Class Kata „Stack“

Implementiere den abstrakten Datentyp *Stack*.

Der Stack als Datenstruktur ist ein *first-in, last-out Speicher*. Elemente, die auf den Stack gelegt werden, werden in umgekehrter Reihenfolge von diesem zurück gegeben. Das zu implementierende Interface des Stack sieht wie folgt aus:

```
public interface IStack<TElement>
{
    void Push(TElement element);

    TElement Pop();
}
```

Die Datenstruktur soll als generischer Typ implementiert werden. Sie verfügt über die beiden Operationen *Push* und *Pop*. Mit *Push* wird ein Element auf den Stack gelegt. *Pop* liefert das oberste Element des Stacks und entfernt es vom Stack. Wird *Pop* auf einen leeren Stack angewandt, wird eine *InvalidOperationException* Ausnahme ausgelöst.

Folgende Tabelle zeigt, wie sich ein Stack verhält:

Zustand des Stacks	**Operation**	**Ergebnis**
Leer	Push(5)	-
5	Pop()	5
Leer	Pop()	Exception
Leer	Push(7)	-
7	Push(9)	-
7, 9	Push(1)	-
7, 9, 1	Pop()	1
7, 9	Pop()	9
7	Pop()	7
Leer		

Variation

Verwende beim automatisierten Testen der Push-Operation nicht die Pop-Operation. Ebenso verwende beim Testen der Pop-Operation nicht die Push-Operation.

Natürlich sollte es auch Tests geben, die demonstrieren, wie Push und Pop im Zusammenspiel funktionieren. Um aussagekräftige Testergebnisse zu erhalten, ist es nützlich, die Tests der einzelnen Operationen voneinander zu isolieren. Wenn der Test der Push-Operation fehlschlägt, der für Pop jedoch erfolgreich ist, sagt dies präziser aus, wo das Problem liegt. Ferner ist auf diese Weise eine Implementation der beiden Operationen in beliebiger Reihenfolge und unabhängig voneinander möglich.