Делегаты, события

Мухортова Н.Н.

Цель

Изучить инструменты ООП делегаты и события. Научиться использовать их при проектировании приложения

Определение делегатов

Делегаты - это указатели на методы

```
delegate int Operation(int x, int y);
```

```
delegate void GetMessage();
```

Методы, на которые ссылаются делегаты, должны иметь те же параметры и тот же тип возвращаемого значения

Определение делегатов

```
delegate void GetMessage(); // 1. Объявляем делегат

 static void Main(string[] args)   {

    GetMessage del; // 2. Создаем переменную делегата

    if (DateTime.Now.Hour < 12)      {

       del = GoodMorning; // 3. Присваиваем этой переменной адрес метода

    }      else      {

       del = GoodEvening;      }

    del.Invoke(); // 4. Вызываем метод

    Console.ReadLine();   }
```

Определение делегатов

```
private static void GoodMorning()   {
      Console.WriteLine("Good Morning");
   }
private static void GoodEvening()   {
      Console.WriteLine("Good Evening");
   }
```

Определение делегатов

Присваивание делегату адреса метода через конструктор

```
delegate int Operation(int x, int y);

private static int Add(int x, int y)  {

    return x+y;

}

private static int Multiply (int x, int y)  {

    return x * y;

}
```

Определение делегатов

```
static void Main(string[] args)   {

    // присваивание адреса метода через контруктор

    Operation del = new Operation(Add); // делегат указывает на метод Add

    int result = del.Invoke(4,5);

    Console.WriteLine(result);

    del = Multiply; // теперь делегат указывает на метод Multiply

    result = del.Invoke(4, 5);

    Console.WriteLine(result);
```

Делегаты как параметры методов

```
delegate void GetMessage();
private static void Show_Message(GetMessage  _del)   {
    _del.Invoke();
  }
  private static void GoodMorning()   {
    Console.WriteLine("Good Morning");
  }
  private static void GoodEvening()   {
    Console.WriteLine("Good Evening");   }
```

Делегаты как параметры методов

```
static void Main(string[] args)   {
      if (DateTime.Now.Hour < 12)      {
         Show_Message(GoodMorning);
      }      else      {
          Show_Message(GoodEvening);
      }
      Console.ReadLine();
   }
```

Применение делегатов

Делегаты позволяют создать функционал методов обратного вызова, уведомляя другие объекты о произошедших событиях

Пример в отдельном файле

Применение делегатов

```
class Account{
    public delegate void AccountStateHandler(string message);    // Объявляем делегат
    AccountStateHandler _del;  // Создаем переменную делегата
  // Регистрируем делегат
  public void RegisterHandler(AccountStateHandler del)   {
     _del = del;
  }
  // Далее остальные строки класса Account
```

Применение делегатов

```
public void Withdraw(int sum){
  if (sum <= _sum)   {
     _sum -= sum;
    if (_del != null)
       _del($"Сумма {sum} снята со счета");
  }   else   {
    if (_del != null)
       _del("Недостаточно денег на счете");
  }}
```

Применение делегатов

```
static void Main(string[] args)  {
    Account account = new Account(200); // создаем банковский счет
    // Добавляем в делегат ссылку на метод Show_Message
    // а сам делегат передается в качестве параметра метода RegisterHandler
    account.RegisterHandler(new Account.AccountStateHandler(Show_Message));
    account.Withdraw(100); // Пытаемся снять деньги
    account.Withdraw(150); // Пытаемся снять деньги
    Console.ReadLine();
}
```

Применение делегатов

```
private static void Show_Message(String message)   {

   Console.WriteLine(message);

}
```

События

События объявляются в классе с помощью ключевого слова event, после которого идет название делегата

// Объявляем делегат

public delegate void AccountStateHandler(string message);

// Событие, возникающее при выводе денег

public event AccountStateHandler Withdrawn;

Пример в файле

Обработчик события

```
class AccountEventArgs{
    // Сообщение
    public string Message{get;}
    // Сумма, на которую изменился счет
    public int Sum {get;}
    public AccountEventArgs(string mes, int sum)  {
        Message = mes;
        Sum = sum;  }
}
```

Выводы

В C# делегаты образуют основные строительные блоки для событий.

Делегат — это тип, который определяет сигнатуру метода. В C# вы можете создать экземпляр делегата, указывающий на другой метод.

И те и другие обеспечивают сценарии позднего связывания, в которых взаимодействие компонента осуществляется путем вызова метода, известного только во время выполнения. И те и другие поддерживают методы с одним или несколькими подписчиками. Иногда это называют поддержкой одноадресности и многоадресности.