Exercises to Extension Methods

Program the following extension methods in a separate class **`Extensionmethods.cs`**. In a WPF (or console) app these methods should be used. Some of them are quite simple, others are more difficult.

- Write an extension method for string that returns the first and last two characters of the string.
- Write an extension method **`Yesterday`** for DateTime that returns the date exactly 24 hours before the specified one.
- Implement the function to calculate the factorial of a number as an extension Method **`Fact()`** for the type **`int`**. Remember the calculation of factorial: n!=n*(n-1)*…*1, e.g. 6! = 6*5*4*3*2*1
- Write an Extension method **`UseOnly`** for List<string>, which takes a function of type **`delegate bool CheckFunction(string)`** as a parameter and returns a List<string>. The returned list should contain only those elements for which the function returns true.
- Program the same thing for List <double>, i.e. with a **`delegate bool CheckFunction (double)`**
- Try the same for generic lists
- Write an Extension method **`Transform`** for List<Person>, which converts such a list into a list of strings by accepting a suitable delegate as parameter.

```
var persons = new List<Person>
{
  new Person {Firstname = "Hansi", Lastname = "Huber", Age = 66},
  new Person {Firstname = "Heinzi", Lastname = "Prüller", Age = 77},
  new Person {Firstname = "Susi", Lastname = "Maurer", Age = 55}
};
var fullnames = persons.Transform(x => $"{x.Firstname} {x.Lastname} [{x.Age}]");
```

  - therefore, the output of fullnames should be:

```
Hansi Huber [66]
Heinzi Prüller [77]
Susi Maurer [55]
```

- Write a **`ToUsefulString`** extension method for DateTime that converts the date to relative times:
  - Vorgestern 12:30
  - Gestern 17:21
  - Heute 08:45
  - vor 1 h 12 min (if less than 4 hours ago)
  - Jetzt
  - in 2 h 09 min (if less than 4 hours ahead)
  - Morgen 06:12
  - Übermorgen 20:15
  - otherwise the format Di 22.11. 14:30 shall be returned
- Create an object array **`object[] persons`** that stores person objects as above, but as anonymous objects.

```
var person = new { ...};
```

  Convert this array with **`JsonSerializer.Serialize(…)`** to a JSON string and print it to the console.
  Hint: this is more or less how WebApi works when using REST services.
- The method **Sort** (which is already available for collections) can be given as a parameter a delegate of the form **delegate int Comparison <T> (T x, T y)**. Call Sort with a lambda expression that sorts a string collection by the length of each string. The delegate method must therefore deliver:
  - 0 if both strings are the same length
  - 1 if the first string is longer than the second
  - -1 if the first string is shorter than the second