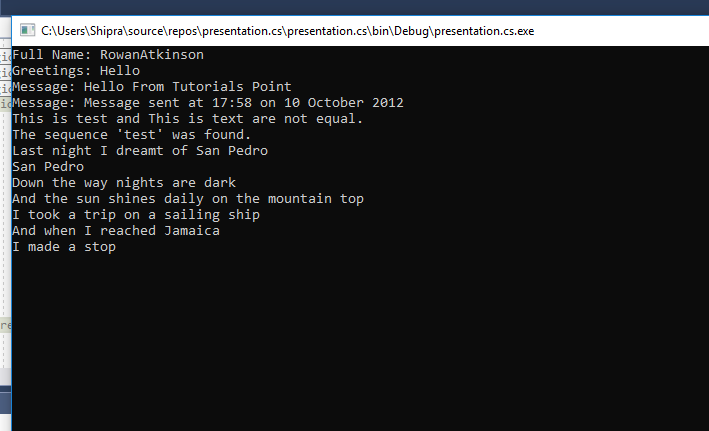
3.Getting a Substring



4.Joining Strings



[**..\source\repos\presentation.cs\presentation.cs\presentation.cs.csproj**](file:///C:\Users\Shipra\source\repos\presentation.cs\presentation.cs\presentation.cs.csproj)

****

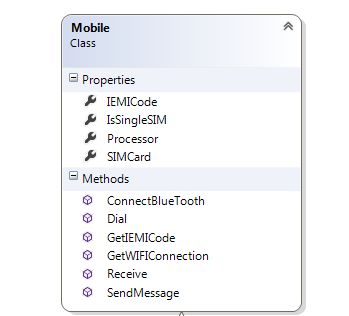
**PROPERTIES:**

C# Properties doesn't have storage location. C# Properties are extension of fields and accessed like fields. The Properties have accessors that are used to set, get or compute their values.

## Usage of Properties:

1. C# Properties can be read-only or write-only.
2. We can have logic while setting values in the C# Properties.
3. We make fields of the class private, so that fields can't be accessed from outside the class directly. Now we are forced to use C# properties for setting or getting values.

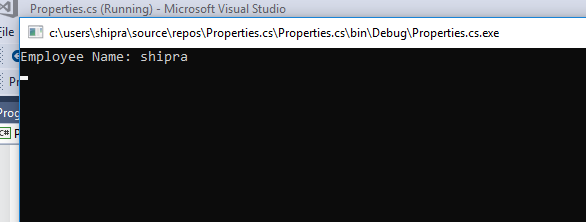
## Real World Example:



## Code Snippets:

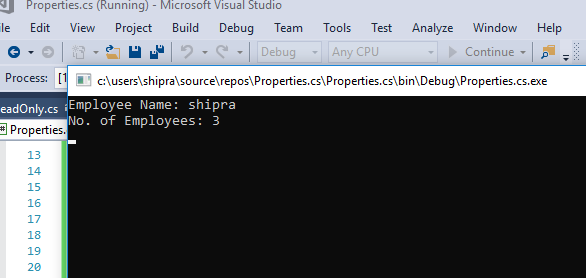
1.Properties Example

..\source\repos\Properties.cs\Properties.cs\Properties.cs.csproj



2.Read only property

[..\source\repos\Properties.cs\Properties.cs\ReadOnly.cs](../source/repos/Properties.cs/Properties.cs/ReadOnly.cs)

****

**TYPE CONVERSION:**

Definition : .

Type conversion is converting one type of data to another type. It is also known as Type Casting. In C#, type casting has two forms −

* **Implicit type conversion** − These conversions are performed by C# in a type-safe manner. For example, are conversions from smaller to larger integral types and conversions from derived classes to base classes.
* **Explicit type conversion** − These conversions are done explicitly by users using the pre-defined functions. Explicit conversions require a cast operator.

Real World Example:

You have given two vessels both are of different size, one is of small in size and other is large in size. Assume small vessel is filled with water and the large vessel can completely accommodate the small vessel’s water in it. But if large vessel is completely filled with water and small one is empty then small vessel cannot accommodate the whole water because of its small size and some of the water will get loss.

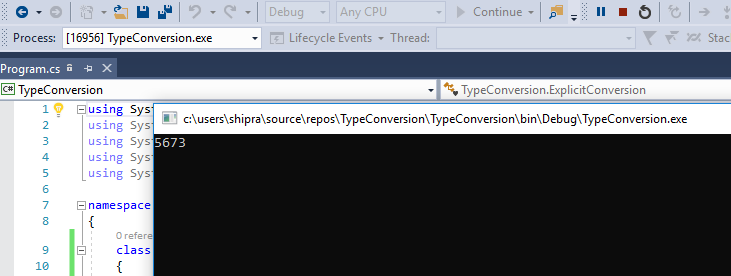
## C# Type Conversion Methods

C# provides the following built-in type conversion methods −

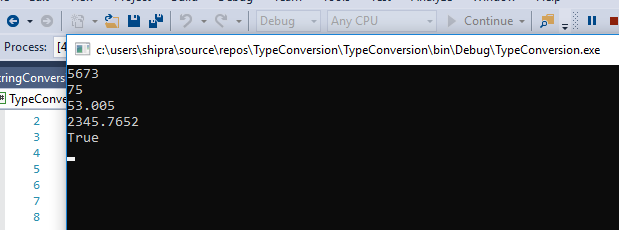
|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | **ToBoolean**  Converts a type to a Boolean value, where possible. |
| 2 | **ToByte**  Converts a type to a byte. |
| 3 | **ToChar**  Converts a type to a single Unicode character, where possible. |
| 4 | **ToDateTime**  Converts a type (integer or string type) to date-time structures. |
| 5 | **ToDecimal**  Converts a floating point or integer type to a decimal type. |
| 6 | **ToDouble**  Converts a type to a double type. |
| 7 | **ToInt16**  Converts a type to a 16-bit integer. |
| 8 | **ToInt32**  Converts a type to a 32-bit integer. |
| 9 | **ToInt64**  Converts a type to a 64-bit integer. |
| 10 | **ToSbyte**  Converts a type to a signed byte type. |
| 11 | **ToSingle**  Converts a type to a small floating point number. |
| 12 | **ToString**  Converts a type to a string. |
| 13 | **ToType**  Converts a type to a specified type. |
| 14 | **ToUInt16**  Converts a type to an unsigned int type. |
| 15 | **ToUInt32**  Converts a type to an unsigned long type. |
| 16 | **ToUInt64**  Converts a type to an unsigned big integer. |

Code Snippets:

[..\source\repos\TypeConversion\TypeConversion\Program.cs](../source/repos/TypeConversion/TypeConversion/Program.cs)



[..\source\repos\TypeConversion\TypeConversion\StringConversion.cs](../source/repos/TypeConversion/TypeConversion/StringConversion.cs)

****