1. Type conversions in C#

One of the main rules related with the usage of variables in C# is that once you declare the type of a variable, you cannot change it. Following the fact that C# is strongly typed programming language it cannot be assigned a value of another type as well. However, sometimes you may need to convert the "value" of a variable as part of assigning that value to a variable of a different type. One typical scenario can be assigning string value (entered as input from console) in variable declared as integer.

Therefore, in order to achieve this, you can use different techniques known as type conversions in C#.

- **<u>Type.Parse(value)</u>**
  - Converts a value and throws an exception if it can't be parsed (null is included).

```
int number = int.Parse(Console.ReadLine());
```

- **<u>Convert.ToType(value)</u>**
  - Same effect as Parse but converts null in to 0 instead of throwing exception.

```
string number = "10.50";
double convertedNumber = Convert.ToDouble(number);
```

- **<u>Type.TryParse(value, out variable)</u>**
  - Same as Parse but when a value is converted it stores it in a variable and returns true, otherwise it returns false instead of throwing an exception.

```
string consoleInput = Console.ReadLine();

int ifParsedValue;
bool convertingStatus = int.TryParse(consoleInput, out ifParsedValue);
```

Documentation link:
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/casting-and-type-conversions

2. Naming (Capitalization) conventions

- **Pascal Case**
  - Capitalizes the first character of each word.

```
string CourseName = "C# Basic";
```

- **Camel Case**
  - Capitalizes the first character of each word except the first word.

```
string courseName = "C# Basic";
```

- **Hungarian**
  - Names start with their abbreviated types - not recommended in C#.

```
string strCourseName = "C# Basic";
```