Lab 4 – Operator overloading & Extension methods

Hinweis: Das Aufgabenblatt enthält neben den Aufgaben zum Modulpraktikum auch Zusatzfragen zum Verständnis und zur Repetition des Stoffes. Diese Fragen müssen nicht für die Fallstudie beantwortet werden und sind freiwillig. Sie können jedoch zur Prüfungsvorbereitung verwendet werden.

Aufgabe 1 - Dijkstra

Zunächst sollen Sie in der Klasse `Links` in der Methode `FindShortestRouteBetween` einen Suchalgorithmus implementieren.

Verwenden Sie dazu den Dijkstra-Algorithmus, der die kürzeste Route zwischen den Städten `fromCity` und `toCity` berechnet, die mit dem übergebenen `TransportMode mode` zurückgelegt werden kann.

a) Fügen Sie den Code des Dijkstra-Algorithmus aus `DijkstraAlgorithm.cs` der Klasse `Links` hinzu. Achten Sie darauf, dass Ihr Event-Aufruf erhalten bleibt.

   Damit der Code funktioniert müssen Sie zwei zusätzliche Methoden implementieren. Aus den Datentypen und den Code-Kommentaren lässt sich auf die geforderte Funktionalität schliessen.

   Verwenden Sie das yield Statement wo möglich.

   Für die Interessierten: Details zum Dijkstra-Algorithmus finden Sie in der Wikipedia (http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm).

b) Überschreiben Sie `City.Equals()` so, dass zwei `City`-Objekte gleich sind, wenn sowohl deren Name als auch deren Land übereinstimmen. Gross-/Kleinschreibung sollen dabei nicht unterschieden werden. Beachten Sie die Guidelines von Microsoft: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/how-to-define-value-equality-for-a-type

Aufgabe 2 – Operator overloading

Erweitern Sie die Klasse `WayPoint` so, dass zwei `WayPoints` mittels `+`, `-`, `+=` und `-=` addiert bzw. subtrahiert werden können. Die Operatoren sollen so funktionieren, dass die Koordinaten addiert bzw. subtrahiert werden und der Name des ersten `WayPoints` erhalten bleibt.

Aufgabe 3 – Extension methods & yield

a) Das Einlesen und Parsen von Textfiles, wie es in `Cities.ReadCities()` gemacht wird, ist eine Funktion, die auch in anderen Klassen und Projekten nützlich sein könnte. Implementieren Sie eine neue Extension-Methode für die Klasse `TextReader`, die eingelesenen Textzeilen direkt als Liste von String-Arrays zurückgibt und folgendermassen verwendbar ist:

```
public int ReadCities(string filename)
{
   using (TextReader tr = new StreamReader(filename))
   {
      IEnumerable<string[]> citiesAsStrings = tr.GetSplittedLines('\t');

      //for…
      {
           cities.Add(new City(cs[0].Trim(), cs[1].Trim(),
                   int.Parse(cs[2]),
                   double.Parse(cs[3], CultureInfo.InvariantCulture),
                   double.Parse(cs[4], CultureInfo.InvariantCulture)));
      }

      //…
```

Implementieren Sie die entsprechende Extension-Methode und ersetzen Sie den bestehenden Einlese-Code mit obigem Code. Legen Sie die Extension Methode in der Klasse `TextReaderExtensions` im Namespace `Fhnw.Ecnf.RoutePlanner.RoutePlannerLib.Util` (d.h. im Unterverzeichnis Util) an.

b) Da die neue Extension-Methode sehr generisch ist, können Sie diese auch gerade für das für das Einlesen der Links-Datei in der `ReadLinks` Methode der `Links` Klasse verwenden. Führen Sie auch diese Änderung entsprechend durch.

Aufgabe 4

Testen Sie Ihre Implementierung mit den Unit-Tests in den Files «`*Test_Lab04.cs`» im «`caseStudyFiles`» Ordner. Beachten Sie die zusätzlichen Testdaten-Files.

Testfragen

- Welche Möglichkeiten gibt es, um bestehende Klassen um eigenen Code zu ergänzen? Was sind die jeweiligen Vor- bzw. Nachteile?
- Was sind nullable types? Wie funktionieren sie?
- Was sind die Vor- und Nachteile von unsafe code?
- Was sind anonyme Typen? Was sind deren Vor- und Nachteile?