
07 – Delegates und Events

Agenda

- Delegates
- Events

Delegates

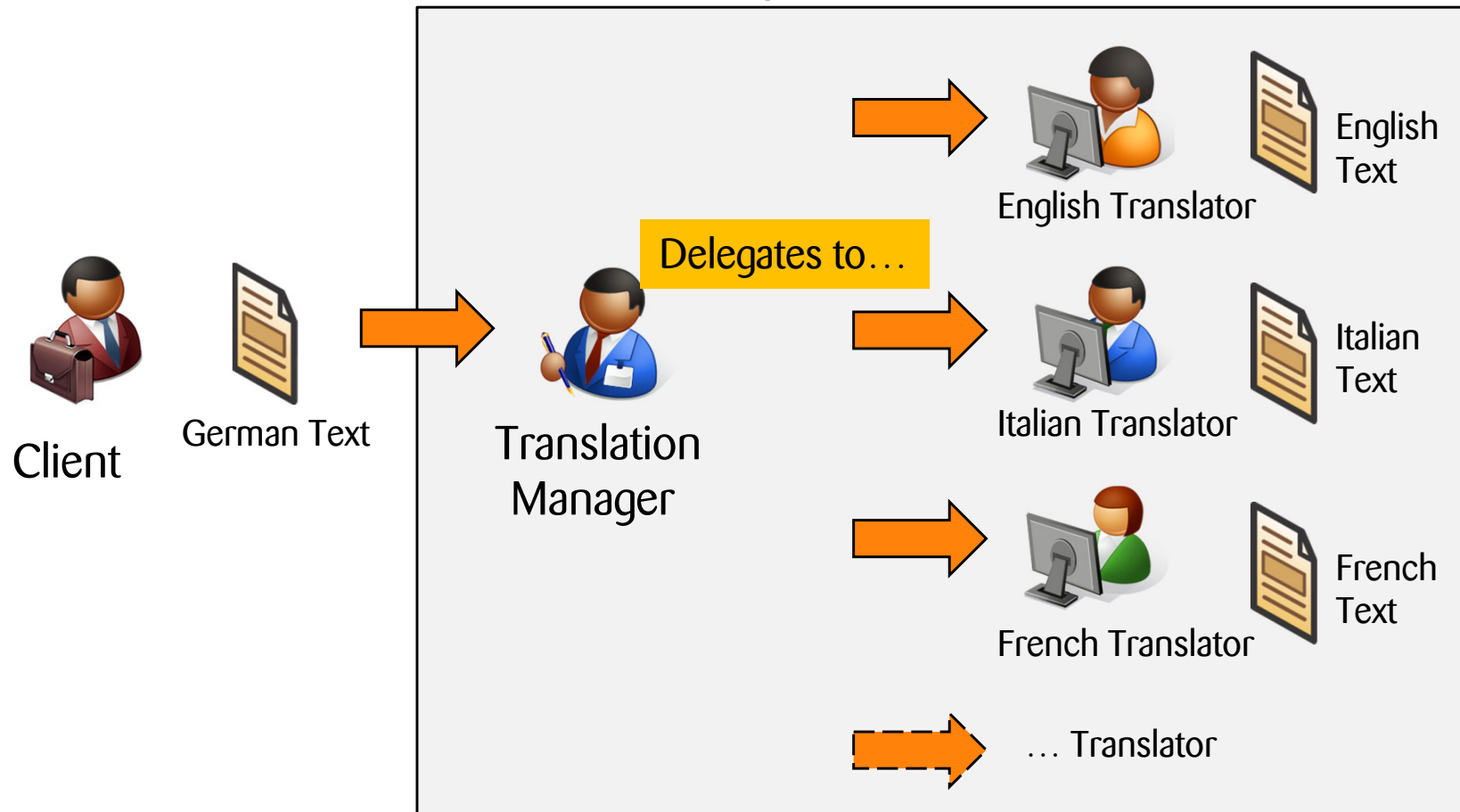
- Ein Delegate ist technisch gesehen eine Klasse die von `System.MulticastDelegate` ableitet
- Funktioniert ähnlich wie eine Referenz auf eine oder mehrere Methoden
- Ersetzt den klassischen Funktionszeiger
- Deklariert wird nur der Kopf der späteren Funktion, implementiert wird die Funktion in der Klasse

```
public delegate void WriteHandler(string word);
```

Delegates

Ausgangslage

- Text übersetzen in diverse Fremdsprachen



Delegates

Handler-Implementation

```
public delegate void WriteHandler(string word);
```

```
public class EnglishWriter
{
    private

    public void Write(string word)
    {
        Console.WriteLine(dict[word]);
    }
}
```

Methode mit gleicher
Signatur wie der Delegate

Implementierung Translation-Manager

```
public delegate void WriteHandler(string word);

public class OutputManager
{
    public WriteHandler Writers;

    public void Write(string word)
    {
        if (Writers != null)
        {
            Writers(word);
        }
    }
}
```

Delegates - Hinzufügen

Anwendung der Delegates

```
class Program
{
    static void Main(string[] args)
    {
        OutputManager outputManager = new OutputManager();
        ItalianWriter italianWriter = new ItalianWriter();
        EnglishWriter englishWriter = new EnglishWriter();

        outputManager.Writers += new WriteHandler(italianWriter.Write);
        outputManager.Writers += new WriteHandler(englishWriter.Write);

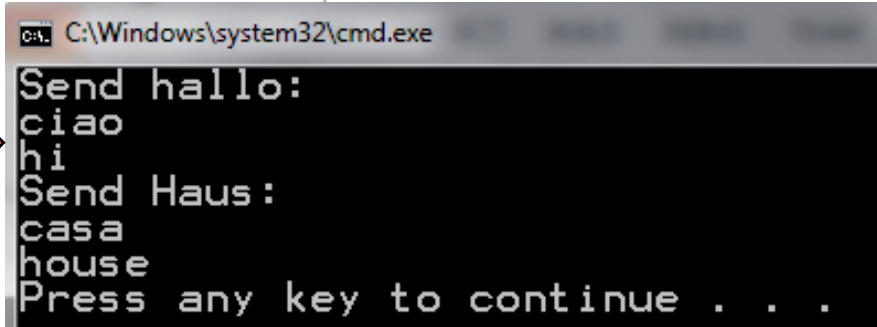
        string word = "hallo";
        Console.WriteLine("Send {0}: ", word);
        outputManager.Write(word);

        word = "Haus";
        Console.WriteLine("Send {0}: ", word);
        outputManager.Write(word);
    }
}
```

Delegates
hinzufügen



Output



```
C:\Windows\system32\cmd.exe
Send hallo:
ciao
hi
Send Haus:
casa
house
Press any key to continue . . .
```

Delegates - Entfernen

```
class Program
{
    static void Main(string[] args)
    {
        OutputManager outputManager = new OutputManager();
        ItalianWriter italianWriter = new ItalianWriter();
        EnglishWriter englishWriter = new EnglishWriter();

        outputManager.Writers += new WriteHandler(italianWriter.Write);
        outputManager.Writers += new WriteHandler(englishWriter.Write);

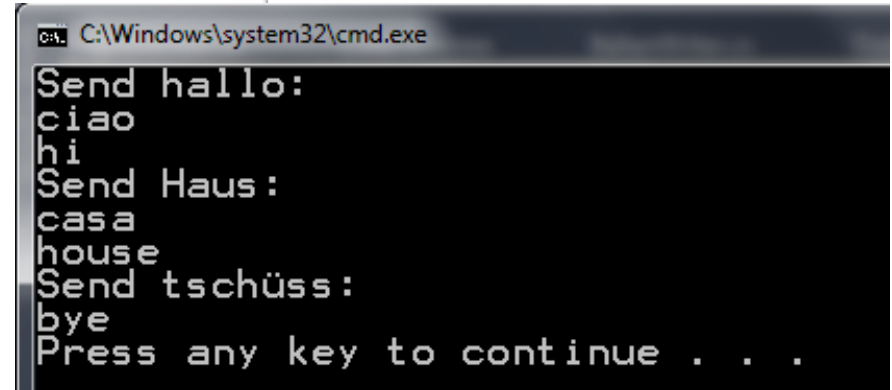
        string word = "hallo";
        Console.WriteLine("Send {0}: ", word);
        outputManager.Write(word);

        word = "Haus";
        Console.WriteLine("Send {0}: ", word);
        outputManager.Write(word);

        outputManager.Writers -= new WriteHandler(italianWriter.Write);

        word = "tschüss";
        Console.WriteLine("Send {0}: ", word);
        outputManager.Write(word);
    }
}
```

Output



```
C:\Windows\system32\cmd.exe
Send hallo:
ciao
hi
Send Haus:
casa
house
Send tschüss:
bye
Press any key to continue . . .
```

Delegates
entfernen

Übung 7.1



Delegates

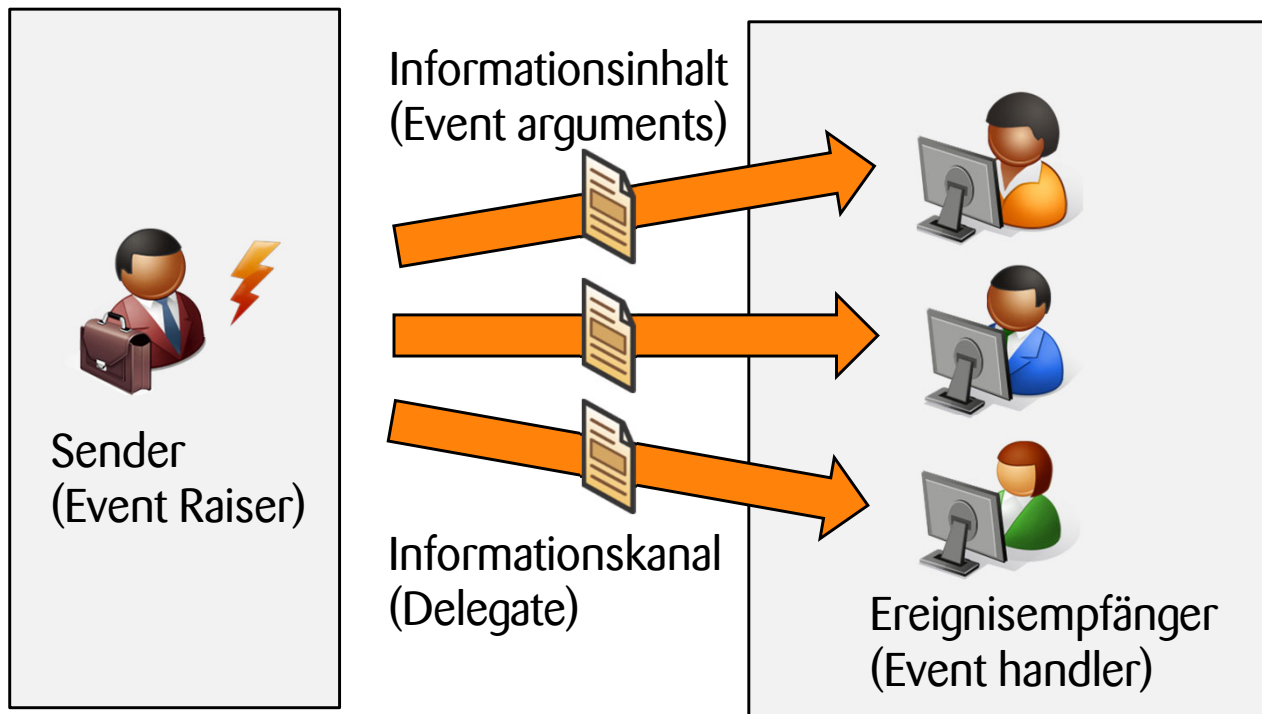
- 1) Deklariere ein Delegate `SendHandler`, welcher Meldungen versendet.
- 2) Schreibe zwei Klassen `EmailSender` und `SMSSender`. Beide haben eine Methode `Send()`, welche eine Meldung vom Typ `string` versenden kann.
Anmerkung: zur Vereinfachung soll die Meldung auf die Konsole geschrieben werden.
- 3) Schreibe eine Klasse `MessageManager`, welche Meldungen via Delegates (Email und SMS) versenden kann
- 4) Schreibe eine Klasse `TestTreiber`, welche die je nach Tastatureingabe die Meldung als E-Mail, SMS oder beides versendet.

Beispiel Konsolen Ausgabe:

SMS: Hallo aus Uster :-)

EMAIL: --- Hallo aus Uster ---

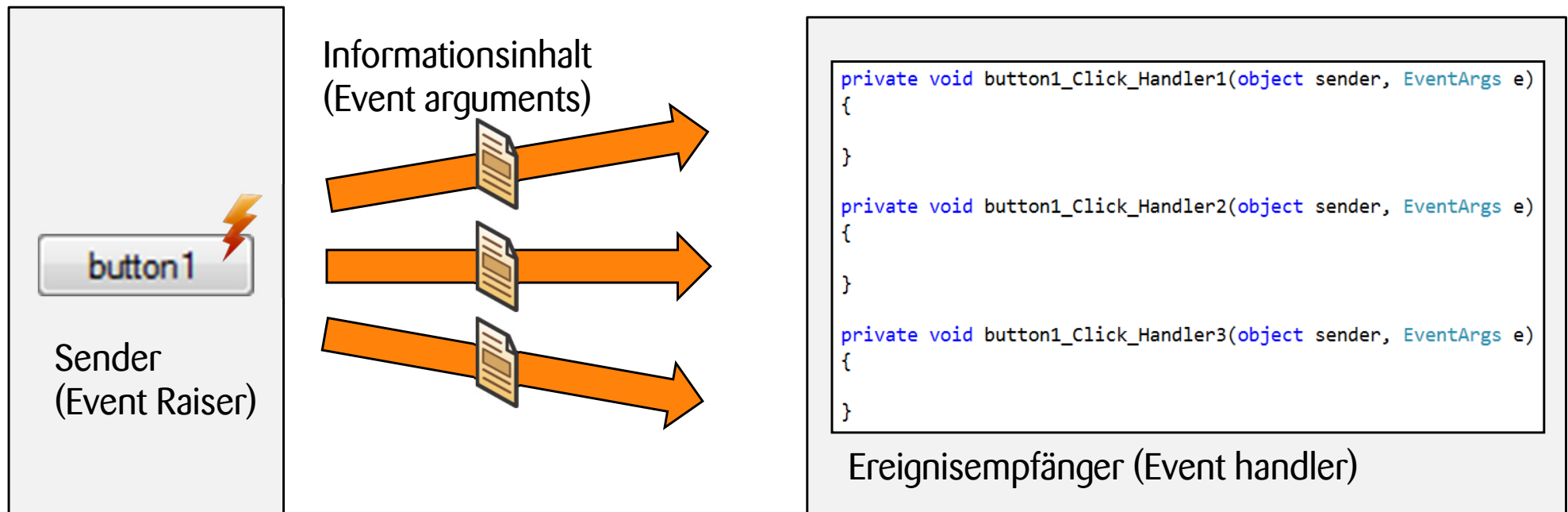
Einführung



Events

Einführung

```
this.button1 = new System.Windows.Forms.Button();
```



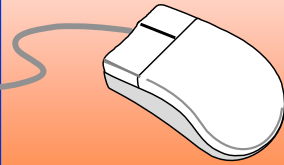
```
this.button1.Click += new System.EventHandler(this.button1_Click_Handler1);
this.button1.Click += new System.EventHandler(this.button1_Click_Handler2);
this.button1.Click += new System.EventHandler(this.button1_Click_Handler3);
```

Folie 11

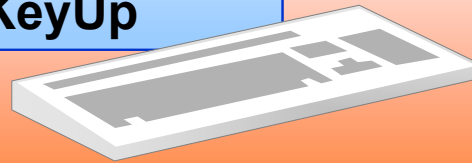
Windows-Events

- **Maus und Tastatur**

MouseDown,
MouseUp,
MouseMove,
MouseEnter,
MouseLeave,
MouseHover



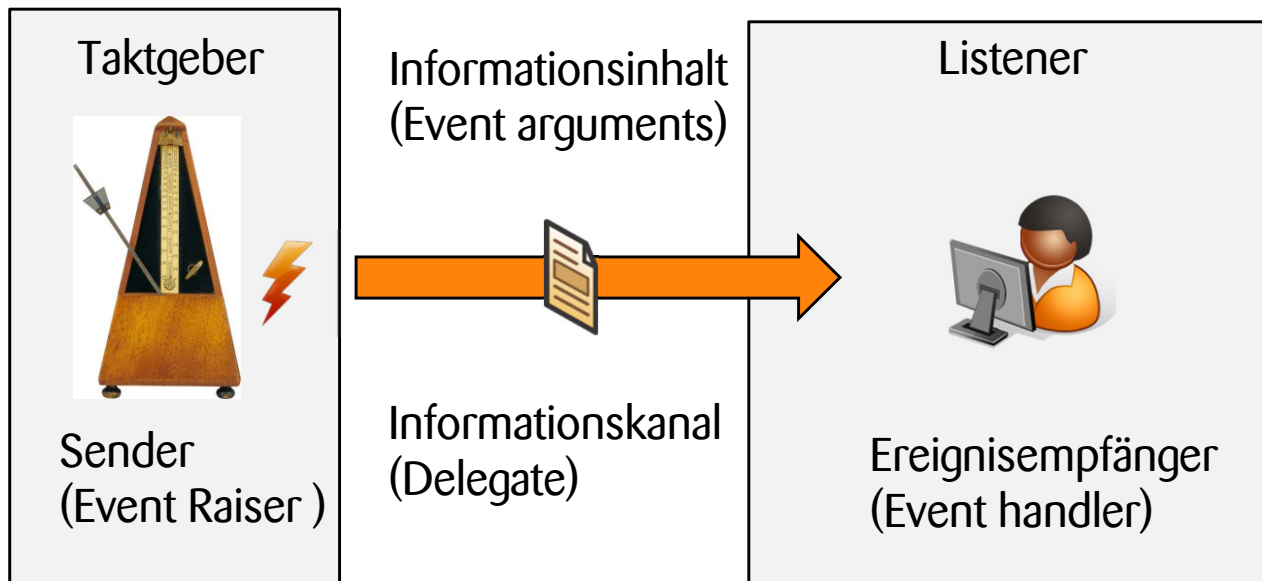
KeyPress,
KeyDown,
KeyUp



Eigene Events definieren

- Syntax: `event <Delegate> <EventName>;`

1. Beispiel – Taktgeber → Listener (1/4)

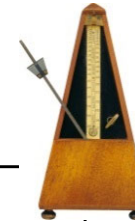


1. Beispiel – Taktgeber → Listener (2/4)

■ Taktgeber

```
public class SimpleTaktgeber
{
    public delegate void TickHandler(SimpleTaktgeber m, EventArgs e);
    public event TickHandler Tick;
    public EventArgs e = null;
    public int Interval { get; set; }

    public void Start()
    {
        while (true)
        {
            Thread.Sleep(Interval);
            if (Tick != null)
            {
                Tick(this, e);
            }
        }
    }
}
```

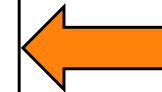


1. Beispiel – Taktgeber → Listener (3/4)

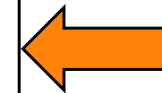
■ Listener

```
public class SimpleListener
{
    public void Subscribe(SimpleTaktgeber m)
    {
        m.Tick += new SimpleTaktgeber.TickHandler(HeardIt);
    }

    private void HeardIt(SimpleTaktgeber m, EventArgs e)
    {
        System.Console.WriteLine("Simple Taktgeber: Takt empfangen");
    }
}
```



Registrierung



Event-handler

1. Beispiel – Taktgeber → Listener (4/4)

■ Testprogramm

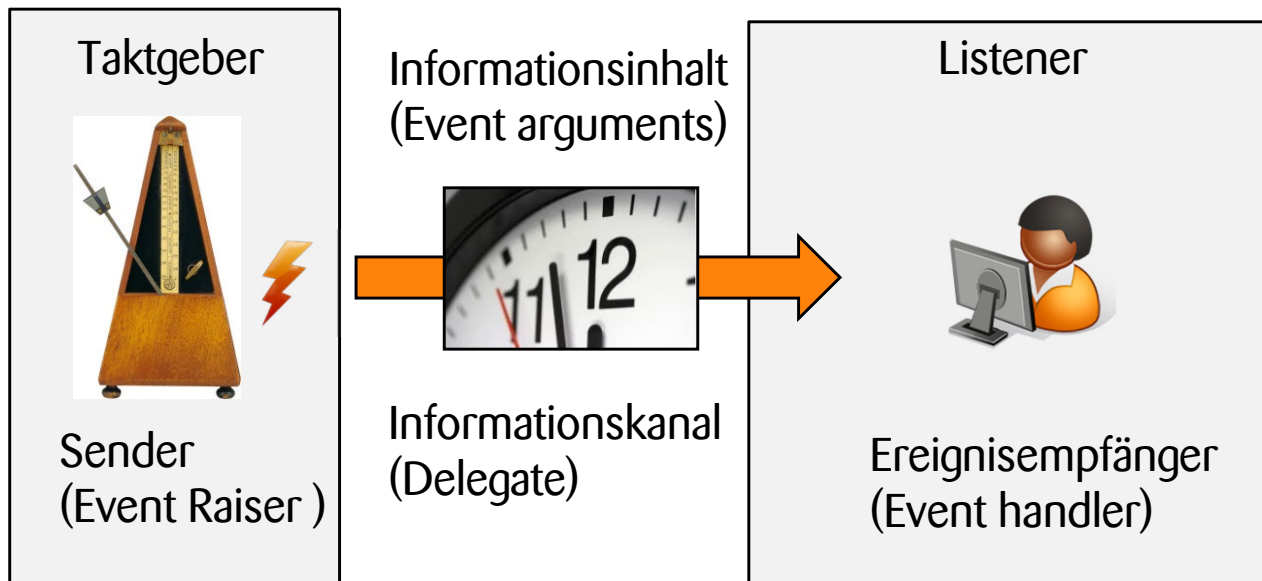
```
class Program
{
    static void Main(string[] args)
    {
        TestSimpleTaktgeber();
    }

    static void TestSimpleTaktgeber()
    {
        SimpleTaktgeber taktgeber = new SimpleTaktgeber() { Interval = 2000 };
        SimpleListener listener = new SimpleListener();
        listener.Subscribe(taktgeber);
        taktgeber.Start();
    }
}
```

Output:

```
Simple Listener: Takt Event empfangen
Simple Listener: Takt Event empfangen
Simple Listener: Takt Event empfangen
Simple Listener: Takt Event empfangen
Simple Listener: Takt Event empfangen
Simple Listener: Takt Event empfangen
```


2. Beispiel – Taktgeber mit Custom Event Args (1/4)



2. Beispiel – Taktgeber mit Custom Event Args (2/4)

■ Taktgeber

```
public class Taktgeber
{
    public delegate void TickHandler(Taktgeber m, EventArgs e);
    public event TickHandler Tick;
    public EventArgs e = null;
    public int Interval { get; set; }

    public void Start()
    {
        while (true)
        {
            Thread.Sleep(Interval);
            if (Tick != null)
            {
                EventArgs args = new EventArgs();
                args.EventTime = DateTime.Now;
                Tick(this, args);
            }
        }
    }
}
```

2. Beispiel – Taktgeber mit Custom Event Args (3/4)

■ Listener

```
public class Listener
{
    public string Name { get; set; }
    public void Subscribe(Taktgeber m)
    {
        m.Tick += new Taktgeber.TickHandler(HeardIt);
    }
    private void HeardIt(Taktgeber m, EventArgs e)
    {
        System.Console.WriteLine("{0} : Takt Event erhalten um {1}", Name, e.EventTime);
    }
}
```

Registrierung

Event-handler

■ Custom Event Argumente

```
public class EventTimeArgs : EventArgs
{
    private DateTime eventTime;
    public DateTime EventTime
    {
        set { eventTime = value; }
        get { return this.eventTime; }
    }
}
```

Event-arguments

2. Beispiel – Taktgeber mit Custom Event Args (4/4)

■ Testprogramm

```
class Program
{
    static void Main(string[] args)
    {
        TestTaktgeber();
    }

    static void TestTaktgeber()
    {
        Taktgeber taktgeber = new Taktgeber() { Interval = 2000 };
        Listener listener = new Listener() { Name = "Listener 1" };
        listener.Subscribe(taktgeber);
        Listener listener2 = new Listener() { Name = "Listener 2" };
        listener2.Subscribe(taktgeber);
        taktgeber.Start();
    }
}
```

Output:

```
Listener 1 : Takt Event erhalten um 03.12.2013 21:03:38
Listener 2 : Takt Event erhalten um 03.12.2013 21:03:38
Listener 1 : Takt Event erhalten um 03.12.2013 21:03:40
Listener 2 : Takt Event erhalten um 03.12.2013 21:03:40
```

Übung 7.2



Events

- Baue das Taktgeber-Beispiel aus den Folien komplett nach