РЕФЕРЕНЦА THIS

Кључна реч "this" користи се када име промењиве у методи подудара са именом поља класе. Дакле, када је исто име атрибута и локалне промењиве.

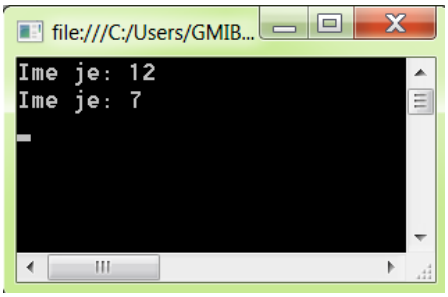
```
using System;
    class Program
    {
        private int d = 7;

        public void Print() {

            int d = 12;

            Console.WriteLine("Ime je: {0}", d);
            Console.WriteLine("Ime je: {0}", this.d);
        }

        static void Main(string[] args)
        {
            Program p = new Program();
            p.Print();
            Console.ReadKey();
        }
    }
```



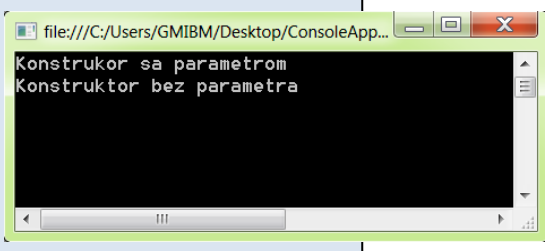
Из наведеног примера можемо да закључимо да се референца **this** може користити за приступ заклоњеним (private пољима) члановима класе.

Видимо да метода Print() има дефинисану локалну промењиву **d** са истим називом као и поље d.

```
Console.WriteLine("Ime je: {0}", d);          приказује вредност локалне промењиве d = 12
Console.WriteLine("Ime je: {0}", this.d);     приказује вредност поља d = 7
```

Кључна реч "this" се користи када у класи постоји неколико конструктора, онда се this користи за позивање конструктора.

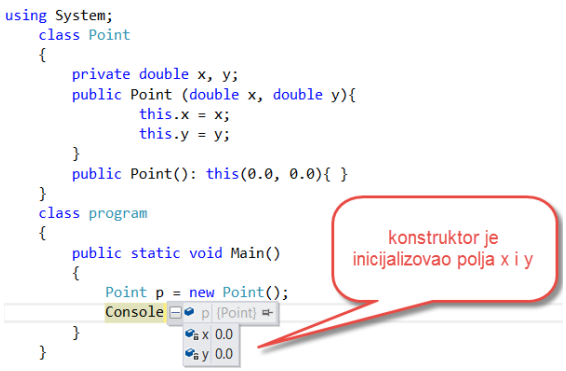
```
using System;
    class Podaci
    {
        public Podaci(): this("skola")
        {
            Console.WriteLine("Konstruktor bez parametra");
        }
        public Podaci(string Name)
        {
            Console.WriteLine("Konstrukor sa parametrom");
        }
    }
    class program
    {
        public static void Main()
        {
            Podaci objekat = new Podaci();
            Console.ReadKey();
        }
    }
```



Када један конструктор позива други, прво се извршава **позвани конструктор**.

1. **Задатак:** Креирати два поља (x и y) и два конструктора. Конструктор без параметара, формира тачку у координатом почетку позивајући конструктор са параметрима.

```
using System;
    class Point
    {
        private double x, y;
        public Point (double x, double y){
                this.x = x;
                this.y = y;
        }
        public Point(): this(0.0, 0.0){ }
    }
    class program
    {
        public static void Main()
        {
            Point p = new Point();
            Console.ReadKey();
        }
    }
```



Могли смо да користимо и следећи израз:
`public Point(): this(0, 0){ }`

АНАЛИЗИРАТИ СЛЕДЕЋЕ ПРИМЕРЕ И ДАТИ ТАЧАН ОДГОВОР:

221. Дата је дефиниција класе у програмском језику C# и састоји се од два конструктора, једне методе и поља x. У дефиницији се користи службена реч **this**. Анализирати дати код и проценити тачност следећих исказа. Заокружити бројеве испред тачних исказа:

```
class TestPrimer {
        public double x;
        public TestPrimer(double x){
            this.fun();
            this.x = x;
        }
        public TestPrimer(){
            Console.WriteLine("Podrazumevani konstruktor");
            this(23);
        }
        public void Fun(){
            Console.WriteLine("Poziv metoda fun()");
        }
    }
```

1. this.Fun() у конструктору TestPrimer(double x) може се поједноставити и заменити само са Fun().
2. this.x у конструктору TestPrimer(double x) може се поједноставити и заменити само са x.
3. позив конструктора this(23) унутар другог конструктора TestPrimer() је прво шта се извршава и мора се писати одмах после декларације `public TestPrimer():this(23)`
4. this(23) у конструктору Test() мора се заменити са прецизнијим изразом this(23.0).

2

223. Дата је дефиниција класе у програмском језику C# и састоји се од два конструктора и поља x и y. У четвртом реду допунити подразумевани конструктор без параметара класе Point, који формира тачку у координатном почетку, позивајући конструктор са параметрима.

```
1. public class Point {
2.     private double x, y;
3.     public Point(double x, double y) {
4.             this.x = x; this.y = y;
5.         }
6.     public Point()_____________
7. }
```

Заокружити бројеве испред тачних одговора:

1. `public Point(): base(0, 0) { }`
2. `public Point(): this(0, 0) { }`
3. `public Point() { Point(0,0); }`
4. `public Point(): this(0.0, 0.0) { }`
5. `public Point() { Point(0.0, 0.0); }`

2

222. Дата је дефиниција класе у програмском језику C# и састоји се од два конструктора, методе и поља x и y. У петом реду дефинисан је конструктор са параметрима који формира тачку са координатама x и y. Заокружити наредбе којима се може допунити дефиниција конструктора. **2**

```
1. public class Point {
2.     private double x, y;
3.     public Point() { x = 0; y = 0; }
4.     public void set(double xx, double yy) { x = xx; y = yy; }
5.     public Point(double x, double y) { _____________; }
6. }
```

Заокружити бројеве испред тачних одговора:

1. this.x=x; this.y=y;
2. x=x; y=y;
3. set(x,y);
4. set(this.x,this.y);
5. x=this.x; y=this.y;

167. Дата је дефиниција класе у програмском језику C# и састоји се од два конструктора, методе и поља x и y. У шестом реду написати конструктор копије објекта класе Point. **2**

```
1.  public class Point {
2.      private double x, y;
3.      public Point() { x = 0; y = 0; }
4.      public void Set(double xx, double yy){ x=xx; y=yy; }
5.      public Point(Point p) {
6.          _____________________//Odgovor
7.      }
8.  }
```

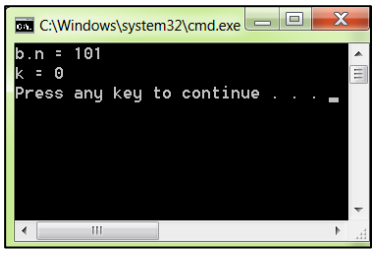
Заокружити број испред очекиваног одговора:

1. this(p.x, p.y);
2. this(p);
3. Set(p);
4. Set(p.x, p.y);

183. Дат је код програма у програмском језику C# и састоји се од две класе у једној датотеци. Анализирати дати код и проценити која се вредност поља b.n приказује првом наредбом `Console.WriteLine` (ред седам) приликом извршавања овог програма. Заокружити број испред очекиваног одговора:

```
using System;

class Program
{
    static void Main(string[] args)
    {
        int k = 0;
        Brojac b = new Brojac();
        for (int i = 0; i < 100; i++) Inc(b, k);
        Console.WriteLine("b.n = " + b.n);
        Console.WriteLine("k = " + k);
    }
    public static void Inc(Brojac b, int k)
    {
        b.n++;
        k++;
    }
}
class Brojac {
    public int n;
    public Brojac(int n) { this.n = n;}
    public Brojac() { this.n = 1; }
}
```



1. b.n = 101
2. b.n = 100
3. b.n = 99
4. b.n = 0
5. b.n = 1