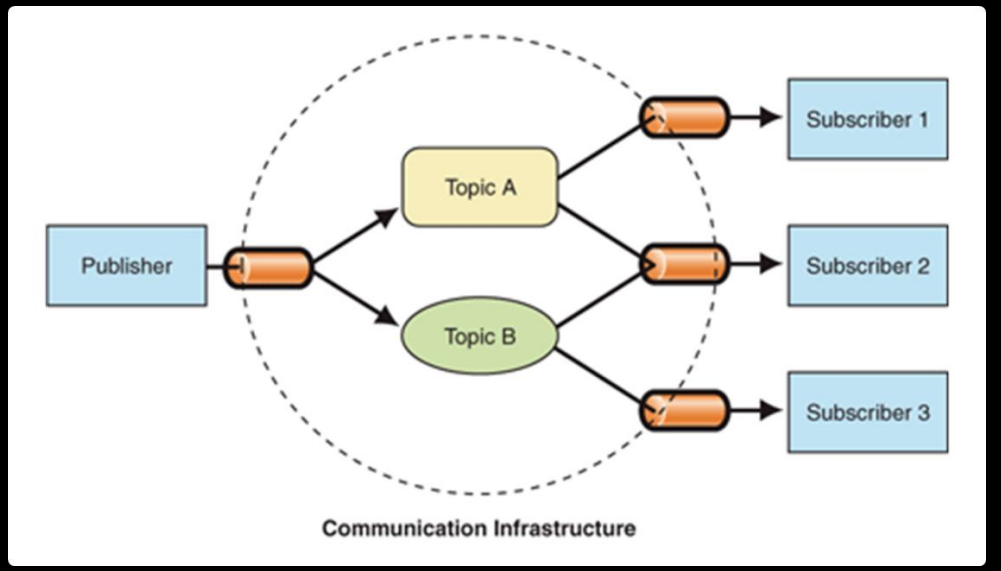
Въведение в събитийното програмиране

Учителски екип

Обучение за ИТ кариера

https://it-kariera.mon.bg/e-learning/



Съдържание

1. Събития
2. Дефиниране на събития
3. Известяване за събития
4. Цикъл на събитията
5. Шаблон Наблюдател

Какво са събитията?

- Механизъм за уведомление
  - Чрез тях един компонент известява други компоненти , че нещо се е случило

- Играят централна роля в .NET платформата и по-специално в графичните и уеб приложения

Как работят събитията?

- Обектът, който поражда дадено събитие, е негов подател
- Обектът, който го получава, се нарича негов получател
- За да получат едно събитие, получателите първо трябва да се "абонират за събитието"
- Обектите, създаващи събития не е необходимо изрично да знаят кои обекти ще обработват събитието
- Събитието се обработва от метод в обекта-получател
- При събитие се подават изпращач и EventArgs (данни за събитието)

Дефиниране на събитие

- За да може даден компонент да уведомява другите за събитие, то трябва да е дефинирано в неговия клас с ключовата дума **event**
- Например в класа Button можем да видим следното:

```
public event EventHandler DoubleClick;
```

```
public event MouseEventHandler DoubleMouseClick;
```

- Събитията в C# са инстанции на делегати
- Делегатите са специални типове в C#, които съдържат референция (указател) към метод като своя стойност
  - Описва сигнатурата на метода (параметрите и връщаната стойност)

Метод - получател на събитие

- Даден компонент може да получи и обработи събитие от друг компонент чрез свой метод
- Например при двойно щракване върху бутон се генерира метод, който ще служи за обработка на събитието по натискането му:

```
private void button1_Click(object sender, EventArgs e) { … }
```

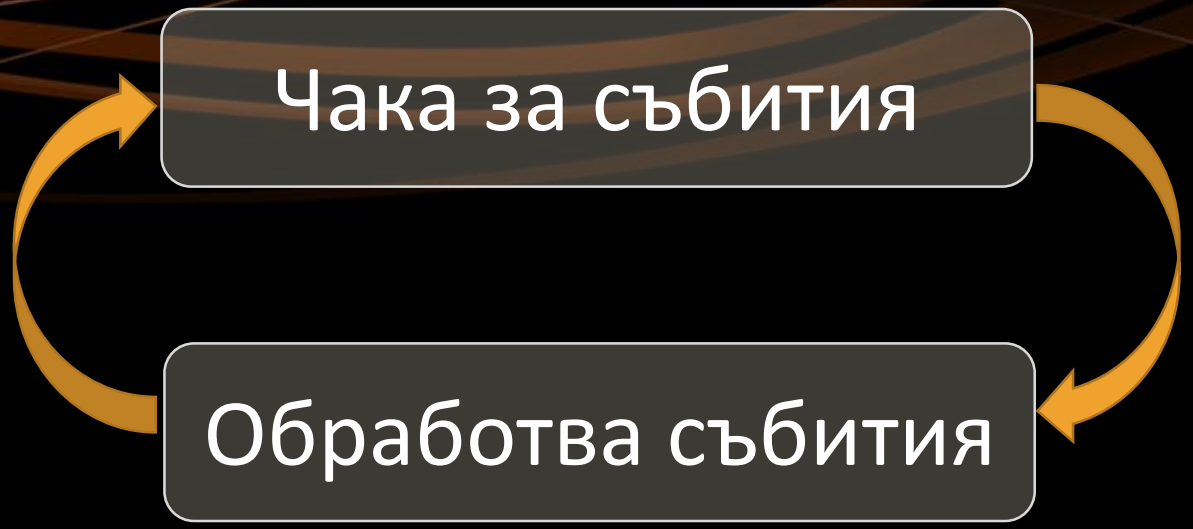
- А в метода InitializeComponent автоматично се добавя реда:

```
this.button1.Click += new System.EventHandler(this.button1_Click);
```

- Чрез него метода се абонира за получаване на събития от бутона
  - Отписването от абонамент за събитието става с -=

Цикъл на събитията

- Получава събития от ОС
- Извиква методи за обработката на някои от тях




```
while (message != "quit")
{
  // чака за събитие на ОС
  message = GetMessage();

  ProcessMessage(message);
}
```

Обработка на събитието Click на мишката UI – Пример

```
public partial class MainWindow : Window
{
  public MainWindow()
  {
    this.InitializeComponent();
    this.MouseDown += this.MainWindow_MouseClick;
  }
  private void MainWindow_MouseClick(object sender,
                                     MouseButtonEventArgs e)
  {
    MessageBox.Show(string.Format("Mouse clicked at ({0}, {1})",
                    e.MouseDevice.GetPosition(this).X,
                    e.MouseDevice.GetPosition(this).Y));
  }
}
```

Получава данни за щракване на мишката като `MouseButtonEventArgs`

"Наблюдател/слушател" шаблон в проектирането

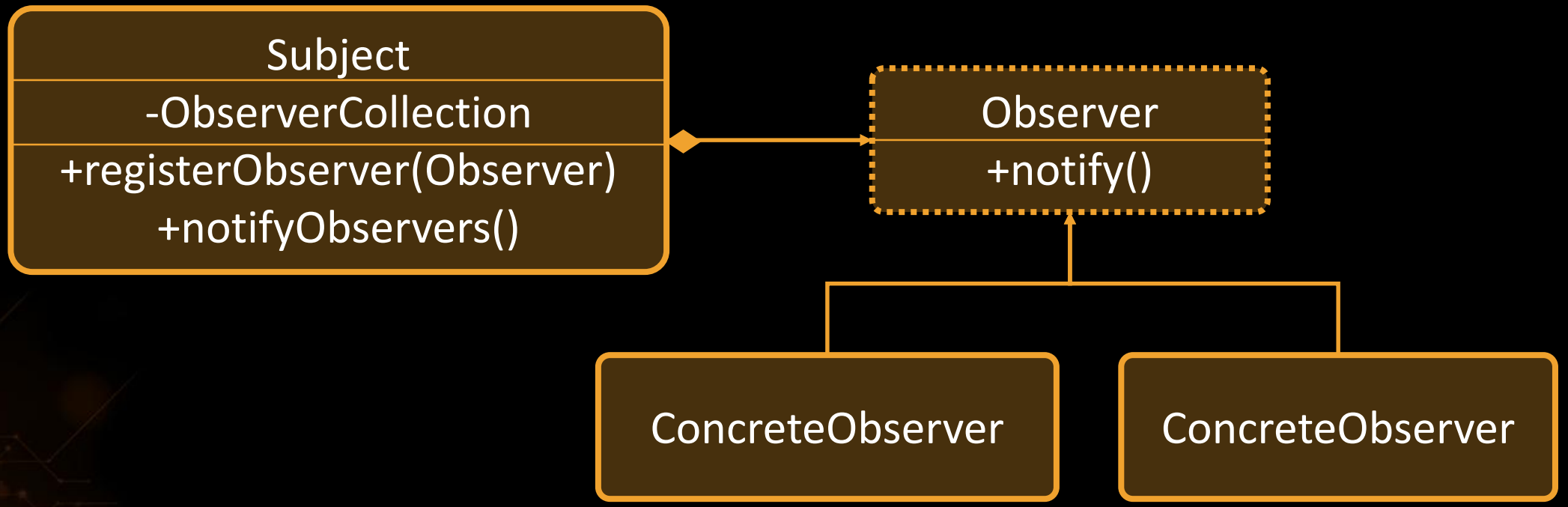
- Дефинира връзка **”едно към много” (one-to-many)**
- **Наблюдателите** се уведомяват при настъпване на събитие

```
Subject subject = new Subject();
subject.addObserver(new Observer());
subject.addObserver(new Observer());

subject.NotifyObservers();
```

Наблюдател – UML диаграма

- Субект, наблюдател
- ConcreteObserver



Задача: Наблюдател

- Създайте набор от интерфейси, с които един обект може да уведомява други за някакво събитие, случило се при него
- интерфейс **`ISubject`**
  - **`void Register(IObserver)`**
  - **`void Unregister(IObserver)`**
  - **`void NotifyObservers(int)`**
- интерфейс **`IObserver`**
  - **`Notify(int)`**

Решение: Наблюдател (интерфейси)

```
public interface Subject
{
  void Register(IObserver observer);
  void Unregister(IObserver observer);
  void NotifyObservers(int data);
}
```

```
public interface Observer
{
    void Notify(int data);
}
```

Решение : Наблюдател (реализация)

```
public void Register(IObserver observer) {
  this.observers.add(observer);
}

public void Unregister(IObserver observer) {
  this.observers.remove(observer);
}

public void NotifyObservers(int data) {
  foreach (IObserver observer in observers) {
    observer.Notify(data);
  }
}
```

Какво научихме?

- Събитията позволяват абонамент за известия относно нещо, случващо се в обект
- Получателите на събитието се абонират за него и го обработват, след като събитието настъпи
- Записването е с +=, а отписването с -=
  - Когато събитие „се случи", всички абонати получават уведомление за него
  - Получават и кой е изпращача и друга информация, чрез EventArgs

Въведение в събитийното програмиране

Въпроси?

Лиценз

- Настоящият курс (слайдове, примери, видео, задачи и др.) се разпространяват под свободен лиценз "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International"

- Благодарности: настоящият материал може да съдържа части от следните източници
  - Книга "Основи на програмирането със C#" от Светлин Наков и колектив с лиценз CC-BY-SA