Versuchen Sie immer, Ihren Code zu kommentieren!

Aufgabe 1

Erstellen Sie eine generische Flasche, welche mit verschiedenen Getränken gefüllt werden kann. Eine Flasche hat eine Variable inhalt, welches den Inhalt der Flasche angibt.
Sie hat die Methode IstLeer(), welche true zurückliefert wenn die Flasche leer ist. Die Methode Füllen(), welche die Flasche mit einem Getränk füllt. Und die Methode Leeren(), welche das gefüllte Getränk als Rückgabeparameter liefert und die Flasche leert.

Getränke sind Bier, Rotwein und Weißwein. Jedes Getränk hat einen Namen.

- Biere haben eine Brauerei, eine Methode, die die Brauerei als String zurückliefert und eine Methode, welche Informationen zum Bier auf der Konsole ausgibt.

- Weine haben eine Herkunft. Zudem zwei Methoden, eine die die Herkunft als String zurückliefert und eine die die Informationen zum Wein auf der Konsole ausgibt.

Erstellen Sie innerhalb von Main einige Flaschen und Getränke und füllen Sie die Flaschen mit den unterschiedlichen Getränken.

Zusatz:
Verwenden Sie Constraints um sicherzustellen, dass nur Getränke in die Flaschen gefüllt werden können.

Aufgabe 2

Erstellen Sie eine parametrisierte Klasse SimpleList, die eine einfach verkettete lineare Liste beschreiben soll. Ihre Implementierung muss folgende Operationen unterstützen:

1. Elemente in die Liste einfügen
2. Elemente aus der Liste löschen
3. Elemente in der Liste suchen
4. Anzahl der gerade in der Liste enthaltenen Elemente zurückgeben
5. Aus der Liste ein Array von Elementen erzeugen

Ihre Liste soll dabei mit allen Typen arbeiten können, die das Interface IComparable implementieren.

Aufgabe 3

Erstellen Sie auf der Definition der Liste aus Aufgabe 2 eine parametrisierte Klasse Set. Diese Klasse soll eine Menge beschreiben. Eine Menge unterscheidet sich von der einer Liste darin, dass jedes Element nur **genau einmal** in der Menge auftauchen darf.

Ihre Implementierung der Menge soll folgende Operationen enthalten:

1. Elemente in die Menge einfügen
2. Elemente aus der Menge löschen
3. Elemente in der Menge suchen
4. Anzahl der gerade in der Menge enthaltenen Elemente zurückgeben
5. Aus der Menge ein Array von Elementen erzeugen
6. Zwei Mengen gleichen Typs vereinigen können
7. Die Schnittmenge zweier Mengen gleichen Typs bilden können

Aufgabe 4

Implementieren Sie einen binären Baum (BST = binary search tree) mit dem folgenden Interface:

```
public interface IBinaryTree<T> where T : IComparable<T>
{
    void Clear();
    void Insert(T value);
    void Delete(T value);
    bool Contains(T value);
    BinaryTreeNode<T> Search(T value);
    void PrintInorder();
}

public class BinaryTreeNode<T> where T : IComparable<T>
{
    public T Data { get; set; }
    public BinaryTreeNode<T> Left { get; set; }
    public BinaryTreeNode<T> Right { get; set; }

    public BinaryTreeNode(T value)
    {
        Data = value;
    }
}
```

Testen Sie die Klasse! Fügen Sie dazu die folgenden Werte der Reihe nach ein:

50, 100, 25, 1, 10, 75, 65, 85, 61, 45, 35, 15, 10

Rufen Sie anschließend die Methode `PrintInorder()` auf. Werden die Zahlen aufsteigend sortiert ausgegeben?