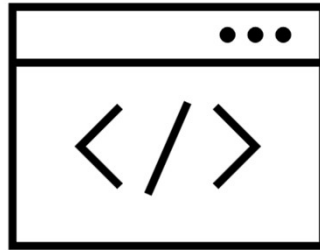


ООП

Инкапсуляция

Наследование

Полиморфизм



Классы. Свойства



КубГТУ, Кафедра «Информационных систем и программирования»

Как уже отмечалось ранее, данные по умолчанию считаются закрытыми, т.е. для них установлен спецификатор `private`.

Для того чтобы определить для пользователя способ доступа к полям класса, можно использовать такой функциональный элемент класса как **свойство**.

Свойство позволяет сделать поле доступным только для чтения или только для записи (при этом можно проверить допустимость присваиваемых полю значений).



Синтаксис свойства

```
[атрибуты] [спецификаторы] тип имя_свойства  
{  
  [get { код_доступа_для_чтения} ]  
  [set { код_доступа_для_записи} ]  
}
```

Код доступа представляет собой блоки операторов, которые выполняются при получении (**get**) или при установке (**set**) свойства. Может отсутствовать либо часть get, либо set, но не обе одновременно.

Блок get должен содержать оператор return, возвращающий выражение, для типа которого должно существовать неявное преобразование к типу свойства.



Обратится к свойству для получения значения можно, например, следующим образом:

```
x = имя_класса.имя_свойства
```

Обратится к свойству для установки значения можно, например, следующим образом:

```
имя_класса.имя_свойства = значение
```



Внесем следующие изменения в класс Circle:



1. Сделаем поля `x`, `y`, `radius`, `name` доступными только для чтения;
2. Создадим свойства для работы с закрытыми полями `x`, `y`, `radius`, позволяющие получать и устанавливать значения соответствующих полей; для поля `radius` в блоке `set` будем проводить проверку допустимости устанавливаемого значения;
3. Создадим свойство для работы с полем `name`, доступным только для чтения;
4. В конструкторе 1 инициализацию поля `radius` будем проводить через соответствующее свойство для проверки допустимости устанавливаемого значения;
5. Создадим свойство, доступное только для чтения, позволяющее определять площадь круга.



```
namespace MyProgram
```

```
{
```

```
class Circle
```

```
{
```

```
private int x;  
private int y;  
private int radius;  
static readonly string name="Окружность";
```

```
public Circle(int radius)
```

```
{
```

```
Radius= radius;
```

```
}
```

```
public Circle (int x, int y, int radius) :this (radius)
```

```
{
```

```
this.x = x;
```

```
this.y = y;
```

```
}
```

```
public void Show()
```

```
{
```

```
Console.WriteLine("{0} с центром в точке  
({1},{2}), радиусом {3}", name, x, y, radius);
```

```
}
```

//теперь поля закрыты и доступ из
другого класса невозможен

//устанавливаем поле radius через
одноименное свойство, для проверки
допустимости значения



КубГТУ, Кафедра «Информационных систем и программирования»

```
public int X
{
    get
    {
        return x;
    }
    set
    {
        x = value;
    }
}
```

//свойство для обращения к
закрытому полю x

```
public int Y
{
    get
    {
        return y;
    }
    set
    {
        y = value;
    }
}
```

//свойство для обращения к полю y




```
public int Radius
{
    get
    {
        return radius;
    }
    set
    {
        if (value>0)
        {
            radius = value;
        }
        else
        {
            radius=0;
            Console.WriteLine("Недопустимое
            значение радиуса {0}", value);
        }
    }
}
```

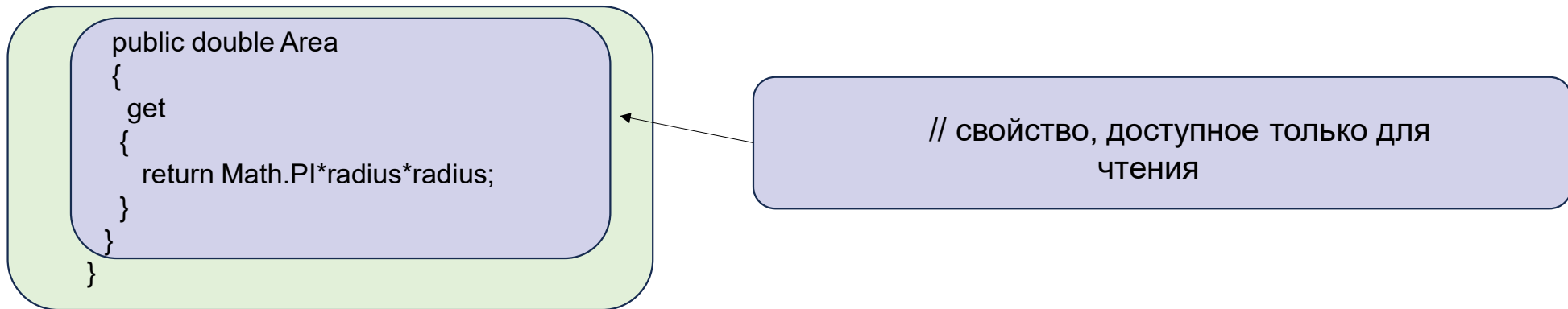
//свойство для обращения к полю
radius

```
public string Name
{
    get
    {
        return name;
    }
}
```

//свойство для работы с закрытым readonly
полем



КубГТУ, Кафедра «Информационных систем и программирования»



Таким образом, свойство **Area** не пытается получить или установить значение какого-либо закрытого поля класса, оно решает самостоятельную задачу – вычисляет площадь круга.

Поэтому свойство можно использовать как разновидность метода. Данный пример при обнаружении ошибочных данных выводит сообщение напрямую на консоль.



```
static void Main()
```

```
{
```

```
    Circle oneCircle = new Circle(-1);
```

```
    oneCircle.Show();
```

```
    Console.WriteLine();
```

```
    oneCircle.X=-1;
```

```
    oneCircle.Y=1;
```

```
    oneCircle.Radius=2;
```

```
    oneCircle.Show();
```

```
    Console.WriteLine("Заданная {0} имеет  
    площадь {1:f}", oneCircle.Name,  
    oneCircle.Area);
```

```
}
```

// параметр value принимает значение -1

// параметр value принимает значение 1

// параметр value принимает значение 2



Результат работы программы:

Недопустимое значение для радиуса окружности -1

Окружность с центром в точке $(0, 0)$ радиусом 0

Окружность с центром в точке $(-1, 1)$ радиусом 2

Заданная Окружность имеет площадь 12.57

