Handout Events

Events in C# sind eine Möglichkeit für Objekte, miteinander zu kommunizieren

Sie folgen dem publish-subscribe Muster

Sie werden verwendet, um "Abonnenten" zu benachrichtigen, wenn etwas passiert

Delegates

Events basieren auf Delegates

Ein Delegate ist ein Typ, welcher Referenzen zu Methoden speichern kann

Zudem werden Delegate genutzt, um die Signatur von Methoden zu definieren:

```
public delegate void MyEventHandler(string message);
```

Events deklarieren

```
[public|protected|internal|private] event Delegatmethode Eventname
```

```
public event MyEventHandler _myEvent;
```

Die Klasse, welches das Event besitzt, wird "Publisher" genannt

Event auslösen

Info: `EventHandler` ist vom System vordefiniert und akzeptiert 2 Parameter:
`object` *sender* und System.`EventArgs` *e*

Ausgelöst wird das Event durch: *Eventname?*.`Invoke`(*parameter*);

```
public class EventPublisher
{
    public event EventHandler MyEvent;
    public void TriggerEvent()
    {
         MyEvent?.Invoke(this, EventArgs.Empty);
      // ? -> Wenn Event == null, dann wird es nicht ausgelöst
    }
}
```

Event abonnieren

```
public class Subscriber
{
    private EventPublisher _publisher;
    public Subscriber(EventPublisher publisher)
    {
        _publisher = publisher;
        _publisher.MyEvent += HandleEvent;
    }
    public void HandleEvent(object sender, EventArgs e)
    {
        Console.WriteLine("Subscriber received the event!");
       _publisher.MyEvent -= HandleEvent;
    }
}
```

Generics

```
public class CustomEventArgs : EventArgs
{
    public string Message { get; }
    public int Value { get; }

    public CustomEventArgs(string message, int value)
    {
        Message = message;
        Value = value;
    }
}
```

Anschließend zuweisen:

```
public event EventHandler<CustomEventArgs> MyEvent;
```

…und auslösen:

```
public void TriggerEvent()
{
     MyEvent?.Invoke(this, new CustomEventArgs("Hello World", 42));
}
```