Модуль 12: Свойства и индексаторы

Обзор

- Использование свойств
- Создание и использование индексаторов

Использование свойств

- Зачем использовать свойства?
- Использование аксессоров
- Сравнение свойств с полями
- Сравнение свойств с методами
- Типы свойств

Зачем использовать свойства?

- **Преимущества использования свойств:**
  - Удобный способ инкапсуляции информации внутри класса
  - Прозрачный синтаксис

  Пример: o.SetValue(o.GetValue()+1);

  o.Value++;

Использование аксессоров

- **Можно работать со свойством как с открытой переменной-членом класса**
  - **get** –аксессор предназначен для чтения свойств
  - **set** – аксессор предназначен для установки свойства

```
class Button
{
    public string Caption // Property
    {
        get { return caption; }
        set { caption = value; }
    }
    private string caption; // Field
}
```

Типы свойств

- **Для чтения и записи**
  - Реализуют **get** - и **set** -аксессоры
- **Только для чтения**
  - Реализован только **get** –аксессор
  - Не константы
- **Только для записи**
  - Реализован только **set** –аксессор
- **Статические свойства**
  - Для обращения к статическим данным, хранящим информацию на уровне всего класса

Сравнение свойств с полями

- **Свойства – это «умные поля»**
  - **get** –аксессор может возвращать расчетное значение
- **Сходства**
  - Одинаковый синтаксис создания и использования
- **Различия**
  - Свойства не определяют область памяти
  - Свойства нельзя передавать в методы как **ref** или **out**

Сравнение свойств с методами

- **Сходства**
  - И те, и другие содержат исполняемый код
  - И те, и другие можно использовать для инкапсуляции данных
  - И те, и другие могут быть virtual, abstract или override
- **Различия**
  - Синтаксические – для работы со свойствами не используются круглые скобки
  - Семантические – свойства не могут быть **void** или принимать параметры

Создание и использование индексаторов

- Что такое индексатор?
- Сравнение индексаторов с массивами
- Сравнение индексаторов со свойствами
- Использование параметров при определении индексаторов
- Пример

Что такое индексатор?

- **Индексатор позволяет получать доступ к объекту по индексу подобно тому, как это реализовано в массивах**
  - Удобно, если свойство может принимать различные значения
- **Создание индексатора**
  - Создайте свойство с именем *this*
  - Определите тип индекса
- **Использование индексатора**
  - Для чтения или записи проиндексированного свойства используйте синтаксис для массивов

Создание одномерных индексаторов

```
тип_элемента this[int индекс]
{
// Аксессор считывания данных,
get {
// Возврат значения, заданного  элементом индекс

// Аксессор установки данных,
set {
// Установка значения, заданного элементом индекс
      }

}
```

- *тип_элемента* — базовый тип индексатора
- Параметр *индекс* получает индекс опрашиваемого (или устанавливаемого) элемента

Использование параметров при определении индексаторов

- **При создании индексатора**
  - Необходимо определить хотя бы один индекс
  - Укажите значение для каждого из параметров
  - Не используйте модификаторы **ref** или **out**

```
class MultipleParameters
{
public string this[int one, int two]
{
get { ... }
set { ... }
}
...
}
...
MultipleParameters mp = new MultipleParameters( );
string s = mp[2,3];
...
```

Сравнение индексаторов с массивами

- **Сходства**
  - И те, и другие используют синтаксис для массивов
- **Различия**
  - Индексаторы могут использовать индексы различных типов
  - Индексаторы можно перегружать – можно создать несколько индексаторов с индексами различного типа
  - Индексаторы – это не переменные, они не определяют область памяти. Их нельзя передавать в качестве параметров, используя **ref** или **out**

Сравнение индексаторов со свойствами

- **Сходства**
  - И те, и другие используют **get**- и **set** -аксессоры
  - Ни те, ни другие не определяют область памяти
  - Ни те, ни другие не могут быть void
- **Различия**
  - Индексаторы можно перегружать
  - Индексаторы не могут быть статическими

Лабораторная работа 12: Использование свойств и индексаторов