



Национална програма
"Обучение за ИТ умения и кариера"
<https://it-kariera.mon.bg>

Министерството на
образованието и науката
<https://www.mon.bg>



Методи

Описване на поведението на класа

Съдържание

1. Memoгу
2. Getter и Setter memoгу

Елементи на класа

- Клас се дефинира чрез състояние и поведение
- **Полетата** съхраняват състоянието
- **Методите** описват поведението

```
class Dice {  
    int sides;  
    string type;
```

Полеता

```
    void Roll(){ ... }  
}
```

Метод

Мемогу

Те са **изпълним код** (алгоритъм), който променя състоянието

```
class Dice {  
    public int sides;  
    private Random rnd = new Random();  
    public int Roll()  
    {  
        int rollResult = rnd.Next(1, this.sides + 1);  
        return rollResult;  
    }  
}
```

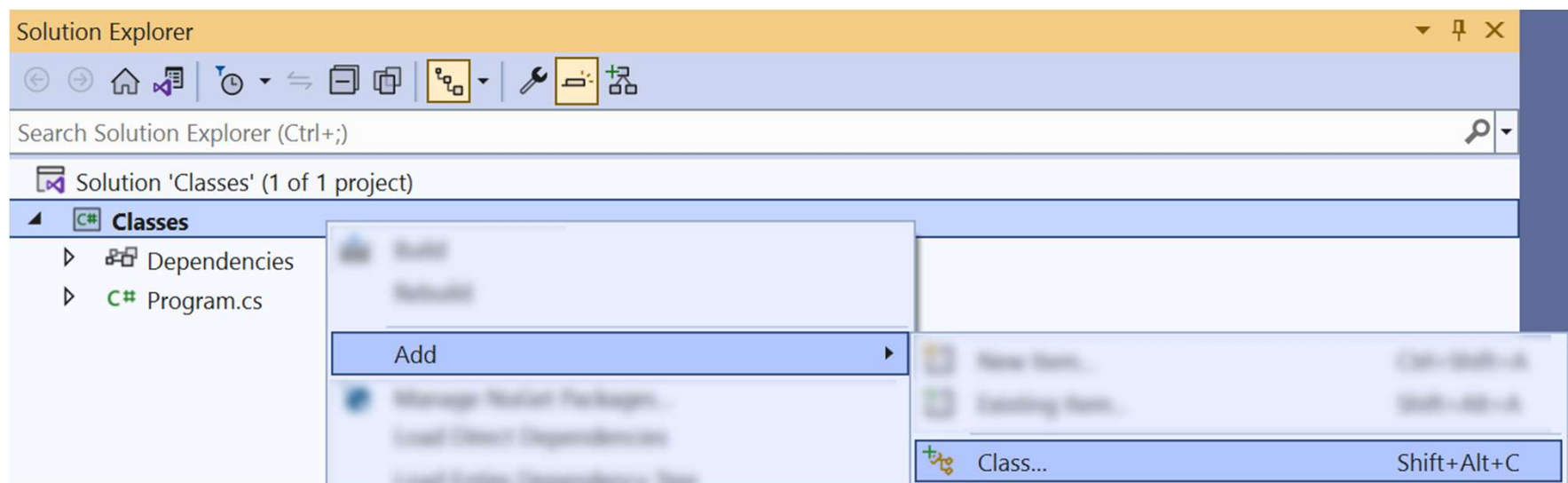
this сочи към
тази инстанция

Задача: Дефинирайте клас Person

Задача: Дефинирайте клас **Person**, като за него павете информация за името и възрастта на човек и реализирайте единствено действието **IntroduceYourself()**, което отпечатва представяне на човека. След това създайте и използвайте обект от класа **Person**.

Решение: Дефинирайте клас Person [1/4]

- Нека първо да създадем файл за този клас:
[Project] → [Add Class] или десен бутон върху
проекта [Add] → [New Item] → [Class]



- Внимавайте с именуването на класа

Решение: Дефинирайте клас Person [2/4]

```
class Person {  
    private string name;  
    private int age;  
    public String Name { // реализираме свойство Name  
        get { return name; }  
        set { name = value; }  
    }  
    public int Age { // реализираме свойство Age  
        get { return age; }  
        set { age = value; }  
    }  
    public void IntroduceYourself() {  
        Console.WriteLine("Здравейте! Аз съм {0} и съм на {1} години.", name, age);  
    }  
}
```

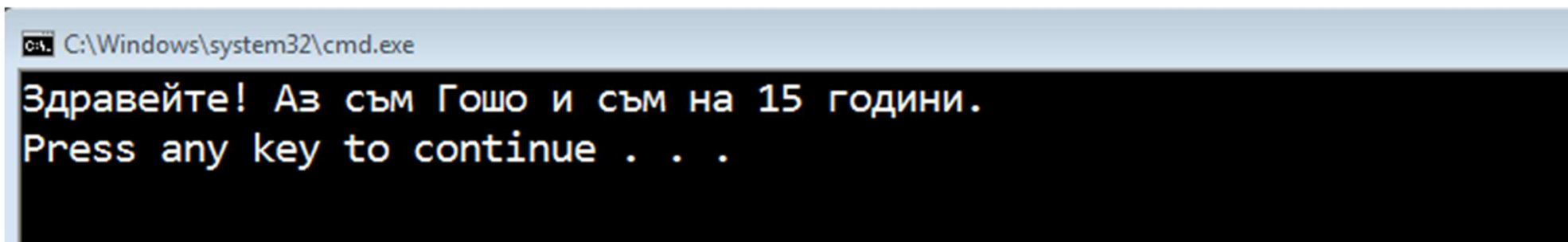
Решение: Дефинирайте клас Person [3/4]

Сега е време да използваме класа и да направим обект в [Main](#) метода ни в [Program.cs](#)

```
static void Main(string[] args) {  
    Person firstPerson = new Person();  
    firstPerson.Name = "Гошо";  
    firstPerson.Age = 15;  
  
    firstPerson.IntroduceYourself();  
}
```


Решение: Дефинирайте клас Person [4/4]

- Ако сте работили правилно ще получите:

A screenshot of a Windows command prompt window. The title bar at the top shows the path 'C:\Windows\system32\cmd.exe'. The command prompt itself has a black background with white text. It displays two lines of output: 'Здравейте! Аз съм Гошо и съм на 15 години.' followed by 'Press any key to continue . . .' on the next line.

```
C:\Windows\system32\cmd.exe
Здравейте! Аз съм Гошо и съм на 15 години.
Press any key to continue . . .
```

- Аналогично създайте `secondPerson` и `thirdPerson` и извикайте `IntroduceYourself` и за тях

Задача: Getter-и и Setter-и

- Създайте клас BankAccount

- == private

BankAccount

-id:int

Връщане на num

-balance:double

+Id:int

+Balance:double

+Deposit(double amount):void

+Withdraw(double amount):void

+ == public

```
public static void Main()
{
    BankAccount acc = new BankAccount();

    acc.ID = 1;
    acc.Deposit(15);
    acc.Withdraw(5);

    Console.WriteLine(acc.ToString());
}
```

Предефинирайте
toString()

Pewehue: Getter-u u Setter-u

```
private double balance;
public void Deposit(double amount)
{
    this.balance += amount;
}
public void Withdraw(double amount)
{
    this.balance -= amount;
}
public override string ToString()
{
    return $"Account {this.id}, balance {this.balance}";
}
```

Задача за напреднали: Тестов клиент

- Създайте тестов клиент за тестване на класа **BankAccount**
- Поддържани команди:
 - **Create** {Id}
 - **Deposit** {Id} {Amount}
 - **Withdraw** {Id} {Amount}
 - **Print** {Id}
 - **End**

```
Create 1
```

```
Create 1
```

```
Deposit 1 20
```

```
Withdraw 1 30
```

```
Withdraw 1 10
```

```
Print 1
```

```
End
```

Съществуваща
сметка

Недостатъчен
баланс

Account already exists

Insufficient balance

Account ID1, balance 10.00

.2f

Решение: Тестов клиент [1/2]

```
var accounts = new Dictionary<int, BankAccount>();
string command;
while ((command = Console.ReadLine()) != "End")
{
    var cmdArgs = command.Split();
    var cmdType = cmdArgs[0];
    switch (cmdType)
    {
        case "Create": Create(cmdArgs, accounts); break;
        case "Deposit": Deposit(cmdArgs, accounts); break;
        case "Withdraw": Withdraw(cmdArgs, accounts); break;
        case "Print": Print(cmdArgs, accounts); break;
    }
}
```

Решение: Тестов клиент [2/2]

```
// създаване на сметката  
var id = int.Parse(cmdArgs[1]);  
if (accounts.ContainsKey(id))  
    Console.WriteLine("Account already exists");  
else  
{  
    var acc = new BankAccount();  
    acc.ID = id;  
    accounts.Add(id, acc);  
}  
// TODO: разписване на останалите команди...
```

Какво научихме днес?

- **Методите** описват поведението на обектите, може да променят състоянието на обектите и/или го достъпват и анализират
- **Getter** и **Setter** методите служат за достъп до и промяна на полетата на обекта



Национална програма
"Обучение за ИТ умения и кариера"
<https://it-kariera.mon.bg>

Министерството на
образованието и науката
<https://www.mon.bg>



**SoftUni
Foundation**



Документът е разработен за нуждите на Национална програма "Обучение за ИТ умения и кариера" на Министерството на образованието и науката (МОН), базиран е на учебно съдържание и методика, предоставени от фондация "Софтуерен университет" и се разпространява под свободен лиценз CC-BY-NC-SA (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).