Arbeitsauftrag 2A HIFS

Programmieren Sie einen Binären Suchbaum in dem ganze Zahlen gespeichert werden können und der folgende Operationen zur Verfügung stellt:

- Werte in den Baum einfügen
    - Der Baum soll nur eindeutige Werte speichern. Sollte versucht werden, eine Zahl, die bereits im Baum vorhanden ist erneut einzufügen, so soll eine Exception ausgelöst werden.
- Werte suchen
- Alle Werte in Form einer Liste retour geben lassen.

Die Programmierung des Binären Suchbaums soll in einem eigenen Class-Library-Projekt realisiert werden, welches mit einem Oberflächen-Projekt (wahlweise Console App oder WPF) zu einer Lösung kombiniert wird.

Nennen Sie das Projekt mit dem Binären Suchbaum „{Nachname}_MyDataStructures“.

Vergleich Einfach verkettete Liste und Binärer – Suchbaum

Die beiden Datenstrukturen sollen hinsichtlich Such-Performance verglichen werden. Verwenden Sie dazu ihre eigene einfach verkettete Liste (verwenden Sie nicht die generische Liste von C#!). Die Liste soll in das Class-Library Projekt integriert werden.

Um den Performance-Test mit den beiden Datenstrukturen durchzuführen, müssen die Datenstrukturen mit (sehr vielen) Werten gefüllt werden. Diese Informationen liegen in Form eines Files vor.

Nachdem die beiden Datenstrukturen mit den Werten gefüllt wurden, sollen die Suchoperationen durchgeführt werden. Speichern Sie dabei die zu suchenden Werte in einer Liste ab (dies können Zufallszahlen sein). Jede Zahl in dieser Liste soll dann im Baum bzw. in der Liste gesucht werden. Man muss mehrere Werte suchen, damit man die Zeitdauer der Suche messen kann.

Zeitmessungen führen Sie mit der Stop-Watch-Klasse durch:

https://msdn.microsoft.com/de-de/library/system.diagnostics.stopwatch(v=vs.110).aspx

Hier ein weiteres Beispiel einer Zeitmessung (misst wie lange es dauert, die Testdaten zu generieren):

```
Stopwatch stopWatch = new Stopwatch();
stopWatch.Start();

for (int i = 0; i < numbers; i++)
{
    generatedNumbers.Add(rnd.Next(NumbersFrom, NumbersTo));
}

// In TextDatei schreiben
using (BinaryWriter writer = new BinaryWriter(File.Open("randomNumbers.bin", FileMode.Create)))
{
    foreach (int i in generatedNumbers)
        writer.Write(i);
}

stopWatch.Stop();
// Get the elapsed time as a TimeSpan value.
TimeSpan ts = stopWatch.Elapsed;

Console.WriteLine("Random DataFile successfully generated. It took {0} milliseconds", ts);
```