Службена реч this

Службена реч this се упућује на инстанцу класе или се користи као модификатор првог параметра проширене методе.
Најчешће се користи за представљање (qualify) чланова сакривених под сличним именом, за додавање објекта као параметра другим методама и за декларацију индексера.
Статички чланови метода или функција, пошто постоје на нивоу класе и нису део објекта, не могу користити this.

Пример: У VS креирати нов пројекат са именом Klase (namespace Klase), креирати класу Program и у њу сместити главну методу и методу Pomoc. У Solution Explorer десни клик на назив пројекта Klase, Add New Item,па на прозору изабарти Class (empty class definition) и креирати класу Tacka. Стартовати програм.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Klase
{
    class Program
    {
        static void Pomoc()
        {
            Tacka pocetna = new Tacka();
        }
        static void Main()
        {
            Pomoc();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Klase
{
    class Tacka
    {
        public Tacka()
        {
```

```
            // TODO:
        }
    }
}
```

Када се код стартује нема грешке пошто компајлер аутоматски генерише код за дифолтни конструктор за Point klasu.
Код за овај конструктор се не види пошто компајлер не генерише исказе.
Програмери често додају //TODO: пошто VS препознаје ову форму коментара и омогућава брзо лоцирање линије кода било где у апликацији преко Task List прозора.

Изменити код:

```
class Tacka
{
     public Tacka(int x, int y)
      {
          Console.WriteLine($"x:. y:");
      }
}
```

Сада компајлер генерише грешку:
There is no argument that corresponds to the required formal parameter 'x' of'Tacka.Tacka(int, int)'
Ова порука говори да позив дифолтном конструктору унутар Pomoc методе је сада погрешан пошто више не постоји дифолтни конструктор.
Написали смо наш конструктор за Tacka класу, па компајлер не генерише дифолтни конструктор.
Ако се дода дифолтни конструктор:

```
namespace Klase
{
    class Tacka
    {
        public Tacka()
        {
            Console.WriteLine("Pozvan difoltni konstruktor");
        }
        public Tacka(int x, int y)
        {
            Console.WriteLine($"x:{x}. y:{y}");
        }
    }
}
```

Сада програм ради и исписује се на екрану: Pozvan difoltni konstruktor
Додати у код:

```
Tacka pocetna = new Tacka();
Tacka krajnja = new Tacka(1366, 768);
```

Овако се додају два int поља у класу Tacka да би представили координате специфичне тачке, па се могу искористити модификовани конструктори за иницијализацију ових поља.
Даје:

Pozvan difoltni konstruktor
x:1366. y:768

Модификовати код:

```
class Tacka
{
        private int x, y;
        public Tacka()
        {
             Console.WriteLine("Pozvan difoltni konstruktor");
        }
        public Tacka(int x, int y)
        {
             x = x;
             y = y;
      }
}
```

И овакав код ће радити али поставља се питање да ли компајлер зна да је у изразу x = x, прво x поље а друго x параметар?
Компајер то не зна јер поље сакрива све параметре и поља са истим именом.
Зато овај код додељује вредности параметара самом себи што је потпуно непотребно.
Зато се користи службена реч this којом се дефинишу шта су поља а шта параметри са истим именом.
Стављање this испред неке промењиве има значење „поље у овом (this) објекту“.

Модификовати конструктор:

```
public Tacka()
{
     this.x = -1;
     this.y = -1;
}
```

Ако се курсором пређе преко имена x види се да се сада јасно разликује шта је поље а шта параметар.
Укуцати:

```
public Tacka(int x, int y)
{
     this.x = x;
     this.y = y;
}
```

Методе које припадају класи и које раде са подацима који припадају инстанци класе се називају методе инстанце.

Модификовати код убацивањем методе инстанце RacunanjeDaljine у класи Tacka.
Метода прихвата један аргумент drugi и враћа вредност типа double.

Циљ методе је рачунање и враћање растојања између објеката Tacka који се користи за позивање и објекта Tacka који се прослеђује као параметар.

```
public double RacunanjeDaljine(Tacka drugi)
{
     int xrazlika = this.x - drugi.x;
     int yrazlika = this.y - drugi.y;
     return Math.Sqrt((Math.Pow(xrazlika, 2) +
               Math.Pow(yrazlika, 2)));
}
```

Модификовати код убацивањем у методу Pomoc:

```
static void Pomoc()
{
     Tacka pocetna = new Tacka();
     Tacka krajnja = new Tacka(1366, 768);
     double rastojanje = pocetna.RacunanjeDaljine(krajnja);
     Console.WriteLine($"Rastojanje je {rastojanje}");
}
```

Види се да промењива rastojanje се иницијализује резултатом који се добија када се позове метода RacunanjeDaljine на објекат pocetna прослеђујући објекат krajnja као аргумент.
Овде this.x и this.y указују на вредности координата које постоје у пољима објекта pocetna а то су -1 и -1.
Вредности координата у пољима објекта krajnja, идентична са пољима објекта drugi, су 1366 и 768.

Деконструкција објекта

Конструктори се користе за креирање и иницијализацију објекта, обично попуњавањем поља која садржи објекат.
Деконструктори се користе за испитивање објекта и извлачење вредности из његових поља.
Може се креирати деконструктор у класи Tacka за преузимање вредности поља x и y:

```
public void Deconstruct(out int x, out int y)
{
     x = this.x;
     y = this.y;
}
```

Назив деконструктора је увек Deconstruct; мора бити типа void; мора имати најмање један параметар и ти параметри се попуњавају вредностима из поља у објекту.
Параметри се обележавају модификатором out; то значи да ако им се додели вредност, те вредности ће бити прослеђене назад до позиваоца; код у телу деконструктора додељује вредност која се мора вратити до параметара.
Позивање деконструктора:

```
Tacka nova = new Tacka();
(int nekox, int nekoy) = nova;
Console.WriteLine($"{nekox}, {nekoy}");
```

Даје: -1, -1

У позадини позива, компајлер покреће деконструктор, даје му промењиве дефинисане у торци као параметре.
Код у деконструктору попуњава ове промењиве.

Задаци за самосталан рад

1. Креирати класу Broj и у њој дифолтни конструктор и један конструктор са double аргументом. Оба конструктора имају само //TODO:. Главна функција инстанцира нови објекат класе Broj без аргумената и исписује на конзоли тај објекат.

2. Користећи претходни пример, конструктор треба да додели вредност своме пољу x тако што израчуна квадрат броја 10.0 и на то дода 0.1. Коришћењем службене речи this упутити на правилно поље конструктора. Приказати добијени резултат на конзоли.

3. Променити тело конструктора Broj(double x) тако да постане метода инстанце. Креирати методу Racunanje која ће коришћењем службене речи this позвати садржај поља конструктора Broj(double x) и израчунати по формули из претодног задатка потребан резултат.