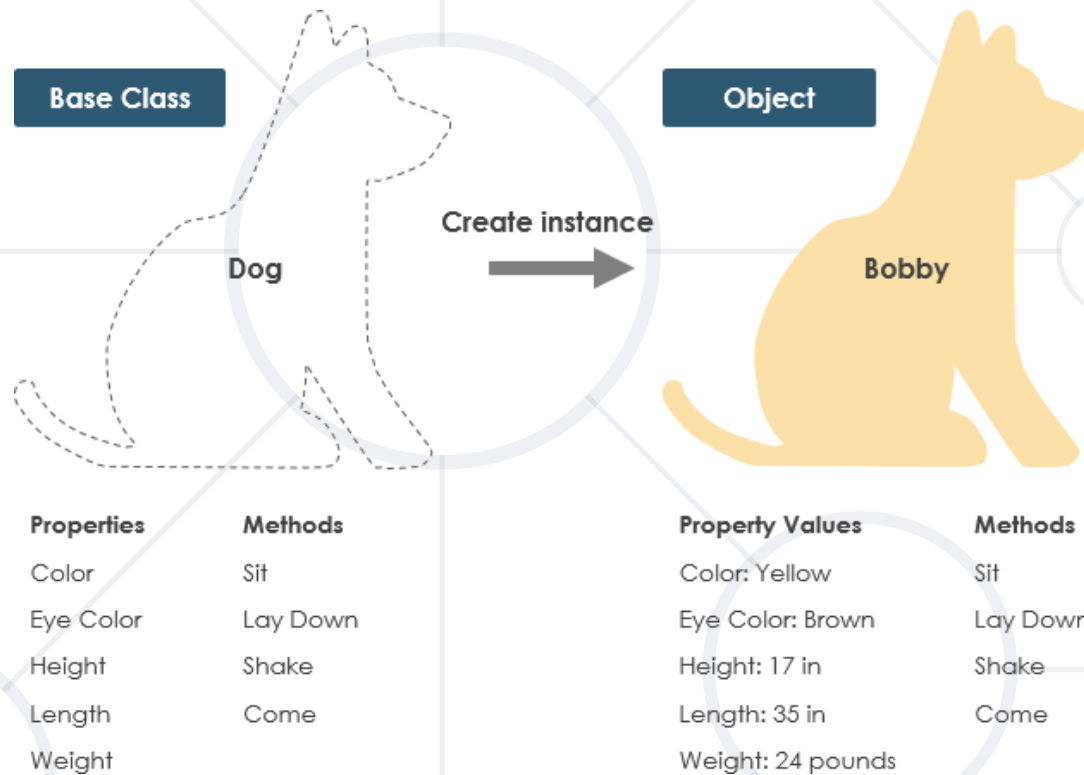


Класове и обекти



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://about.softuni.bg/>

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





Какво е обект? Какво е клас?

Обекти

- **Обектът** съдържа поредица от наименувани стойности.
 - Например обект за рожден ден съдържа **ден, месец** и **година**.
 - Създаване на обект **за рожден ден** :

Birthday

Day = 22

Month = 6

Year = 1990

Име на
обекта

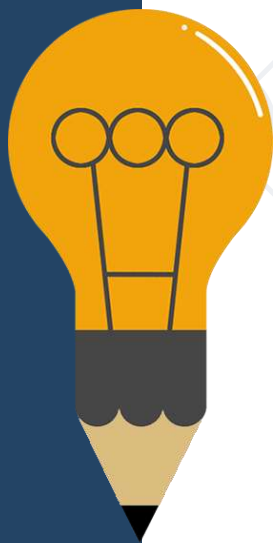
Свойства
на обекта

```
var day = new DateTime(
    2019, 2, 25);
Console.WriteLine(day);
```

Създаваме **нов** обект
от тип **DateTime**

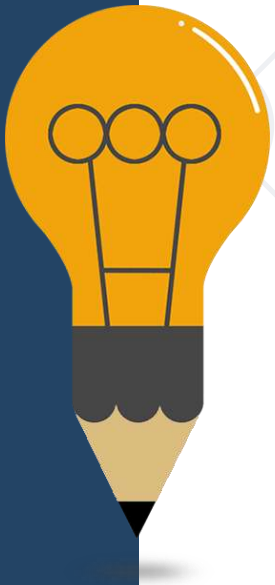
Операторът **new**
създава нов обект

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



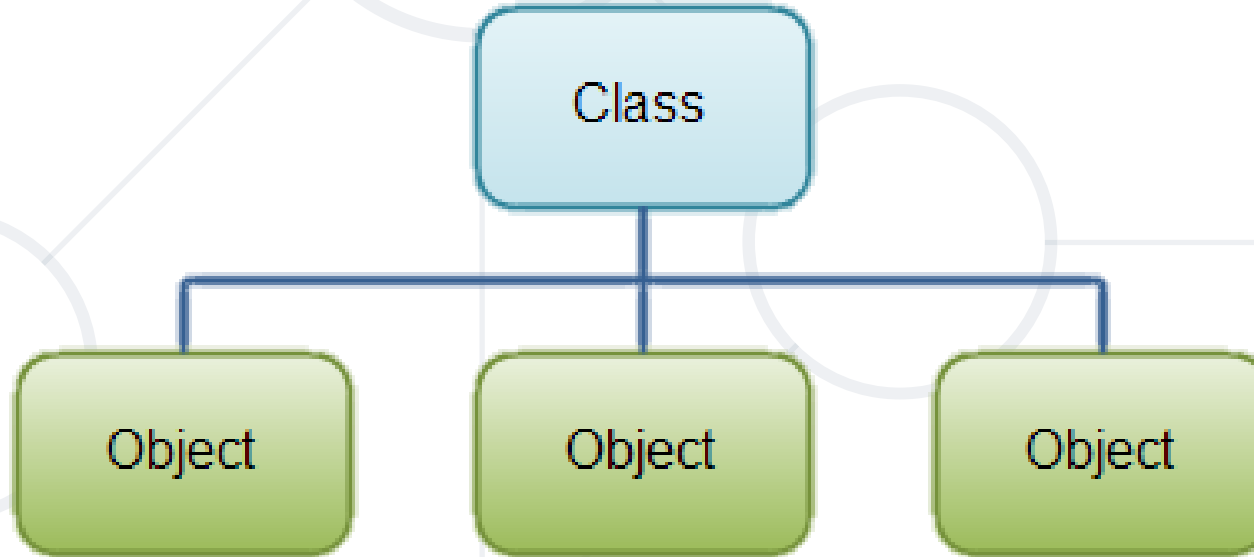
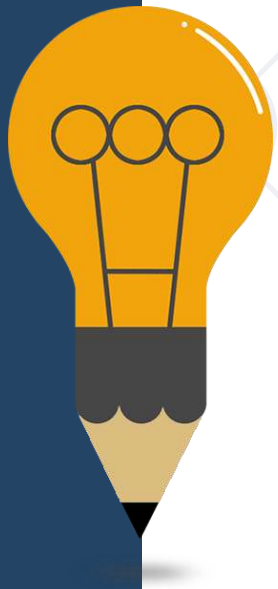
Класове

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



Класове

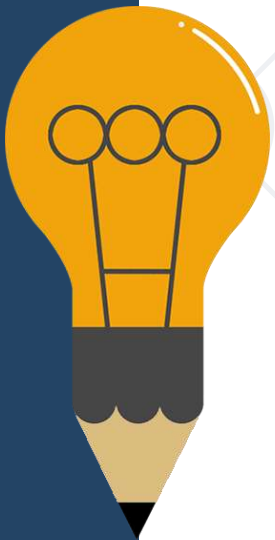
- Един клас може да има множество инстанции (обекти)
 - Примерен клас: **DateTime**
 - Примерни обекти: **peterBirthday**, **mariaBirthday**



Обекти (Инстанции на класове)

- Създаването на обект от дефиниран клас се нарича **инстанциране**
- **Инстанцията** представлява 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);
```



```
DateTime peterBirthday = new DateTime(1996, 11, 27);
DateTime mariaBirthday = new DateTime(1995, 6, 14);
Console.WriteLine("Peter's birth date: {0:d-MMM-yyyy}",peterBirthday);
// 27-Nov-1996
Console.WriteLine("Maria's birth date: {0:d-MMM-yyyy}",mariaBirthday);
// 14-Jun-1995
var mariaAfter18Months = mariaBirthday.AddMonths(18);
Console.WriteLine("Maria after 18 months: {0:d-MMM-yyyy}", mariaAfter18Months);
// 14-Dec-1996
TimeSpan ageDiff = peterBirthday.Subtract(mariaBirthday);
Console.WriteLine("Maria older than Peter by: {0} days", ageDiff.Days);
// 532 days
```




Дефиниране на прости класове

Дефиниране на прости класове

- Класът е **конкретна имплементация** на АТД (абстрактен тип данни)
- Класовете задават **структура** за **описване** и **създаване** на обекти

Ключова дума

Име на класа

```
class Rectangle
```

```
{
```

```
...
```

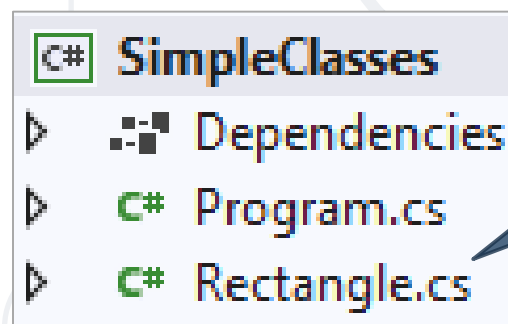
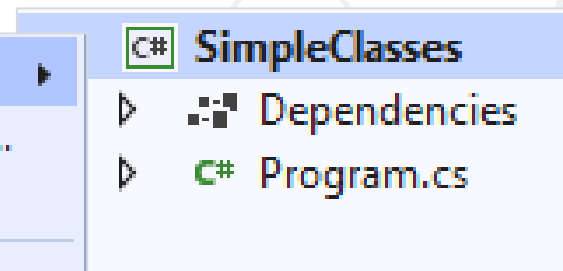
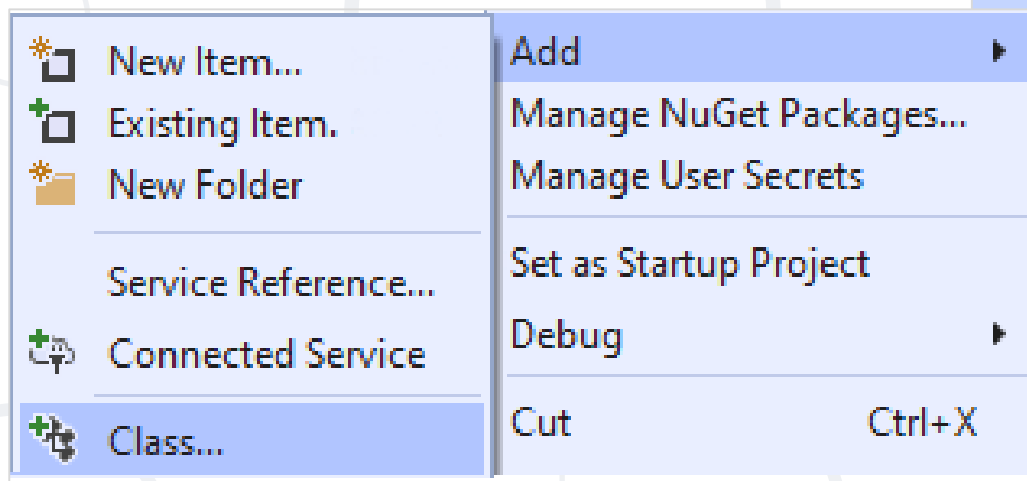
```
}
```

Тяло на класа



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 **PascalCasing**
- Използвайте **описателни съществителни имена**
- **Избягвайте аббревиатури** (с изключение на по-известните като URL, HTTP, etc.)

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



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



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

```
class Rectangle
```

```
{
```

```
    int width;
```

Поле

```
    int Width { get; set; }
```

Свойство

```
    void CalcArea() { ... }
```

Метод

```
}
```

Клас Rectangle (правоъгълник) - пример

- Класът **Rectangle** съдържа свойствата **Width** and **Height**

Rectangle.cs

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

- Един клас може да има **МНОЖЕСТВО ИНСТАНЦИИ** (обекти)

```
class Program
```

```
{
```

```
    public static void Main()
```

```
{
```

```
    Rectangle firstRect = new Rectangle();
```

```
    Rectangle secondRect = new Rectangle();
```

```
}
```

```
}
```

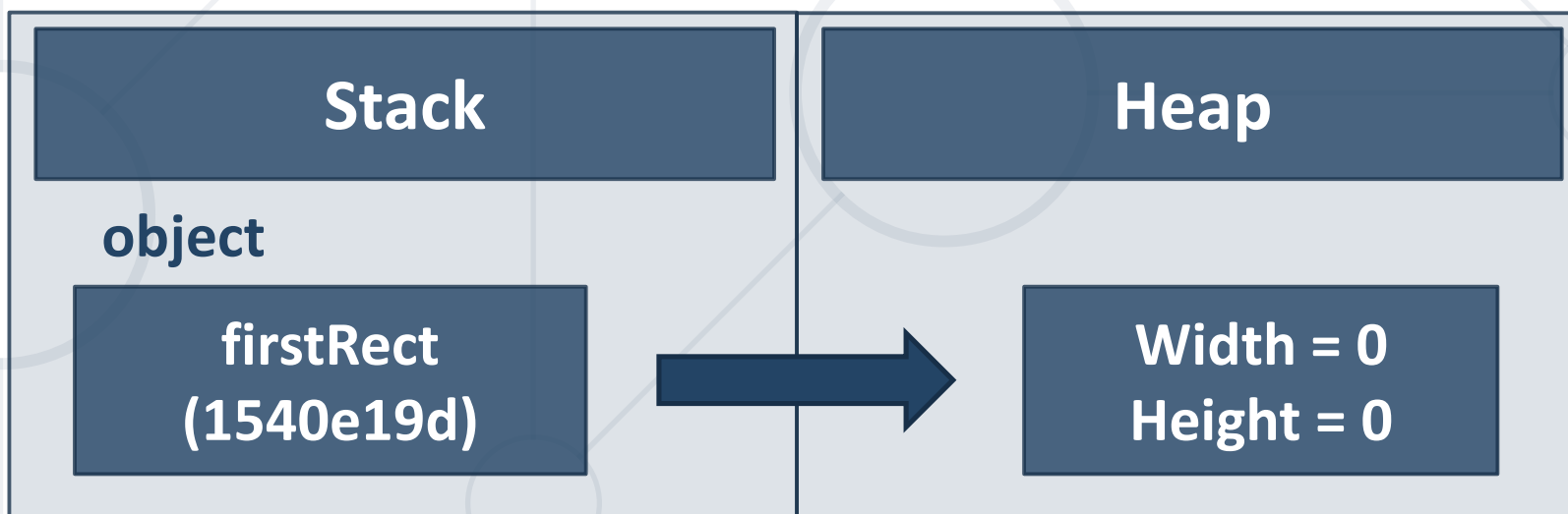
Използвайте ключовата дума **new**, за да създадете обект

Променливата пази **референция** към обекта

Референция към обекта

- Декларирането на променлива създава **референция** в стека
- Ключовата дума **new** заделя място в динамичната памет (heap)

```
Rectangle firstRect = new Rectangle();
```



Дефиниране на прост метод в клас

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

    public int CalcArea()
    {
        return Width * Height;
    }
}
```

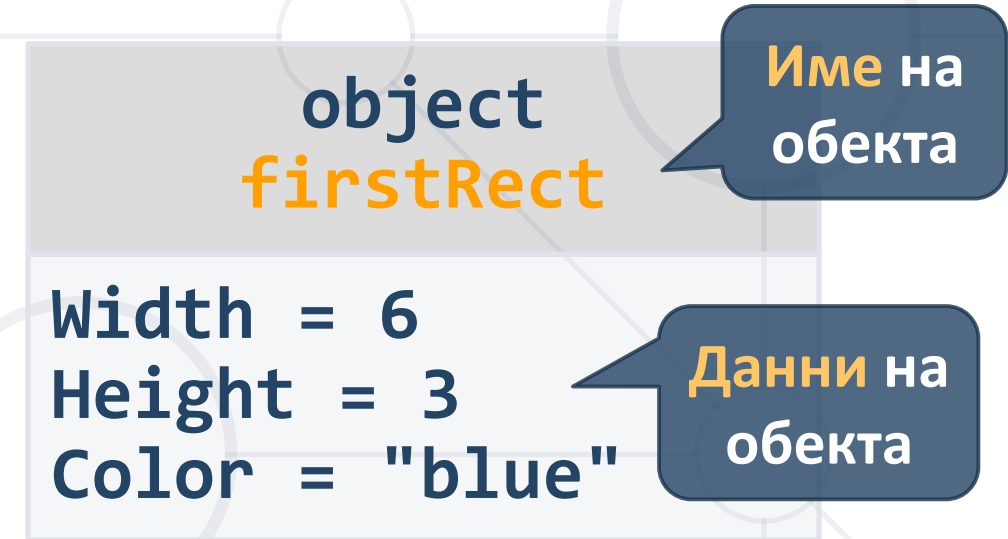
Методите дефинират
действия в класовете

Разлика между класове и обекти

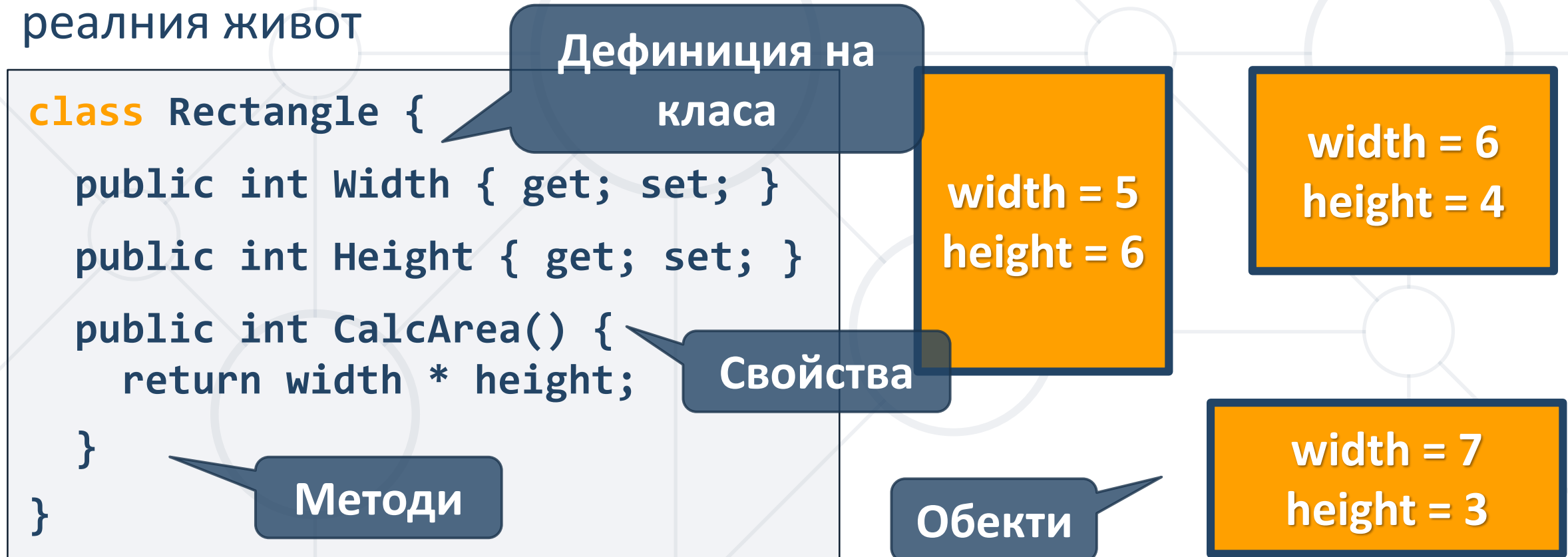
- Класовете задават **структура** за създаване на **обекти**



- Обектът** е единична **инстанция** на класа



- **Обектно-ориентираното програмиране** е концепция за използването на **класове** и обекти за моделиране на примери от реалния живот





Съхраняване на данни в клас

- Полетата на класа имат **тип** и **име**
- Модификаторите определят достъпността.

Модификатор

Полетата трябва
винаги да бъдат
частни

Полетата могат
да бъдат от
всякакъв тип

```
public class Rectangle
{
    private string color;
    private int width;
    private int height;
    private int[] sections;
    private Shape type;
    public int CalcArea() { ... }
}
```

- Използват се, за да се създадат **accessors** и **mutators** (**getter-и** и **setter-и**)

```
public class Rectangle
```

```
{
```

```
    private int width;
```

```
    public int Width
```

```
{
```

```
        public get { return this.width; }
```

```
        public set { this.width = value; }
```

```
    }
```

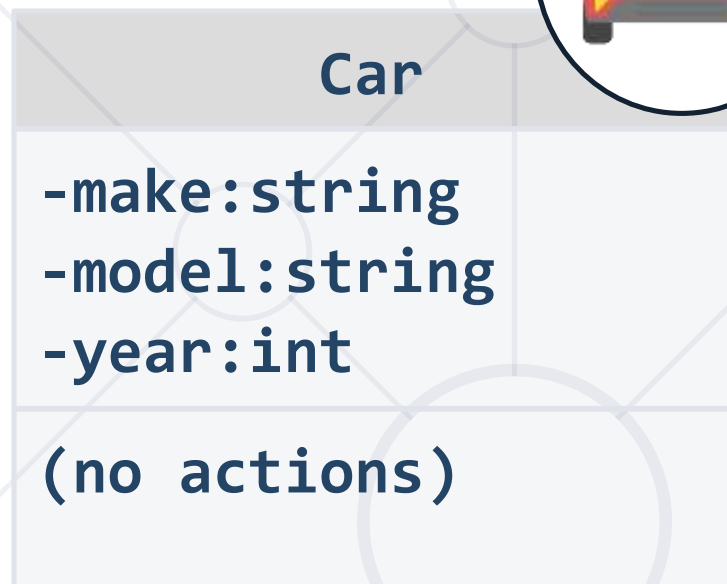
```
}
```

Полето е
частно
(скрито)

Getter-ът дава
достъп до полето

Setter-ът позволява промяна
на полето

- Създайте клас **Car**



```
private string make;
private string model;
private int year;
public string Make
{
    get { return this.make; }
    set { this.make = value; }
}
// TODO: Balance and Year Getter & Setter
```

```
public string Brand { get; private set; }  
public string Make { get; set; }  
public string BrandAndMake  
{  
    get => Brand + " " + Make;  
}
```




Дефиниране на поведение на класа

- 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: Разширение на класа Car

- Създайте клас **Car**

Car
<ul style="list-style-type: none">-make:string-model:string-year:int-fuelQuantity:double-fuelConsumption:double
<ul style="list-style-type: none">+Drive(double distance):void+WhoAmI():string



Решение: Разширение на класа Car (1)

```
// TODO: Get the other fields from previous problem  
private double fuelQuantity;  
private double fuelConsumption;  
// TODO: Get the other properties from previous problem  
public double FuelQuantity {  
    get { return this.fuelQuantity; }  
    set { this.fuelQuantity = value; }}  
public double FuelConsumption {  
    get { return this.fuelConsumption; }  
    set { this.fuelConsumption = value; }}
```

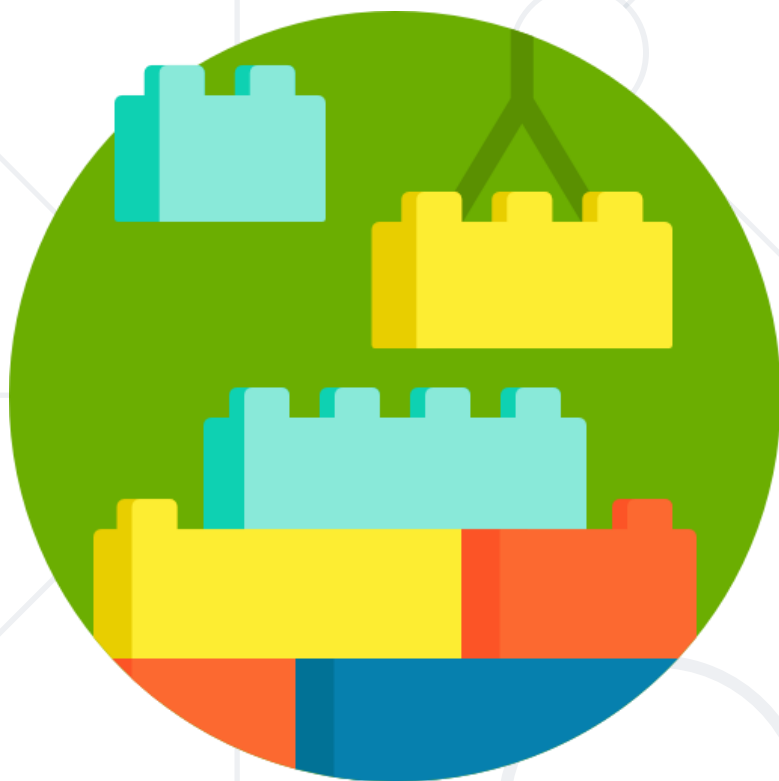
Решение: Разширение на класа Car (2)

```
public void Drive(double distance)
{
    bool canContinue = this.FuelQuantity -
        (distance * this.FuelConsumption) >= 0;

    if (canContinue)
    {
        this.FuelQuantity -= distance * this.FuelConsumption;
    }
    else
    {
        Console.WriteLine("Not enough fuel to perform this trip!");
    }
}
```

Решение: Разширение на класа Car (3)

```
public string WhoAmI()
{
    StringBuilder sb = new StringBuilder();
    sb.AppendLine($"Make: {this.Make}");
    sb.AppendLine($"Model: {this.Model}");
    sb.AppendLine($"Year: {this.Year}");
    sb.Append($"Fuel: {this.FuelQuantity:F2}L");
    return sb.ToString();
}
```



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()
    {
        var figure = new Rectangle();
    }
}
```


- Конструкторите **задават първоначалното състояние на обекта**

```
public class Rectangle {  
    int width;  
    int height;  
    string color;  
    public Rectangle(int width, int height, string color)  
    {  
        this.width = width;  
        this.height = height;  
        this.color= color;  
    }  
}
```

Първоначално състояние на обекта (2)

```
public class Rectangle {  
    int width;  
    int height;  
    private int[] sections;  
  
    public Rectangle(int width, int height, string color)  
    {  
        this.width = width;  
        this.height = height;  
        this.sections= new int[(width * height)/2];  
    }  
}
```

- Един клас може да има множество конструктори

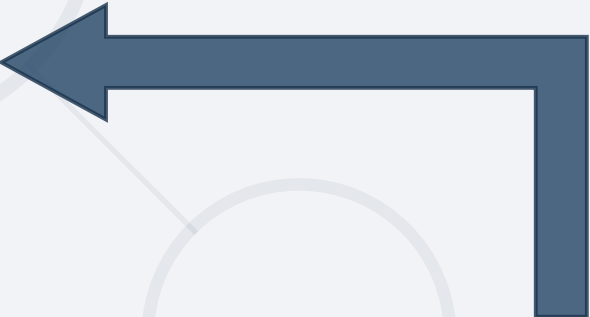
```
public class Rectangle {  
    private string color;  
  
    public Rectangle()  
    {  
        this.color = "white";  
    }  
  
    public Rectangle(string color)  
    {  
        this.color = color;  
    }  
}
```

Конструктор **без**
параметри

Конструктор **със**
параметри

- Един конструктор може да извика другия

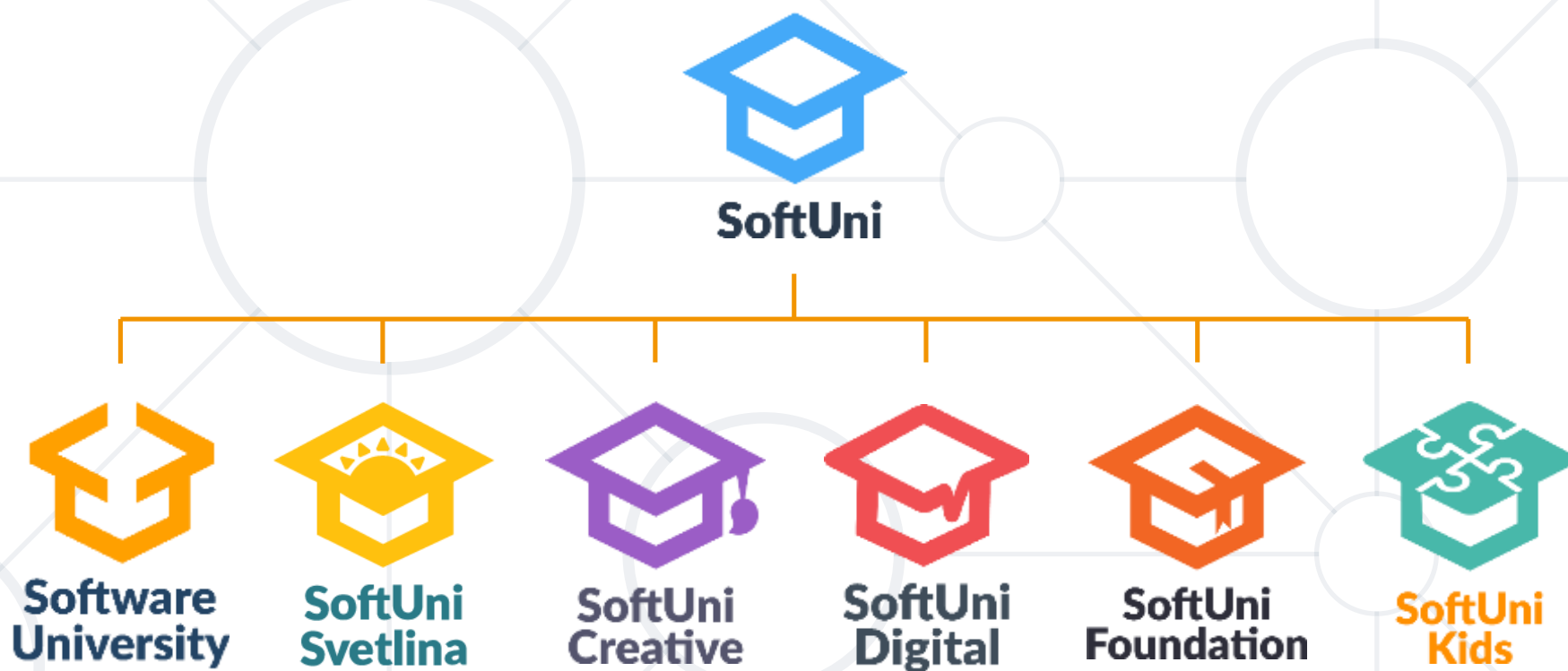
```
public class Person {  
    private string name;  
    private int age;  
    public Person()  
    {  
        this.age = 18;  
    }  
    public Person(string name) : this()  
    {  
        this.name = name;  
    }  
}
```



Извиква
конструктора по
подразбиране

- Класовете дефинират **структура** за обектите
- Обектите са **инстанции на дадения клас**
- NET Core предоставя **хиляди готови за използване класове**
- **Класовете** задават структура за **описание** и **създаване** на обекти
- 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 **защитено авторско съдържание**
- Нерегламентирано копиране, разпространение или използване е незаконно
- © СофтУни – <https://softuni.org>
- © Софтуерен университет – <https://softuni.bg>

