Тема 7
Свойства на класовете

1. Концепция за използване на свойства на класа

Един от принципите на обектно ориентираното програмиране е капсулирането на данните в тялото на класа, чрез което те остават скрити и достъпни единствено за собствените методи на класа. Така изменението на вътрешното представяне на данните ще се отрази само на програмния код в класа, без да се налагат каквито и да е изменения в останалия код. Това се постига като се използват модификатори на достъп ***private*** или в някои случаи - ***protected***. Съществуват обаче не малко ситуации, при който е небходимо да се достъпи дадено поле от метод, който не принадлежи на класа.

В C# съществуват два начина за достъп до полетата на даден клас – чрез използване на модификатор за достъп ***public*** или посредством свойства. В първия случай, полето се обявява с модификатор public и съответно, до него имат достъп всички методи, за които класът е достъпен. Например:

```
class Time
{
            public int hour, minute, second;

}
```

В случая, всеки метод има достъп до полетата на клас **`Time`**. Затова при дефиниране на обект **`currentTime`** в метод Main, полетата на обекта са достъпни.

```
class Program
{
      static void Main()
      {
            Time currentTime = new Time();
            currentTime.hour = 10;
            currentTime.minute = 30;
            currentTime.second = 0;

      }
}
```

Достъпът до полетата е възможен, но обявяването на поле като ***public*** противоречи на принципа на капсулиране на данните. Това е причината поради която за достъп до поле на клас се използва свойство.

Свойствата са членове на класа, които не отговарят директно на място за съхраняване на информация, а играят посредническа роля за достъп до полета на класа, който са със статут ***private***. Самото свойство се дефинира с модификатор за достъп ***public***. Достъпът до данните е посредством т.нар. елементи за достъп или аксесоари (accessors).

Елементите за достъп определят операторите и изразите, които се изпълняват при четене или запис на стойност в свойство. Елемент за достът, предназначен за четене на стойност на свойство се обозначава с ключова дума **get**. Елемент, предназначен за модифициране на стойност на свойството се обозначава с ключова дума **set**. Този елемент на достъп предава новата

стойност посредством параметър, определен с ключовата дума **value**. Наличието на get и set методите определя дали свойството е за четене или за запис. Възможни са три варианта:

- Ако в тялото на свойството има само метод **get**, свойството е само за четене.
- Ако в тялото на свойството има само метод **set**, свойството е само за запис.
- Ако в тялото на свойството има както метод **get**, така и метод **set**, свойството е за четене и запис.

2. Дефиниране на свойство

Общият вид на дефиницията на свойство е следният:

```
тип ИмеСвойство
{
        get
         { тяло на get-метод }

        set
         { тяло на set-метод }
}
```

Подобно на полетата и методите, свойствата също имат тип. Той определя какъв е резултатът, който ще бъде върнат от метод get и каква е стойността, която може да получи метод set. След името на свойството не се задават параметри, но свойството има тяло, в което може да има методи за достъп get и set.

Пример:

```
class Time
{
            private int hour, minute, second;

            public int Hour
            {
                  get { return hour;}
                  set { hour = value;}

            }

}
```

В случая, полето **`hour`** на класа е свързано със свойството **`Hour`**. Често срещана практика е имената на свойствата да съвпадат с имената на полетата, като името на свойството е с главна буква, което го отличава от името на полето. Това е улеснение за програмистите, за да е ясно кое поле с кое свойство е свързано. Свойството **`Hour`** е за четене и запис и осъществява връзка с поле **`hour`** . Аксесоар за достъп get връща стойността на полето, а set задава стойност на полето. Ключова дума value се свързва със зададената стойност. Достъпът до

поле **hour** посредством това свойство може да се илюстрира чрез следния програмен код:

```
class Program
{
      static void Main()
      {
            Time currentTime = new Time();
            currentTime.Hour = 10;


      }
}
```

Тъй като полетата **minute** и **second** не са свързани със свойства, няма как стойностите за минута или секунда на текущото време да бъдат променени или достъпени в метод Main.

Не е задължително свойство на класа да бъде свързано с конкретно поле. В методите get и set може да има програмен код, както във всеки един метод. Пример:

```
class Circle
{
        private double radius;

        public Circle(double r)
        {
            radius=r;
        }

        public double Area
        {
            get
            {
                  return radius*radius*Math.PI;
            }
        }

       public double Radius
        {
            get
            {
                  return radius;
            }
            set
            {
                  radius = value;
            }
        }

}
```

В класа `Circle` е дефинирано свойство **`Area`**, което е само за четене и връща като резултат площта на кръга. Освен това, в класа е дефинирано и свойство `Radius`, което е както за четене, така и за запис и служи за достъп и промяна на стойността на полето `radius`. Класът има само един конструктор, който изисква един параметър от тип double. Следващият примерен код илюстрира работата със свойствата на клас `Circle` в метод Main:

```
class Program
{
      static void Main()
      {
            Circle cir = new Circle(4);
            Console.WriteLine("A circle with a radius {0}.", cir.Radius);
            Console.WriteLine("The area is {0}.", cir.Area);
            Console.Write("Enter a value for the radius:");
            cir.Radius = Convert.ToDouble(Console.ReadLine());
            Console.WriteLine("A circle with a radius {0}.", cir.Radius);
            Console.WriteLine("The area is {0}.", cir.Area);

      }
}
```

3. Алтернативен достъп до поле чрез използвне на методи

Освен чрез използване на свойства, достъп до поле с модификатор private може да бъде направен и чрез методи. По подобие на аксесоарите за достъп get и set, се формират двойка методи Get и Set, които четат и променят стойност на дадено поле. Метод Get няма параметър и връща като резултат стойността на полето, а метод Set е от тип void и има един параметър, който задава стойност на полето. Примерната реализация на програмния код на клас `Circle` чрез използване на методи вместо свойства е следният:

```
class Circle
{
        private double radius;

        public Circle(double r)
        {
            radius=r;
        }

        public double Area()
        {
            return radius*radius*Math.PI;
        }

        public double GetRadius()
        {

            return radius;
        }

        public void SetRadius(double r)
        {
            radius = r;
        }

}
```

В случая, методи `GetRadius` и `SetRadius` заменят свойството `Radius`. Реализацията на намирането на площта чрез метод `Area` е позната от предишните теми. Измененията в програмния код на метод Main са следните:

```
            Circle cir = new Circle(4);
            Console.WriteLine("A circle with a radius {0}.",
```

```
cir.GetRadius());
            Console.WriteLine("The area is {0}.", cir.Area());
            Console.Write("Enter a value for the radius:");
            cir.SetRadius(Convert.ToDouble(Console.ReadLine()));
            Console.WriteLine("A circle with a radius {0}.",
      cir.GetRadius());
            Console.WriteLine("The area is {0}.", cir.Area());
```

Алтернативният достъп до поле посредством методи се използва основно в програмни езици, който не поддържат концепцията за свойства на класа. Въпреки че език C# поддържа свойства, достъп до поле може да се направи и чрез двойка методи Get и Set.

Относно въпроси по темата на адрес: ln_zh_st@yahoo.com