PRAKTIKUMSAUFGABE 03 IM SOSE'2021

Polymorphie, Typkonvertierung, Basisklasse **`System.Object`** und Schnittstellen

Aufgabenstellung:

Entwickeln Sie – basierend auf den vorgegebenen Quellcodeausschnitten auf den folgenden Seiten – eine „HTML Engine", die es erlaubt in der Programmiersprache C# HTML5-Inhalte zu beschreiben und daraus (dynamisch) HTML5-Code zu generieren. Das HTML-Dokument soll durch entsprechende C#-Objekte – so wie in der Hauptmethode **`Program.Main()`** vorgegeben – definierbar sein [siehe `Program.cs`]. Die Generierung selber soll von Ihnen in der statischen Methode **`HTML.Engine.Generate()`** [siehe `HTMLEngine.cs`] implementiert werden. In diesem Zusammenhang müssen Sie ebenfalls die folgenden, in **`Program.Main()`** verwendeten, HTML-Element-Klassen implementieren [in einer separaten Projektdatei `HTMLElements.cs`]:

C# Klasse	HTML Bezeichner („Tag ID"-String)	Verschachtelbar
`DocumentType`	`!DOCTYPE html`	Nein
`Html`	`html`	Ja
`Head`	`head`	Ja
`Title`	`title`	Ja
`Body`	`body`	Ja
`Heading1`	`h1`	Ja
`Heading2`	`h2`	Ja
`Paragraph`	`p`	Ja
`Italic`	`i`	Ja
`Bold`	`b`	Ja
`Underline`	`u`	Ja
`LineBreak`	`br/`	Nein

Jedes HTML-Element besitzt die grundlegende Eigenschaft, dass es mit einem spezifischen „HTML Tag" beschrieben wird. Diese Eigenschaft soll in der HTML-Engine dadurch ausgedrückt werden, dass jede C#-HTML-Element-Klasse die Schnittstelle **`ITagged`** implementiert [Deklaration bereits vorgegeben in `HTMLEngine.cs`]. Zusätzlich kann ein HTML-Element verschachtelt sein, d. h. es enthält dann Subelemente (die dann wiederum verschachtelt sein können). Dies soll mit der Schnittstelle **`INested`** ausgedrückt werden können [siehe `HTMLEngine.cs`]. Verschachtelte Elemente werden in HTML immer mit einem Anfangs- und End-Tag definiert, z. B. `<html>...</html>`. Nicht-verschachtelte Elemente haben im HTML-Code keinen End-Tag, z. B. `<br/>`. Die HTML-Engine soll auch Objekte verarbeiten können, die keine der beiden Schnittstellen implementiert haben, d. h. sowohl **`ITagged`** als auch **`INested`** nicht (z. B. Strings, Zahlen). Solche Objekte sollen intern einfach mittels **`System.Object.ToString()`** direkt in HTML-Code umgewandelt werden.

Vorgegebener (und abgeschlossener) Programmcode in Datei Program.cs:

```
using HTML;

class Program
{
  static void Main()
  {
    // Generate HTML document and store result in string:

    string html = Engine.Generate
    (
      new DocumentType(),

      new Html
      (
        new Head
        (
          new Title("Generated HTML5 Example")
        ),

        new Body
        (
          new Heading1("Welcome to the Technical University oAS Nuremberg"),

          new Paragraph
          (
            "We have a distinct profile ", new Underline("and"), " strive to maintain ",
            new Underline("our"), " leading position among comparable universities."
          ),

          new Heading2("Study for your future"),

          new Paragraph
          (
            12, " departments provide more than ", 40, " degree programs in ",
            new Bold("engineering, business, design and social sciences."),
            new LineBreak(),
            "If you have questions, please contact the ",
            new Italic("Student Counseling Service"), " or the ",
            new Italic("Student Office.")
          )
        )
      )
    );

    // Write resulting HTML string:

    System.Console.WriteLine(html);
    System.Console.ReadKey(true);

    // Show HTML string in default browser:

    System.IO.File.WriteAllText("Example.html", html);
    System.Diagnostics.Process.Start("Example.html");
  }
}
```

Vorgegebenes und zu erweiterndes Gerüst der „HTML Engine“ in Datei HTMLEngine.cs:

```
namespace HTML
{
  public interface ITagged { string TagId { get; } }

  public interface INested { object[] Elements { get; } }

  public static class Engine
  {
    public static string Generate(params object[] elements)
    {
      // TODO...
    }
  }
}
```

Tipp: Sie können auf dem beiliegenden Code-Gerüst, das die o. g. vorimplementierten Quellcode-Dateien enthält, aufbauen (Siehe „prog2_ss2021_praktikum_aufgabe_03-vorimplementierung.zip“).

Geforderte Ausgabe auf dem Bildschirm nach Ausführung des Programms:

```
<!DOCTYPE html>
<html>
<head>
<title>Generated HTML5 Example</title>
</head>
<body>
<h1>Welcome to the Technical University oAS Nuremberg</h1>
<p>
We have a distinct profile <u>and</u> strive to maintain <u>our</u> leading position
among comparable universities.
</p>
<h2>Study for your future</h2>
<p>
12 departments provide more than 40 degree programs in <b>engineering, business,
design and social sciences.</b><br/>
If you have questions, please contact the <i>Student Counseling Service</i> or the
<i>Student Office.</i>
</p>
</body>
</html>
```

Hinweis: Ihr resultierender HTML-Code muss nicht unbedingt so „schön“ – wie oben dargestellt – formatiert sein, d. h. Sie können die Zeilenumbrüche gerne auch anders setzen (oder notfalls sogar weglassen).

Geforderte Anzeige des folgenden HTML-Inhalts im Browser nach Ausführung des Programms:

Welcome to the Technical University oAS Nuremberg

We have a distinct profile and strive to maintain our leading position among comparable universities.

Study for your future

12 departments provide more than 40 degree programs in **engineering, business, design and social sciences.**
If you have questions, please contact the *Student Counseling Service* or the *Student Office.*

Anmerkung: Das vorgegebene Beispielprogramm erstellt die entsprechende HTML-Datei `Example.html` im „Build-Verzeichnis“ Ihres Projektes, z. B. `C:\Users\MyName\Source\Repos\HTMLEngine\bin\Debug`.

Tipps zur Implementierung der Generierungsmethode `HTML.Engine.Generate()`:

- Beachten Sie die vier unterschiedlichen Fälle, die ein Objekt erfüllen kann:
  Es ist „getaggt“, es ist „verschachtelt“, beides oder keins von beiden.
- Implementieren Sie die Verarbeitung von verschachtelten HTML-Elementen (siehe **`INested`**)
  rekursiv – Das wird viel einfacher als in der iterativen Version!

Tipps zur Implementierung der einzelnen HTML-Element-Klassen:

- Verwenden Sie für die verschachtelten HTML-Element-Klassen Konstruktoren, die eine variable Anzahl von Parametern unterstützen (Siehe Schlüsselwort **`params`**).
- Sie können (müssen aber nicht) auch eine Basis-Hilfsklasse implementieren, die die Funktionalität einer der Schnittstellen „vorimplementiert“. Dadurch könnten Sie sich etwas Tipparbeit bei der Implementierung sparen und so die Definition neuer HTML-Elemente vereinfachen.

Allgemeine Tipps:

- Der resultierende Quellcode ist nicht komplex oder umfangreich (für **`HTML.Engine.Generate()`** ca. 30-35 Zeilen, jede Elementklasse nur ein bis drei Zeilen)! Die Herausforderung bei dieser Aufgabe ist eher die *Konzeption vorab* und die Verwendung von Schnittstellen, Typumwandlung etc.
- Mit der folgenden Zeile können Sie Ihren HTML-Code überprüfen und zu Testzwecken („Debugging“) „schön formatiert“ ausgeben („Exception“ im Falle eines ungültigen HTML-Codes):

```
System.Console.WriteLine("\n<" + new DocumentType().TagId + ">\n" +
                         System.Xml.Linq.XElement.Parse(html.Replace("\n", "")));
```