Lab 07 – Asynchronous Programming

Aufgabe 1

Die Berechnung der kürzesten Strecke zwischen zwei Orten kann relativ lange dauern. Daher soll die Ausführung dieser Funktion asynchron durchgeführt werden.

Implementieren Sie dazu neben der bereits bestehenden synchronen Berechnungsmethode eine asynchrone Methode `FindShortestRouteBetweenAsync`, die die synchrone Methode aufruft und deren Ergebnis zurückgibt.

Die asynchrone Ausführung soll nach dem Task-based Asynchronous Pattern (TAP) realisiert werden; die asynchrone Ausführung soll ebenfalls die gefundene Route zurückgeben.

Aufgabe 2

Erweitern Sie die Implementation aus Aufgabe 1 um die Möglichkeit eines Fortschritt-Reports. Bauen Sie in der Methode `FindShortestRouteBetween` dazu nach den verschiedenen Schritten entsprechende Aufrufe des Fortschritt-Callbacks ein. Es soll jeweils eine Text-Meldung ausgegeben werden, welche Aktion gerade durchgeführt wurde:

1. Es müssen mindestens 5 Fortschritts-Aufrufe durchgeführt werden
2. Jede Meldung, die übergeben wird, soll der Struktur „`<action> done`“ folgen, wobei `<action>` für die gerade durchgeführte Aktion steht.

Hinweis: Für diese Aufgabe müssen Sie die Originalmethode `FindShortestRouteBetween` um einen zusätzlichen `IProgress` Parameter erweitern, z.B.:

```
public List<Link> FindShortestRouteBetween(string fromCity, string toCity,
                          TransportMode mode, IProgress<string> reportProgress)
{
    // hier die bisherige, erweiterte Implementierung
}
```

Um grosse Refactorings zu vermeiden, können Sie den neuen Parameter mit dem Default Wert null initialisieren. Dann läuft auch der bisherige Code weiter, bei dem die Methode ohne den neuen Parameter aufgerufen wurde.
Wichtig: Sie müssen jedoch beim Aufruf der Report-Methode auf null abtesten (Stichwort nullable types)

Aufgabe 3

Testen Sie Ihre Implementierung mit den Unit-Tests in den Files «`*Test_Lab07.cs`» im «`caseStudyFiles`» Ordner.

Testfragen

- Die Keywords `async`/`await` werden vom Compiler interpretiert und in entsprechenden C#-Code umgesetzt. Erklären Sie mit eigenen Worten die Funktionsweise der beiden Keywords; insbesondere die Rollen der Konzepte `TaskCompletionSource` und `ContinueWith()`.
- Was sind die Vorteile von `async`/`await` gegenüber dem manuellen Einsatz von Threads?
- Sie schreiben eine Applikation, die tausende Anfragen an Web-Sites macht. Warum ist hierfür die Verwendung der Thread-API (also für jede Anfrage einen eigenen Thread erzeugen) ungeeignet? Warum entsteht dieses Problem mit dem `async`/`await` Konzept nicht?
- Kann man mit `async`/`await` einen Sortier-Algorithmus auf einer Multi-Core-CPU beschleunigen?