

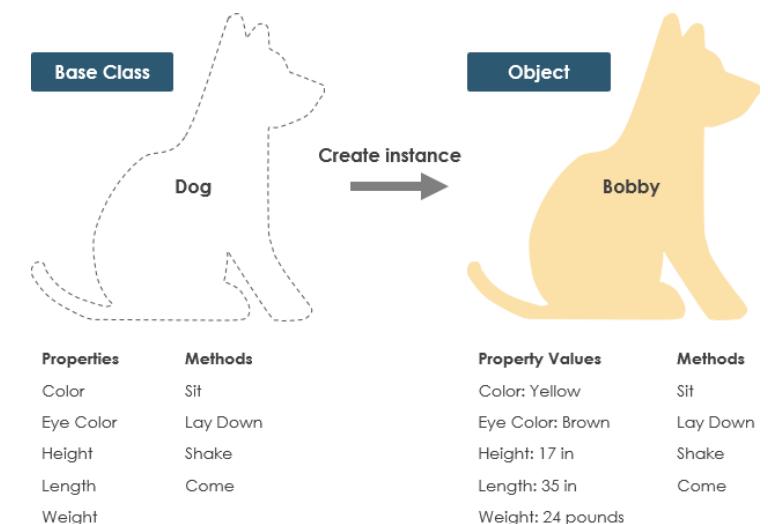
Класове и обекти

Полета, свойства, конструктори, методи



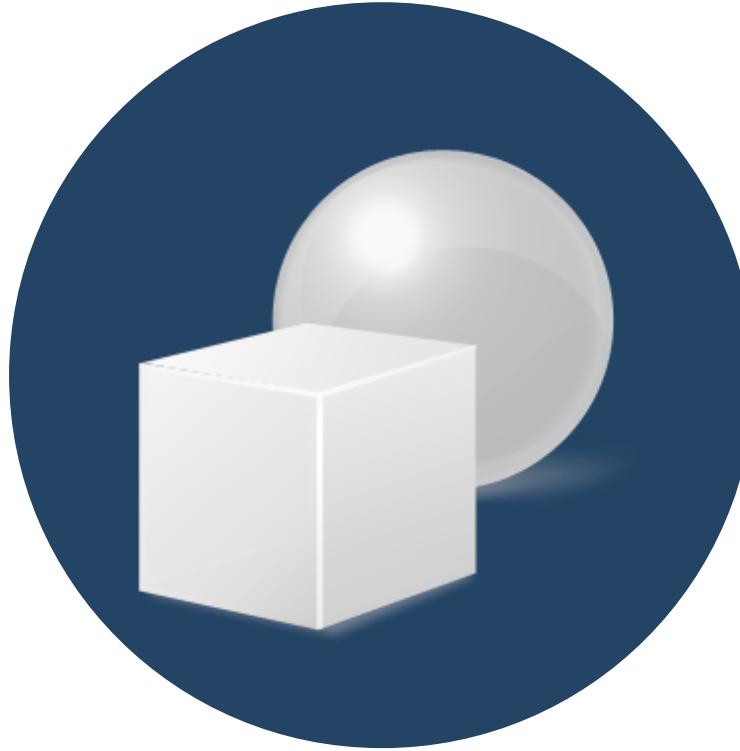
Проект "Отворено учебно съдържание по програмиране и ИТ", СофтУни Фондация

<https://github.com/BG-IT-Edu>



Курс "ООП" Софтуерни и хардуерни науки

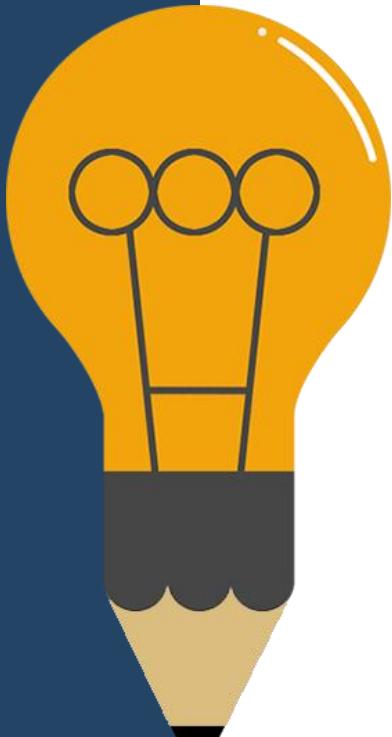
1. Обекти и класове
2. Дефиниране на прости класове
3. Полета и свойства
4. Конструктори
5. Методи



Какво е обект? Какво е клас?

Обекти

- **Обектът** е съвкупност от **именувани стойности**.
 - Например обект за рожден ден съдържа **ден**, **месец** и **година**.
 - Създаване на обект за **рожден ден** :



```
var birthday = new { Day = 22, Month = 6, Year = 1990 };
```

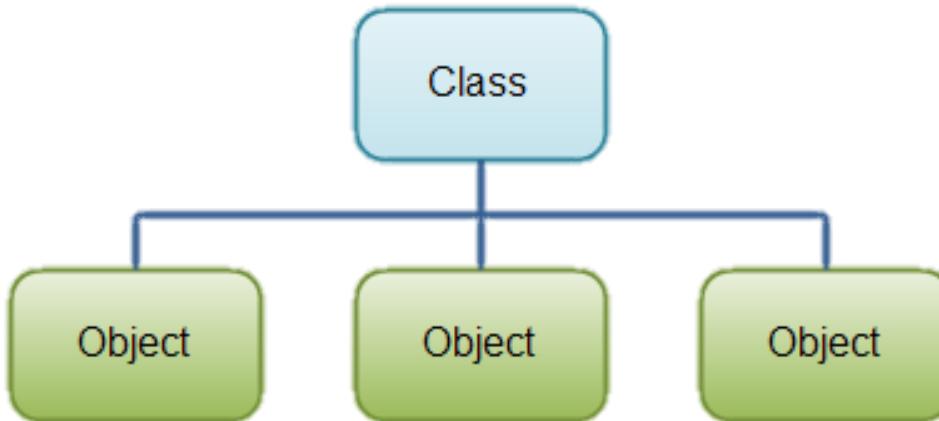
Класове

- В програмирането **класовете** задават **структура на обектите**
 - Имат ролята на **шаблон за обекти** от един и същ тип
- Класовете дефинират:
 - **Данни** (полета, свойства), например **Day, Month, Year**
 - **Действия** (методи), например **AddDays(count), Subtract(date)**



Класове

- Един клас може да има множество инстанции (обекти)



- Примерен клас: **DateTime**
- Примерни обекти: **peterBirthday**, **mariaBirthday**

object
peterBirthday

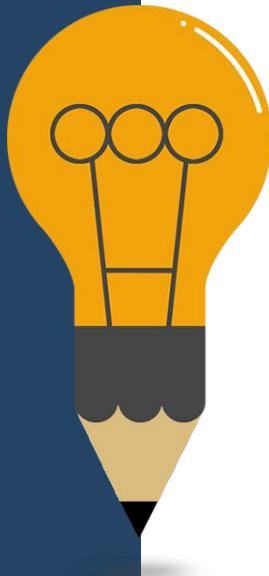
Day = 27
Month = 11
Year = 1996

object
mariaBirthday

Day = 3
Month = 10
Year = 2002

Обекти (Инстанции на класове)

- Създаването на обект от дефиниран клас се нарича **инстанциране**
- **Инстанцията** е самият обект, който се създава по време на изпълнение (runtime)
- Всички инстанции имат еднакво **поведение**



```
DateTime date1 = new DateTime(2018, 5, 5);
```

```
DateTime date2 = new DateTime(2016, 3, 5);
```

```
DateTime date3 = new DateTime(2013, 12, 31);
```

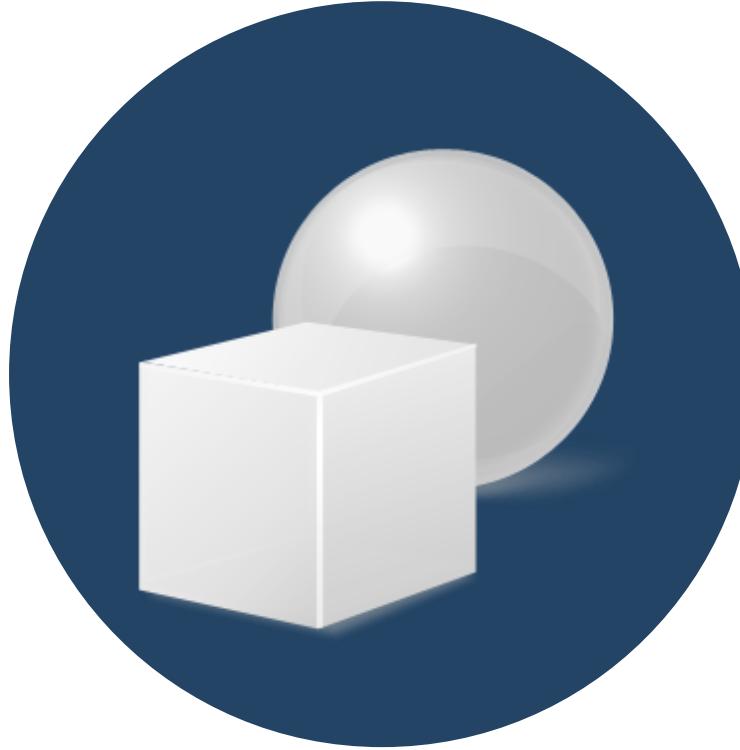
Примери: Обекти и класове

```
DateTime peterBirthday = new DateTime(1996, 11, 27);
DateTime mariaBirthday = new DateTime(1995, 6, 14);
Console.WriteLine($"Peter's birth date: {peterBirthday:d-MMM-yyyy}");
// 27-Nov-1996

Console.WriteLine($"Maria's birth date: {mariaBirthday:d-MMM-yyyy}");
// 14-Jun-1995

DateTime mariaAfter18Months = mariaBirthday.AddMonths(18);
Console.WriteLine($"Maria after 18 months: {mariaAfter18Months:d-MMM-yyyy}");
// 14-Dec-1996

TimeSpan ageDiff = peterBirthday.Subtract(mariaBirthday);
Console.WriteLine("Maria older than Peter by: {ageDiff.Days} days");
// 532 days
```



Дефиниране на прости класове

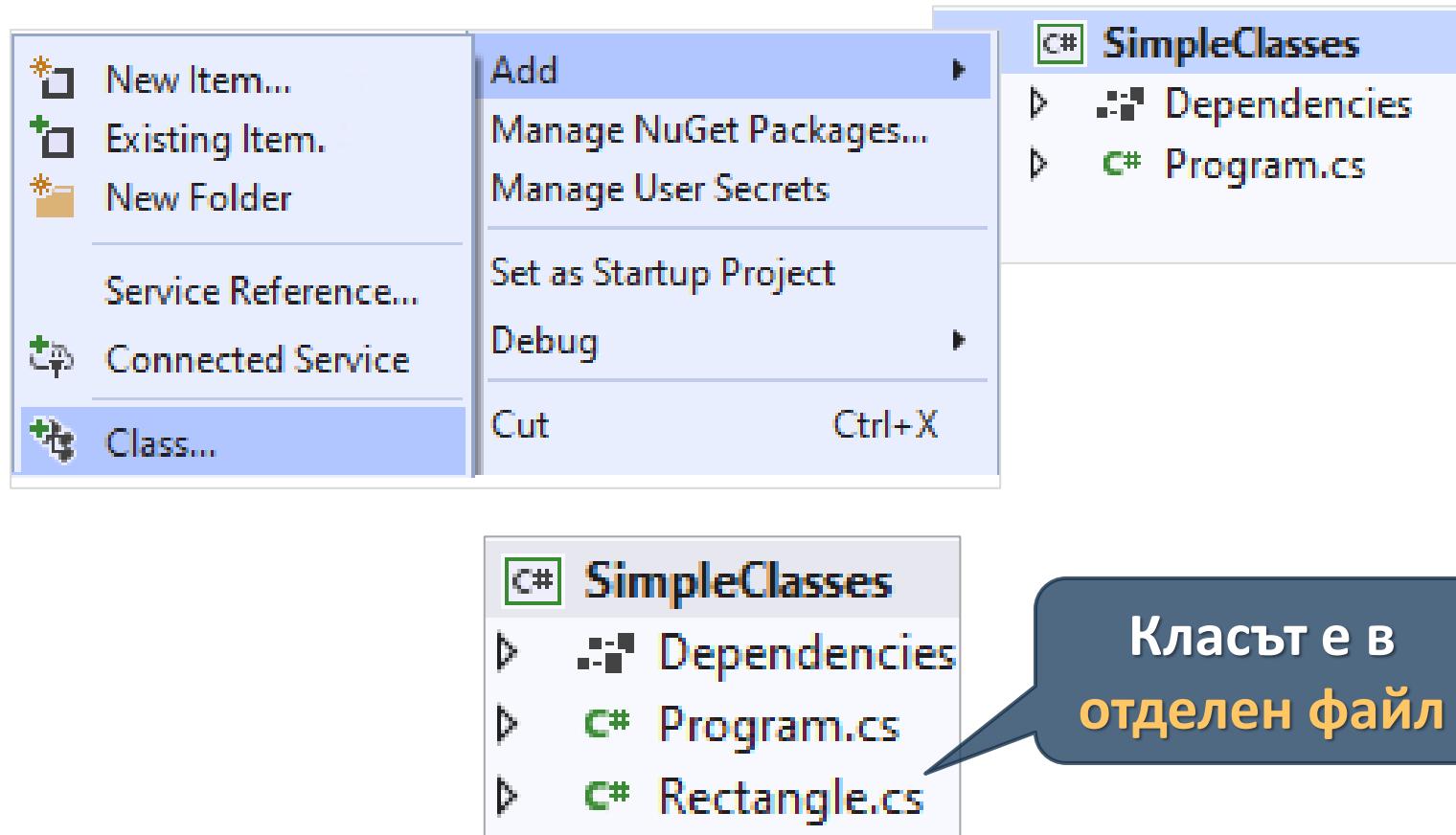
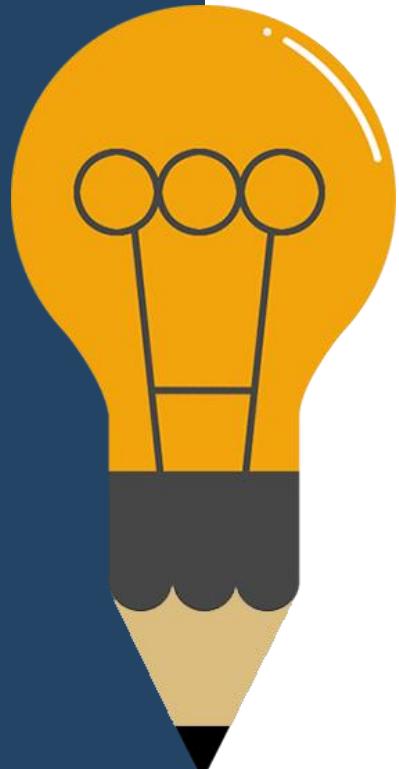
Дефиниране на прости класове

- Можем да **дефинираме клас** чрез следния синтаксис:



Създаване на прост клас Rectangle

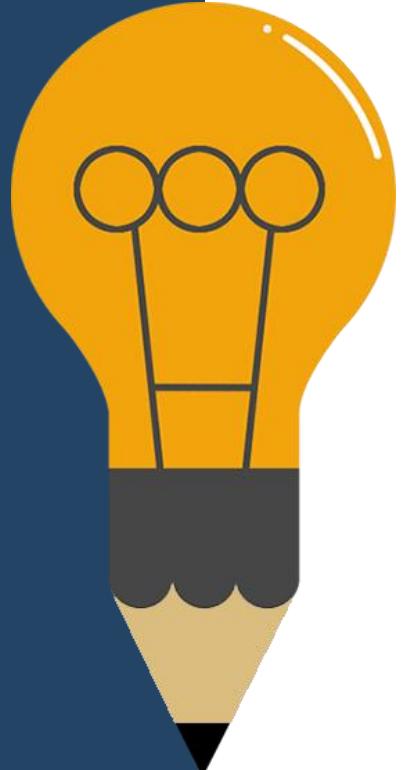
- Създайте файл за класа: [Project] → [Add Class] или:
десен бутон на проекта: [Add] → [New Item] → [Class]



Класът е в
отделен файл

Именуване на класове

- Класовете се именуват със съществителни имена, използвайки **PascalCase**
- Използвайте **описателни съществителни имена**
- **Избягвайте абревиатури** (с изключение на поизвестните като URL, HTTP, etc.)



```
class Dice { ... }  
class BankAccount { ... }
```



```
class TPMF { ... }  
class bankaccount { ... }  
class intcalc { ... }
```



- Членовете се **декларират** вътре в класа
- Членовете могат да бъдат:
 - **Полета** (данни)
 - **Свойства**
(данни + логика)
 - **Методи** (действия)
 - **Конструктори**
 - **Други**

```
class Rectangle
{
    private int width;           Поле
    public int Width { get; set; } Свойство
    public void CalcArea() {...} Метод
    public Rectangle() {}        Конструктор
}
```

Разлика между класове и обекти

- Класовете задават **структура** за създаване на **обекти**
- Обектът е единична **инстанция** на класа

```
class  
Rectangle
```

Width: int
Height: int
Color: string
CalcArea(...)

Име на класа

Данни на класа

Методи на
класа

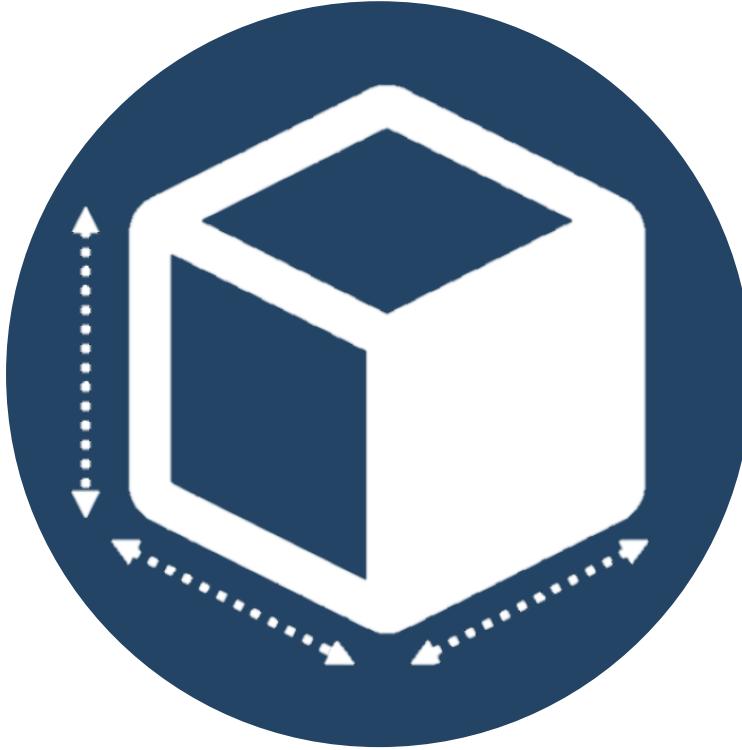
```
object  
firstRect
```

Width = 6
Height = 3
Color = "blue"

Име на
обекта

Данни на
обекта





Полета и свойства

Съхраняване на данни в клас

Полета и модификатори

- Полетата на класа имат **тип и име**
- **Модификаторите** определят **достъпността** (видимостта)

Модификатор

Полетата трябва винаги
да бъдат частни
(скрити)

Полетата могат да
бъдат от **всякакъв тип**

```
public class Rectangle
{
    private int width;
    private int height;
    private string color;
}
```

- Използват се, за да се създадат **accessor-и и mutator-и**
(getter-и и setter-и)

```
public class Rectangle
{
    private int width;
    public int Width
    {
        get { return this.width; }
        set { this.width = value; }
    }
}
```

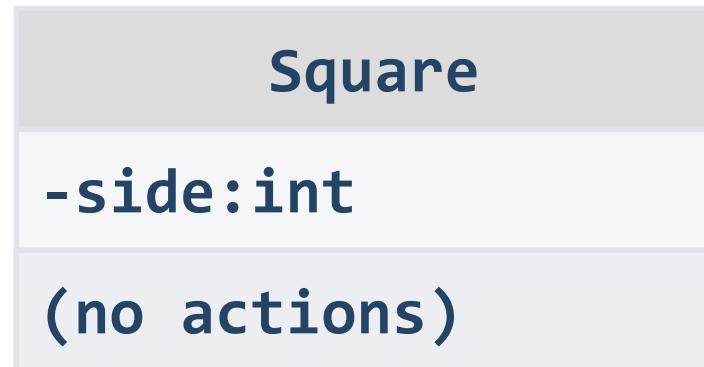
Полето е
частно (скрито)

Getter-ът дава
достъп до полето

Setter-ът позволява
промяна на полето

Задача: Клас "квадрат"

- Създайте клас **Square**, който има частно поле **side** и публично свойство **Side**.

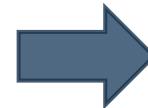
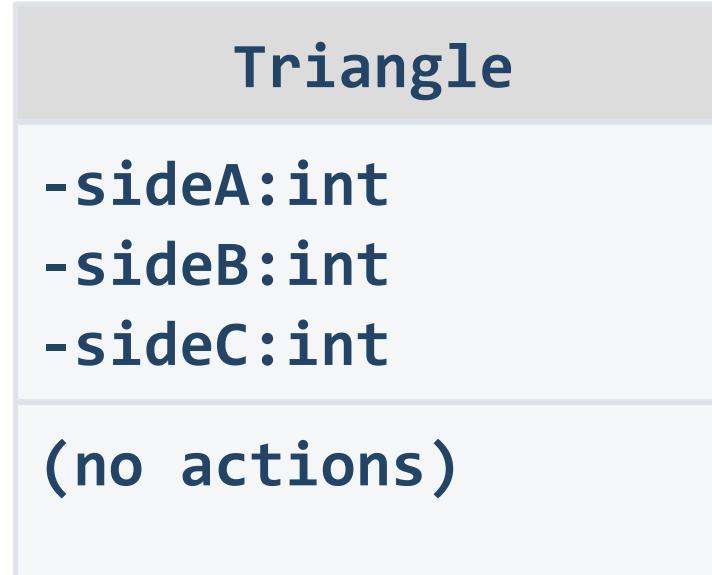


```
private int side;  
  
public string Side  
{  
    get { return this.side; }  
    set { this.side = value; }  
}
```

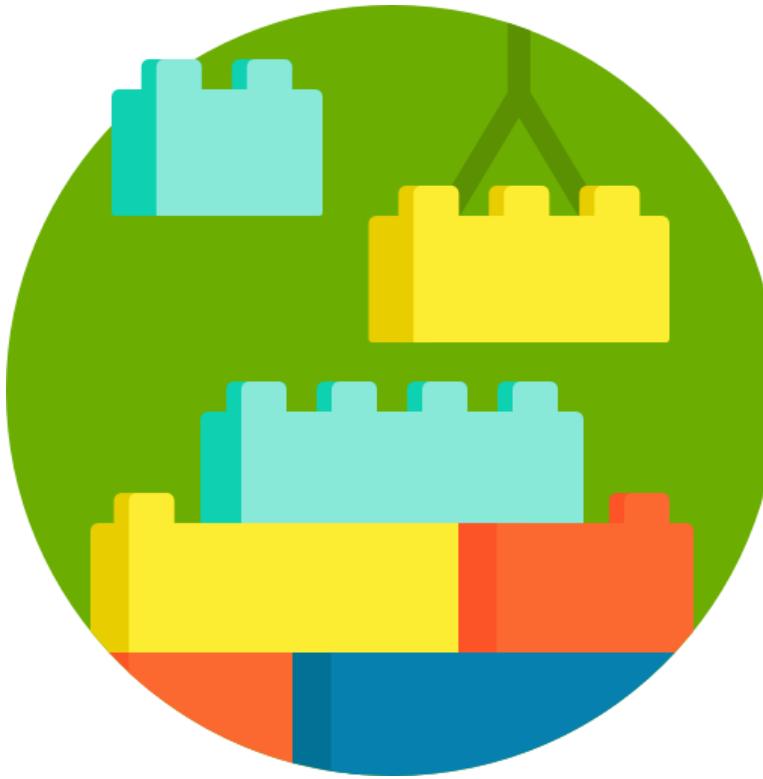
- Важно:** Прочетете изискванията в **документа с упражненията**, преди да предадете решението си в **Judge**

Задача: Клас "триъгълник"

- Създайте клас **Triangle**, който има частни полета за трите страни – **sideA**, **sideB** и **sideC**, и публични свойства за същите страни



```
private int sideA;  
  
public string SideA  
{  
    get { return this.side; }  
    set { this.side = value; }  
}  
  
// TODO: Добавете кода за другите 2 страни
```



Конструктори

Инициализация на обекти

Конструктори

- Когато **конструкторът** е извикан, създава **инстанция** на класа и обикновено инициализира неговите членове
- Класовете в C# се инициализират с **ключовата дума new**



```
public class Rectangle
{
    public Rectangle() {}
}
```

```
public class StartUp
{
    static void Main()
    {
        Rectangle figure = new Rectangle();
    }
}
```

- Конструкторите **задават първоначалното състояние на обекта**

```
public class Rectangle {  
    public int Width { get; set; }  
    public int Height { get; set; }  
    public string Color { get; set; }  
  
    public Rectangle(int width, int height, string color)  
    {  
        this.Width = width;  
        this.Height = height;  
        this.Color = color;  
    }  
}
```

- Можем да създаваме **обекти** от дефинирания клас:

```
Rectangle r1 = new Rectangle(30, 20, "white");
Rectangle r2 = new Rectangle(15, 15, "green");

Console.WriteLine("r1 area: " + r1.Width * r1.Height);
Console.WriteLine("r2 area: " + r2.Width * r2.Height);
```

Задача: Клас "триъгълник" с конструктор

- Използвайте класа **Triangle** от предишната задача и добавете **конструктор**, който приема трите му страни

```
private int sideA;  
  
public Triangle(int sideA, int sideB, int sideC)  
{  
    this.SideA = sideA;  
    // TODO: добавете кода за другите 2 страни  
}
```



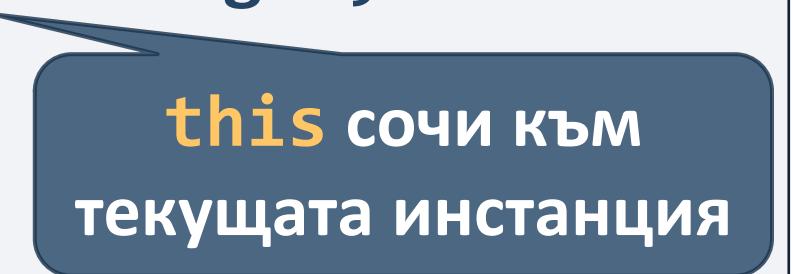
Дефиниране на поведение на класа

Методи, параметри и връщана стойност

- Съхраняват **изпълним код**

```
public class Rectangle
{
    public int Width { get; set; }
    public int Height { get; set; }

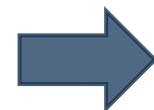
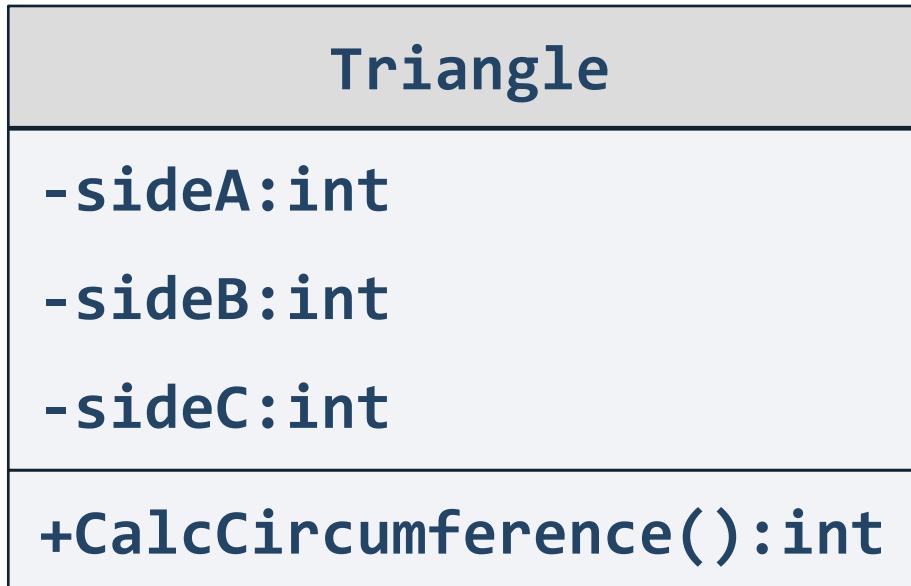
    public int CalcArea()
    {
        int area = this.Width * this.Height;
        return area;
    }
}
```



this сочи към
текущата инстанция

Задача: Клас "триъгълник" с метод

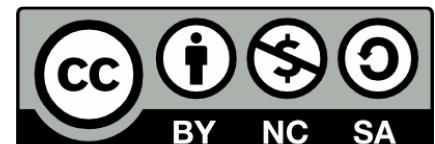
- Към класа **Triangle** добавете метод **calcCircumference()**, който изчислява обиколката на триъгълника.



```
public int CalcCircumference()
{
    return this.SideA
        + this.SideB + this.SideC;
}
```

- Класовете задават структура за описание и създаване на обекти
- Обектите са инстанции на дадения клас
- Класовете имат полета, свойства, методи, конструктори и други членове
- Конструктори:
 - Извикват се при създаване на нови инстанции
 - Инициализират състоянието (state) на обекта

Въпроси?

- Този курс (презентации, примери, демонстрационен код, упражнения, домашни, видео и други активи) представлява **свободно учебно съдържание** и се разпространява под свободен лиценз **CC-BY-NC-SA**
- Проект "**Отворено учебно съдържание по програмиране и ИТ**" към Фондация "Софтуерен университет":
 - <https://github.com/BG-IT-Edu>