Практическая работа 25
Реализация свойств и индексаторов

Цель занятия: Получить практические навыки реализации свойств и индексаторов

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio)
Microsoft Visual Studio

Краткие теоретические сведения

1 Свойства

Свойство — это член, предоставляющий гибкий механизм для чтения, записи или вычисления значения частного (private) поля. Свойства можно использовать, как если бы они являлись открытыми членами данных, хотя в действительности они являются специальными методами, называемыми методами доступа. Это обеспечивает простой доступ к данным и позволяет повысить уровень безопасности и гибкости методов.

В примере, класс TimePeriod хранит сведения о периоде времени. Внутри класса время хранится в секундах, но свойство с именем Hours позволяет клиенту задать время в часах. Методы доступа для свойства Hours выполняют преобразование между часами и секундами.

```
Class TimePeriod
{
private double seconds;

public double Hours
   {
get { return seconds / 3600; }
set { seconds = value * 3600; }
   }
}
class Program
{
static void Main()
   {
TimePeriod  t = new TimePeriod();
t.Hours = 24;
System.Console.WriteLine("Time in hours: " + t.Hours);
}
```

```
}
```

2 Общие сведения о свойствах

Свойства позволяют классу предоставлять общий способ получения и задания значений, скрывая при этом код реализации или проверки.

Метод доступа свойства get используется для возврата значения свойства и должно содержать ключевое слово return, а метод доступа set используется для назначения нового значения. Эти методы доступа могут иметь различные уровни доступа.

Ключевое слово value используется для определения значения, присваиваемого методом доступа set.

Свойства, которые не реализуют метод доступа set, доступны только для чтения.

Для простых свойств, не требующих пользовательского кода метода доступа, рассмотрите возможность использования автоматически реализуемых свойств.

2.1.Метод доступа get

Основная часть метода доступа get похожа на основную часть метода. Она должна возвращать значение типа свойства.

```
class Person
{
private string name;  // the name field
public string Name    // the Name property
{
get
      {
return name;
      }
   }
}
```

При создании ссылки на свойство, кроме случая присвоения ему значения, для чтения значения свойства вызывается метод доступа get. Например:

```
Person person = new Person();
//...
System.Console.Write(person.Name);  //
```

Метод доступа get должен заканчиваться оператором return.

Если в предыдущем фрагменте кода свойству Name не назначается какое-либо значение, это свойство возвращает значение "NA".

2.1.Метод доступа set

Метод доступа set похож на метод, имеющий тип возвращаемого значения void. В нем используется неявный параметр value, тип которого соответствует типу свойства. В следующем примере метод доступа set добавляется в свойство Name:

```
class Person
{
private string name;  // the name field
public string Name    // the Name property
   {
get
      {
return name;
      }
set
      {
name = value;
}
}
}
```

Когда свойству присваивается значение, выполняется вызов метода доступа set с помощью аргумента, предоставляющего новое значение. Например:

```
Person person = new Person();
person.Name = "Joe";
System.Console.Write(person.Name);  // the get accessor is invoked here
```

Использование имени неявного параметра value для объявления локальной переменной в методе доступа set является ошибкой.

3 Индексаторы

Индексаторы в C# обеспечивают естественный синтаксис для доступа к элементам в классах или структурах, которые инкапсулируют список или словарь значений. Индексаторы являются подобными свойствам, но доступ к ним происходит с помощью аргумента индекса, а не имени свойства.

Класс string имеет индексатор, который дает вам возможность получить доступ к каждому из символов, который содержится в строке с помощью индекса типа int:

```
string s = "hello";
Console.WriteLine (s[0]); // 'h'
Console.WriteLine (s[3]); // 'l'
```

Синтаксис использования индексаторов похож на использование массивов, за исключением того, что аргумент индекса может быть любого типа.

Индексаторы имеют те же модификаторы, которые доступны и для свойств.

3.1 Реализация индексаторов в C#

Пример. Создадим класс Book, представляющий книгу, и класс Library, который представляет библиотеку и используется для хранения набора книг. Используем индексаторы для определения класса Library:

```
Class Book
{
publicBook(string name)
{
this.Name=name;
    }
public string Name { get; set; }
}

class Library
{
Book[] books;

public Library()
    {
books = new Book[] { new Book("Отцыидети"), new Book("Войнаимир"), new Book("ЕвгенийОнегин") };
    }

Public int Length
    {
get { return books.Length; }
    }

public Book this[int index]
    {
get
        {
return books[index];
        }
set
        {
books[index] = value;
        }
```

```
    }
}
```

В классе Library определен массив Book, к которому можно обращаться с помощью индексатора. Определение индексатора напоминает свойство: public Book this [int index]. Определяем тип возвращаемого или присваиваемого объекта, то есть Book и через параметр int index способ доступа к элементам.

```
class Program
{
static void Main(string[] args)
{
      Library library = new Library();

Console.WriteLine(library[0].Name);

library[0] = newBook("Преступление и наказание");

Console.WriteLine(library[0].Name);

Console.ReadLine();
}
}
```

В основной программе можно обращаться к объекту Library как к массиву: library[0].Name. Так как индексатор использует тип Book, то library[0] будет представлять объект Book.

Задания

1 Изучить теоретические сведения и задание к работе

2 Реализовать классы в соответствии с вариантом задания, в котором данные описаны в качестве массива или поля класса, которые не являются массивами. В классе определить индексатор. Создать объект на основе созданного класса. Осуществить использование объекта через индексатор.

3 Составить и отладить программу.

Порядок выполнения работы

…

Содержание отчета

1 Название работы
2 Цель работы
3 Технические средства обучения
4 Задания (условия задач)
5 Порядок выполнения работы
6 Листинг программы
7 Вывод

Варианты заданий

1. Реализовать класс STUDENT для хранения информации об успеваемости студента за семестр.
2. Реализовать класс ZARPLATA для хранения информации о заработной плате сотрудника за год.
3. Реализовать класс SPORTSMEN для хранения информации о физических данных спортсмена.
4. Реализовать класс AVTO для хранения информации о параметрах автомобиля.
5. Реализовать класс CREDIT для хранения информации по предоставляемому кредиту.
6. Реализовать класс PAZIENT для хранения информации о данных пациента.
7. Реализовать класс CURS для хранения информации о курсе валют за неделю.
8. Реализовать класс TOVAR для хранения информации о товаре в магазине.
9. Реализовать класс BUDGET для хранения информации о расходах за неделю.
10. Реализовать класс ZVONOK для хранения информации о звонках на перемену.
11. Реализовать класс DOM для хранения информации о параметрах продаваемого объекта.
12. Реализовать класс SEANS для хранения информации о начале сеансов фильма в кинотеатре.
13. Реализовать класс REZULTAT для хранения информации о количестве оценок разного типа на экзамене.
14. Реализовать класс PREPOD для хранения информации о данных преподавателя.
15. Реализовать класс PAZIENT для хранения информации о данных пациента.
16. Реализовать класс TAXI для хранения информации о данных водителей и автомобилей таксопарка.

17. Реализовать класс АРТЕКА для хранения информации о данных лекарств.

18. Реализовать класс TURNIR для хранения информации о соревнованиях турнира.

19. Реализовать класс SKLAD для хранения информации о данных товаров.

20. Реализовать класс MOTO для хранения информации о параметрах мотоцикла.

21. Реализовать класс FLORA для хранения информации о данных растений.

22. Реализовать класс ZOO для хранения информации о данных животных.

23. Реализовать класс HOTEL для хранения информации о данных номеров.

Контрольные вопросы

1. Что такое свойства класса и для чего используются?
2. Общая форма объявления свойства в классе.
3. Преимущества использования свойств в классах.
4. Что такое аксессор?
5. Как влияют модификаторы доступа к элементам класса (private, public) на доступ к свойству?
6. Что называется индексатором?
7. Каким образом индексаторы применяются в программах?
8. Общая форма объявления индексатора.

Используемая литература

1 Гниденко, И. Г. Технология разработки программного обеспечения: учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017.

2 Шарп Джон Ш26 Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017.

3 Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Москва: Эксмо, 2018.

4 Васильев А.Н. Программирование на C# для начинающих. Особенности языка. – Москва: Эксмо, 2019.

5 http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.