

Событийно-ориентированное программирование

Юрий Литвинов
yurii.litvinov@gmail.com

02.03.2018г

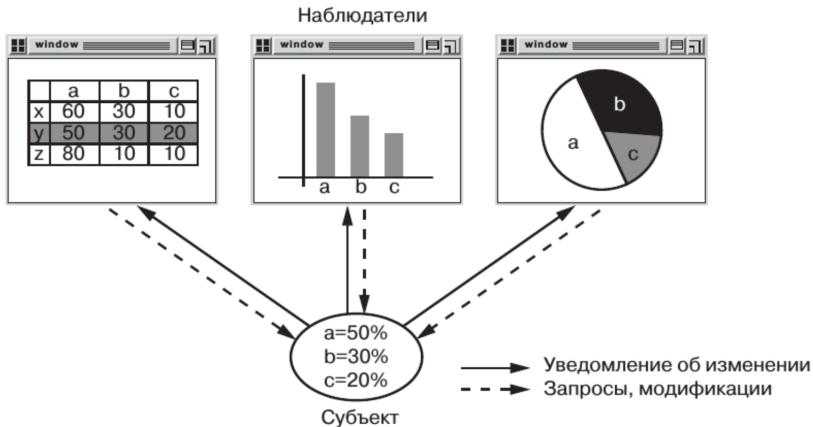
Событийно-ориентированное программирование: зачем?

- ▶ Большая часть приложений большую часть времени чего-то ждёт
- ▶ Активное ожидание потребляет системные ресурсы
- ▶ Приложения с пользовательским интерфейсом:
 - ▶ Цикл обработки событий
 - ▶ Очередь событий
 - ▶ Обработчики
- ▶ Сетевые и распределённые приложения
- ▶ Приложения, работающие с оборудованием

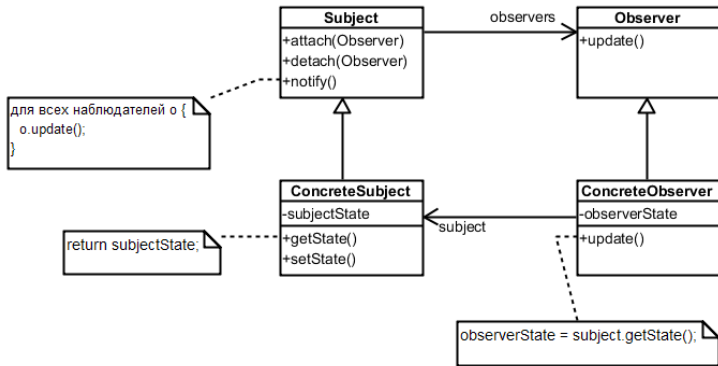
Реализация

- ▶ Callback-и (hooks)
- ▶ Переопределение виртуальных методов библиотечного класса
 - ▶ Inversion of Control — не мы вызываем библиотеку, а она нас
- ▶ Языковая поддержка — event-ы
- ▶ Паттерн “Наблюдатель”

Паттерн “Наблюдатель”



Наблюдатель, структура классов



Пример, кнопки в JavaFX

```
public class Buttons extends Application {  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
        Button button = new Button("Click me");  
  
        button.setOnAction(new EventHandler<ActionEvent>() {  
            @Override  
            public void handle(ActionEvent event) {  
                Platform.exit();  
            }  
        });  
  
        Scene scene = new Scene(button, 200, 100);  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
}
```

Языковая поддержка: event-ы в C#

- ▶ Ключевое слово **event**, декларирующее событие:
public event Action SomethingHappened;
- ▶ На событие можно подписаться:
ourObject.SomethingHappened += DoSomething;
- ▶ И отписаться:
ourObject.SomethingHappened -= DoSomething;
- ▶ А ещё событие можно инициировать:
SomethingHappened?.Invoke();

Пример, консольная игра

EventLoop

```
public class EventLoop
```

```
{
```

```
    public event Action OnLeft;
```

```
    public event Action OnRight;
```

```
    public void Run()
```

```
    {
```

```
        while (true)
```

```
        {
```

```
            var key = Console.ReadKey(true);
```

```
            switch (key.Key)
```

```
            {
```

```
                case ConsoleKey.LeftArrow:
```

```
                    OnLeft?.Invoke();
```

```
                    break;
```

```
                case ConsoleKey.RightArrow:
```

```
                    OnRight?.Invoke();
```

```
                    break;
```

```
            }
```

```
        }
```

```
    }
```

```
}
```


Пример, консольная игра

Game

```
public class Game
{
    public void OnLeft()
        => Console.CursorLeft--;

    public void OnRight()
        => Console.CursorLeft++;
}
```

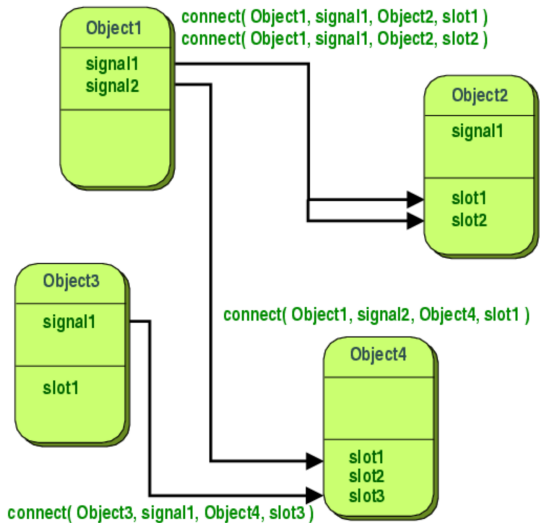
Пример, консольная игра

Main

```
public class Program
{
    public static void Main(string[] args)
    {
        var eventLoop = new EventLoop();
        var game = new Game();

        eventLoop.OnLeft += game.OnLeft;
        eventLoop.OnRight += game.OnRight;
        eventLoop.Run();
    }
}
```

C++/Qt — сигналы и слоты



Пример

```
#include <QObject>
```

```
class Counter : public QObject
```

```
{  
    Q_OBJECT
```

```
public:
```

```
    Counter() { m_value = 0; }
```

```
    int value() const { return m_value; }
```

```
public slots:
```

```
    void setValue(int value);
```

```
signals:
```

```
    void valueChanged(int newValue);
```

```
private:
```

```
    int m_value;
```

```
};
```

Пример событийной архитектуры — ROS

