HAWK Fakultät Naturwissenschaften und Technik Göttingen

Windowsprogrammierung – WiSe17

Übung 3 Events, Generics und Interfaces

Aufgabe 3.1

Erweitern Sie die in Übung 1 definierte Klasse **Film** um die folgenden 3 öffentlichen Events:

- **FilmAngesehen**
- **IstAllzeitFavoritGeworden**
- **BewertungGeändert**

Feuern Sie diese Events an den entsprechenden Stellen innerhalb der **Film**-Klasse. Das Event **BewertungGeändert** soll nur dann ausgelöst werden, wenn sich die **Bewertung** wirklich geändert hat. Als Ereignishandler für die Events **FilmAngesehen** und **IstAllzeitFavoritGeworden** sollen alle Methoden mit der folgenden Signatur in Frage kommen:

```
void BehandleEreignis(object sender)
```

Für das **BewertungGeändert**-Ereignis soll hingegen folgende Signatur verwendet werden:

```
void BehandleEreignis(object sender, int? neueBewertung)
```

Definieren Sie entsprechende Delegat-Typen, um diesen Anforderungen gerecht zu werden. Testen Sie die erweiterte Funktionalität der **Film**-Klasse, indem Sie eine **Film**-Instanz erzeugen, sich bei ihren 3 Events registrieren und die Instanz anschließend so benutzen, dass das Auslösen der Events forciert wird. Geben Sie für jedes auftrende Event eine entsprechende Meldung in der Konsole aus.

Aufgabe 3.2

Gegeben ist das folgende generische Interface:

```
interface IEventListe<T>
{
    T this[int index]
    {
        get;
```

```
        set;
    }

    int AnzahlElemente
    {
        get;
    }

    void ElementHinzufügen(T element);
    void ElementEntfernen(T element);

    event System.EventHandler<T> ElementHinzugefügt;
    event System.EventHandler<T> ElementEntfernt;
}
```

Bei dem obersten Klassenmember (`T this[int index]...`) handelt es sich um einen sogenannten Indexer, der einen Array-ähnlichen Zugriff auf die einzelnen Listenelemente ermöglichen soll. Das C#-Sprachkonzept eines Indexers wurde in der Vorlesung nicht behandelt. Schauen Sie im Internet nach, was man unter einem Indexer in C# versteht. Bei `EventHandler<T>` handelt es sich um einen in die .NET-Klassenbibliothek eingebauten generischen Delegat-Typen, der sinngemäß wie folgt definiert ist:

```
delegate void EventHandler<TEventArgs>(object sender, TEventArgs e);
```

Schreiben Sie eine generische Klasse **MeineEventListe<T>**, die das Interface **IEventListe<T>** implementiert (d.h. davon erbt). Die Klasse soll über ein privates Feld vom Typ **List<T>** verfügen, mit dessen Hilfe Sie den Indexer, die Eigenschaft sowie die beiden Methoden implementieren können. Beim Hinzufügen oder Entfernen eines Elementes soll das entsprechende Event gefeuert werden. Achten Sie beim Event **ElementEntfernt** darauf, dass dieses nur dann gefeuert wird, wenn wirklich ein Element aus der Liste entfernt wurde.

Schreiben Sie ein Testprogramm, in Sie die Funktionsweise der Klasse **MeineEventListe<T>** für 3 unterschiedliche Typparameter demonstrieren.

Hinweis: Schreiben Sie am besten eine oder mehrere generische Testmethoden, die die Funktionalität der Klasse für einen beliebigen Typparameter testen, und rufen Sie diese Methode(n) für unterschiedliche Typparameter auf.