# Практика 15. Свойства и индексаторы

1.Из класса банковский счет удалить методы, возвращающие значения полей номер счета и тип счета, заменить эти методы на свойства только для чтения. Добавить свойство для чтения и записи типа string для отображения поля держатель банковского счета.

Изменить класс BankTransaction, созданный для хранений финансовых операций со счетом, - заменить методы доступа к данным на свойства для чтения.

2.Добавить индексатор в класс банковский счет для получения доступа к любому объекту BankTransaction.

3. В классе здания из практики 9 № 4 все методы для заполнения и получения значений полей заменить на свойства. Написать тестовый пример.

4.Создать класс для нескольких зданий. Поле класса – массив на 10 зданий. В классе создать индексатор, возвращающий здание по его номеру.

## Свойства

Свойства – это наборы функций, которые могут быть доступны клиенту таким же способом, как открытые поля класса. Идея свойства заключается в методе или паре методов, которые ведут себя с точки зрения клиентского кода как поле. Чтобы определить свойство в C#, используется следующий синтаксис:

public string MyProperty

{

get

{

}

set

{

}

}

return MyField; //возвращает значение поля MyField

MyField=value; //задаем значение поля MyField

Средство доступа **get** не принимает никаких параметров и должно возвращать значение того типа, который объявлен для свойства. Для средства **set** в явном виде не передается никаких параметров, но компилятор предполагает, что оно принимает один параметр со значением value, относящийся к типу, указанному для свойства.

Внутри метода get и set можно написать дополнительный код, осуществляющий какую-либо обработку, допустим внутри set поместить код для проверки корректности введенных данных.

Свойство описывается как член класса, предоставляет доступ к полям класса. В соответствии с идеологией ООП имеет смысл поля делать private, а свойства более доступными: public, protected, internal (в зависимости от целей). Также рекомендуется, имена полей начинать с маленькой буквы, а имя свойства, предоставляющего доступ к полю, задавать таким же, как и у поля, но начинать с заглавной буквы.

Существует возможность создать свойство, доступное только для чтения. Для этого надо исключить set из определения свойства. Чтобы определить свойство, доступное только для записи, необходимо из его определения исключить get. Кроме того, для set и get можно применять различные модификаторы доступа, но надо помнить, что модификатор доступа одного из средств get или set должен совпадать с модификатором доступа всего свойства.

## Индексаторы

Индексация позволяет индексировать объекты таким же способом, как массив или коллекцию, позволяет обращаться к членам класса как к массиву.

Пусть объект состоит из нескольких подэлементов (например, listbox состоит из нескольких строк). Индексация позволяет обращаться к подэлементам, как в массивах.

class StringList

{

private string[] list;

public string this[int index]

{

get{ return list[index];} set{ list[index] = value;}

}

}

Индексатор – это свойство с именем this, квадратными скобками и индексом в них. Для использования индексатора, обращаемся к объекту:

StringList myList = new StringList();

myList[3] = “ok”; // индексатор записывает string myString = myList[3]; // индексатор читает

Индексаторы могут быть статичными элементами класса и их можно перегружать для различных типов индексов. Индексаторы не хранят значения и поэтому не могут быть переданы как ref и out параметры.

При объявлении индексаторов необходимо определить, по крайней мере, один параметр, определить значения для всех параметров. Можно определить несколько параметров для одного индексатора:

class MultipleParameters

{

public string this[int one, int two]

{

get{…}

set{…}

}

}

…

MultipleParameters mp = new MultipleParameters(); string s = mp[2,3];

…