10. Übungsblatt Thema: Generic Stack and Delegates

Implementierung eines generischen Stacks

Beschreibung:

Der Name der Klasse soll lauten GenericStack. Eine Instanze diese Klasse soll einem Datentypen als Parameter bekommen.

Dieser Typeparameter wird in der Aufgabenstellung als G bezeichnet.

Die Klasse soll 3 Dinge von außen zur Verfügung stellen.

- Methode Push: Soll einen Wert vom Typ G als Parameter annehmen und diesen an das Ender der innere Liste anhängen.
- Methode Pop: Soll den Wert vom letzten Element Typ G aus der inneren Liste zurück gegeben. Danach soll das letzte Element auch aus der List entfernt werden. Sollte die Liste bereits leer sein sollte eine Exception geworfen werden.
- Property Empty: Liefert true zurück falls der Stack gerade kein Element beinhaltet. Ansonsten soll false zurückgeliefert werden.

Test Program:

```
var stack = new GenericStack<int>();
stack.Push(2);
stack.Push(4);
stack.Push(8);
Console.WriteLine($"stack.Empty = {stack.Empty}");
Console.WriteLine($"stack.Pop() = {stack.Pop()}");
Console.WriteLine($"stack.Pop() = {stack.Pop()}");
Console.WriteLine($"stack.Pop() = {stack.Pop()}");
Console.WriteLine($"stack.Empty = {stack.Empty}");
```

Ausgabe sollte sein:

stack.Empty = False
stack.Pop() = 8
stack.Pop() = 4
stack.Pop() = 2
stack.Empty = True

---

2. Implementierung vom generischen IEnumerable

Beschreibung:

Die Klasse GenericStack soll nun auch das generische IEnumerable implementieren. Das IEnumerable soll also G als Type Parameter bekommen.

Es soll also möglich sein, eine foreach Schleife auf eine Instanze von GenericStack anzuwenden.

Tipp: Hier findet man mehr Infos zu den generischen Interface https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide/generics/generic-interfaces

---

Test Program:

```
var stack = new GenericStack<string>();
stack.Push("Max");
stack.Push("Graf Count");
stack.Push("Lucky one");
foreach (string name in stack)
{
    Console.WriteLine($"Name: {name}");
}
```

Ausgabe sollte sein:

Name: Lucky one
Name: Graf Count
Name: Max

---

3. Implementierung von Funktionen die mit IEnumerable sequences arbeiten

3.1 Implementierung von generischen ForEach

Beschreibung:

Lege eine statische Klasse namens ListUtility an. Lege in der Klasse eine delegate Namens ActionOnOneElement an.

Der Delegate soll einen Typeparameter bekommen. Dieser Delegate soll keine Rückgabewert haben und erwartet einen Parameter. Dieser Parameter ist von Type des gegebenen Typparameters des Delegates.

Diese Klasse soll eine statische Funktion namens ForEach implementieren. Diese Funktion erwartet beim Aufruf einen Typeparameter. Die Funktion erwartet 2 Parameter.

1. Parameter ist ein Objekt welches IEnumerable implementiert.
2. Parameter ist eine Reference auf eine Function von Type des delegates ActionOnOneElement

Die Funktion soll zu jedem Element des 1. Parameters jeweils 2. Parameter als Funktion aufrufen. Dabei ist jedes Element des 1. Parameters der Parameter für den Funktionsaufrufs.

Test Program:

```
var stack = new GenericStack<int>();
stack.Push(2);
stack.Push(4);
ListUtility.ForEach<int>(stack, element => Console.WriteLine(element.ToString()));
var list = new List<char>();
list.AddRange(new char[] { 'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd' });
ListUtility.ForEach<char>(list, letter => Console.Write(letter));
Console.WriteLine();
var commaNumbers = new double[] { 2.0, 4.0, 1.0, 3.0 };
double result = 0.0;
ListUtility.ForEach<double>(commaNumbers, number => result += number);
result /= Convert.ToDouble(commaNumbers.Length);
Console.WriteLine($"result = {result}");
```

Ausgabe sollte sein:

4
2
Hello World
result = 2,5

3.2 Implementierung von generischen Filter

Beschreibung:

Die Klasse soll einen weiteren delegate bekommen. Der delegate heist CheckRoutine. Der Delegate soll einen Typeparameter bekommen. Dieser Delegate hat einen Rückgabewert von Type bool und erwartet einen Parameter. Dieser Parameter ist von Type des gegebenen Typparameters des Delegates.

In der statischen Klasse ListUtility soll nun noch die statische Funktion Filter implementiert werden. Auch diese Funktion bekommt einen Typparameter.

1. Parameter ist ein Objekt welches IEnumerable implementiert.
2. Parameter ist eine Reference einer Function von Type des delegates CheckRoutine. Und return eine Objekt von Type IEnumerable.

Der Rückgabewert soll alle Elemente enthalten welche als Parameter beim Aufruf von CheckRoutine ein true zurück liefert.

Tipp: Objekte von Type GenericStack und andere Datenkonstrukte wie List, Stack oder Queue aus dem Namespace System.Collections.Generics können auch vom Type IEnumerable sein.

Test Program:

```
var list = new List<string>(new string[] { "Auto", "Haus", "mit", "Arbeit", "Code",
"Abs", "ab" });
var result = ListUtility.Filter<string>(list, word => word.Length > 3);
Console.Write("( ");
foreach (string word in result)
{
    Console.Write($"{word} ");
}
Console.WriteLine(")");
var numberStack = new GenericStack<int>();
numberStack.Push(2);
numberStack.Push(5);
numberStack.Push(12);
numberStack.Push(73);
numberStack.Push(440);
var evenNumbers = ListUtility.Filter<int>(numberStack, number => number % 2 == 0);
foreach (int evenNumber in evenNumbers)
{
    Console.WriteLine($"Even number: {evenNumber}");
}
```

Ausgabe sollte sein:

(Code Arbeit Haus Auto)
Even number: 2
Even number: 12
Even number: 440
