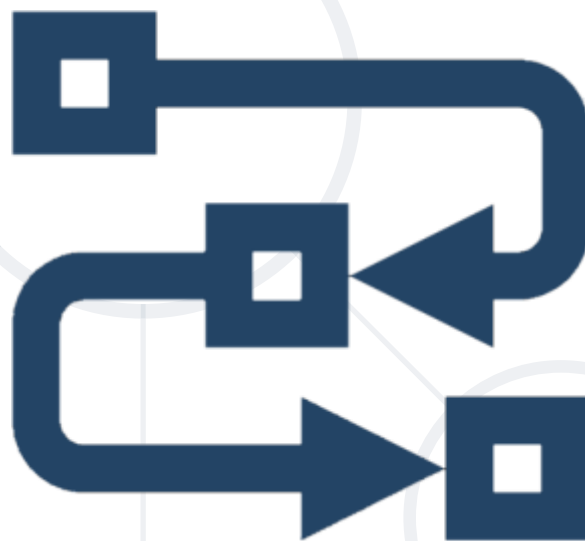


Методи

Дефиниране и използване на методи



SoftUni Team
Technical Trainers



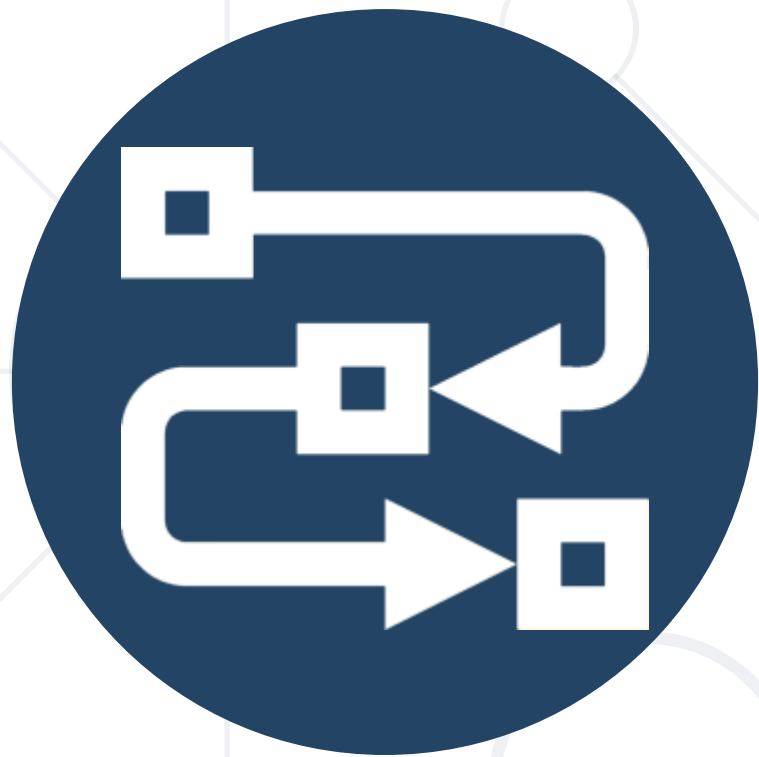
SoftUni

Software University

<https://softuni.bg>

1. Какво е метод?
2. Деклариране и извикване на методи
3. Методи с параметри
4. Стойностни и референтни типове
5. Връщане на стойности от методи
6. Варианти на методи (overloading methods)
7. Ред на изпълнение в програмата
8. Именуване и най-добри практики





Какво е метод?

Какво е метод?

- **Именуван блок от код**, който може в определен момент да бъде извикан
- Пример за дефиниция на метод:

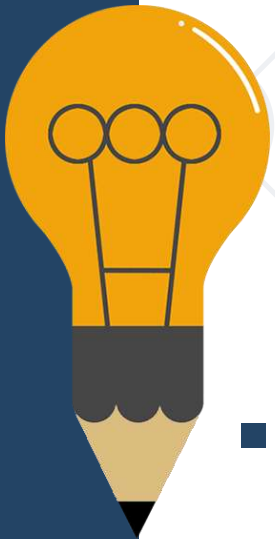
```
static void PrintHelloWorld()  
{  
    Console.WriteLine("Hello World");  
}
```

Името на метода е
PrintHelloWorld

Тялото на
метода винаги
се огражда с
{ }

- Методът може да бъде извикан няколко пъти поред:

```
PrintHelloWorld();  
PrintHelloWorld();
```



Защо използваме методи?



- Методите правят **поддръжката** на кода по-лесна
- Можем да разделяме големи задачи на по-малки части
 - По-добра **организация** на кода
 - По-добра **четимост**
- Избягваме **повторението на код**
- Можем да **преизползваме** код
 - Можем да използваме един метод няколко пъти

Метод от тип Void

- Изпълнява кода между скобите ({})
- Не връща никакъв резултат

```
static void PrintHello()  
{  
    Console.WriteLine("Hello");  
}
```

Отпечатва
"Hello" на
конзолата

```
static void Main()  
{  
    PrintHello();  
}
```

Main() е пример за
метод от тип Void



Деклариране и извикване на методи

Деклариране на методи


Тип, който
връща

Име на метода

Параметри

```
static void PrintText(string text)
{
    Console.WriteLine(text);
}
```

Тяло на
метода

- 
- Методите се декларират **вътре в класа**
 - Променливите, декларирани в метода, са **локални**

- Методите първо се **декларират**, след това се **извикват**

```
static void PrintHeader()  
{  
    Console.WriteLine("-----");  
}
```

Декларация
на метода

- Методите** могат да бъдат **извикани** чрез **името** + **()**:

```
static void Main()  
{  
    PrintHeader();  
}
```

Извикване
на метода

- Методът може да бъде извикан от:

- **Главния метод (Main)**

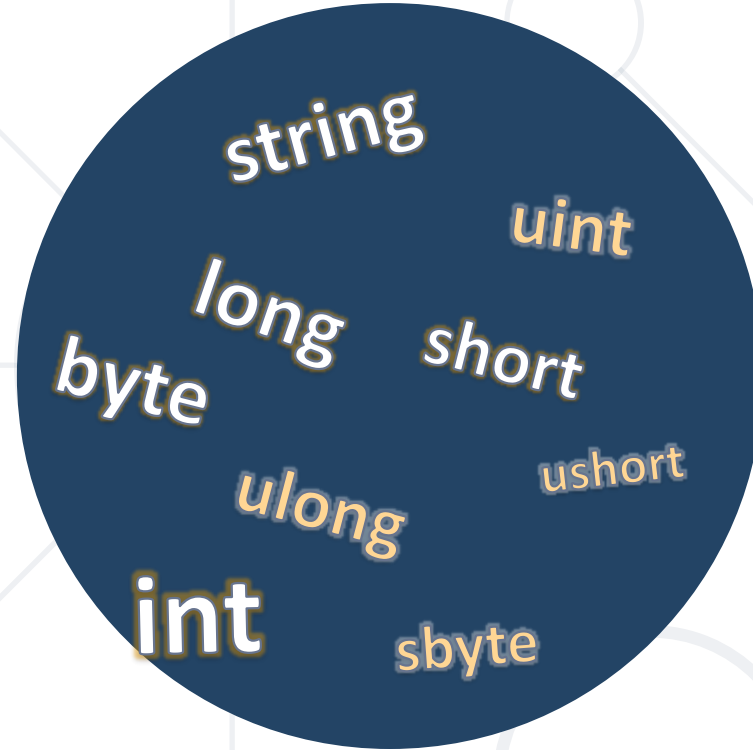
```
static void Main()  
{  
    PrintHeader();  
}
```

- **Друг метод**

```
static void PrintHeader()  
{  
    PrintHeaderTop();  
    PrintHeaderBottom();  
}
```

- **Тялото на същия метод - рекурсия**

```
static void Crash()  
{ Crash(); }
```



Методи с параметри

- **Параметрите** на метода могат да бъдат от **различен тип**.

```
static void PrintNumbers(int start, int end)
{
    for (int i = start; i <= end; i++)
        Console.Write("{0} ", i);
}
```

Приема параметри
start и **end** от тип int

- Извикваме метода с конкретни стойности (**аргументи**)

```
static void Main()
{
    PrintNumbers(5, 10);
}
```

Подаваме аргументите
при извикване на метода

Методи с параметри (2)

- Може да се подават **няколко** параметъра
- Може да се подават параметри от **различен тип**
- Всеки параметър има **име** и **тип**

Няколко параметъра
от различен тип

Тип
на параметъра

Име
на параметъра

```
static void PrintStudent(string name, int age, double grade)
{
    Console.WriteLine("Student: {0}; Age: {1}, Grade: {2}",
        name, age, grade);
}
```

- Методи с кратко тяло може да се дефинират с оператора **=>**:

```
static int Sum(int a, int b) => a + b;
```

- Този синтаксис е идентичен с:

```
static int Sum(int a, int b)
{
    return a + b;
}
```

- Друг пример:

```
static void Print(int x) => Console.WriteLine(x);
```

- Напишете метод, който получава **оценка** между 2.00 и 6.00 и отпечатва **съответната оценка с думи**
 - 2.00 - 2.99 - "Fail"
 - 3.00 - 3.49 - "Poor"
 - 3.50 - 4.49 - "Good"
 - 4.50 - 5.49 - "Very good"
 - 5.50 - 6.00 - "Excellent"

3.33



Poor

4.50



Very good

2.99



Fail

```
static void Main() =>
    PrintInWords(double.Parse(Console.ReadLine()));

private static void PrintInWords(double grade)
{
    string gradeInWords = string.Empty;
    if (grade >= 2 && grade <= 2.99)
        gradeInWords = "Fail";
    // TODO: continue with the rest
    Console.WriteLine(gradeInWords);
}
```


Задача: Знак на цяло число

- Напишете метод, който отпечатва дали дадено цяло число **n** е **положително**, **отрицателно** или **0** :

2



The number 2 is positive.

-5



The number -5 is negative.

0



The number 0 is zero.

Решение: Знак на цяло число

```
static void Main() =>
    PrintSign(int.Parse(Console.ReadLine()));

static void PrintSign(int number)
{
    if (number > 0)
        Console.WriteLine("The number {0} is positive", number);
    else if (number < 0)
        Console.WriteLine("The number {0} is negative.", number);
    else
        Console.WriteLine("The number {0} is zero.", number);
}
```

Проверете решението си тук: <https://judge.softuni.bg/Contests/Practice/Index/3160#1>

- Параметрите може да имат **стойност по подразбиране**:

```
static void PrintNumbers(int start = 0, int end = 100)
{
    for (int i = start; i <= end; i++)
    {
        Console.Write("{0} ", i);
    }
}
```

Стойност по
подразбиране
(default)

- Методът по-горе може да бъде извикан по множество начини:

```
PrintNumbers(5, 10);
```

```
PrintNumbers(end: 40, start: 35);
```

```
PrintNumbers(15);
```

```
PrintNumbers();
```

Параметрите могат да бъдат
пропуснати при извикване на метода

Задача: Отпечатване на триъгълник

- Създайте метод, който да отпечата триъгълник, както е показано в следните примери:

3



```
1
1 2
1 2 3
1 2
1
```

4



```
1
1 2
1 2 3
1 2 3 4
1 2 3
1 2
1
```

Решение: Отпечатване на триъгълник (1)

- Създайте метод, който **отпечатва един ред**, съдържащ числата от **дадено начало (start)** до **даден край (end)**:

```
static void PrintLine(int start, int end)
{
    for (int i = start; i <= end; i++)
    {
        Console.Write(i + " ");
    }
    Console.WriteLine();
}
```

Решението
продължава на
следващия слайд

Решение: Отпечатване на триъгълник (2)

- Създайте метод, който отпечатва **първата половина (от 1 до n)** и след това **втората половина (от n-1 до 1)** от триъгълника:

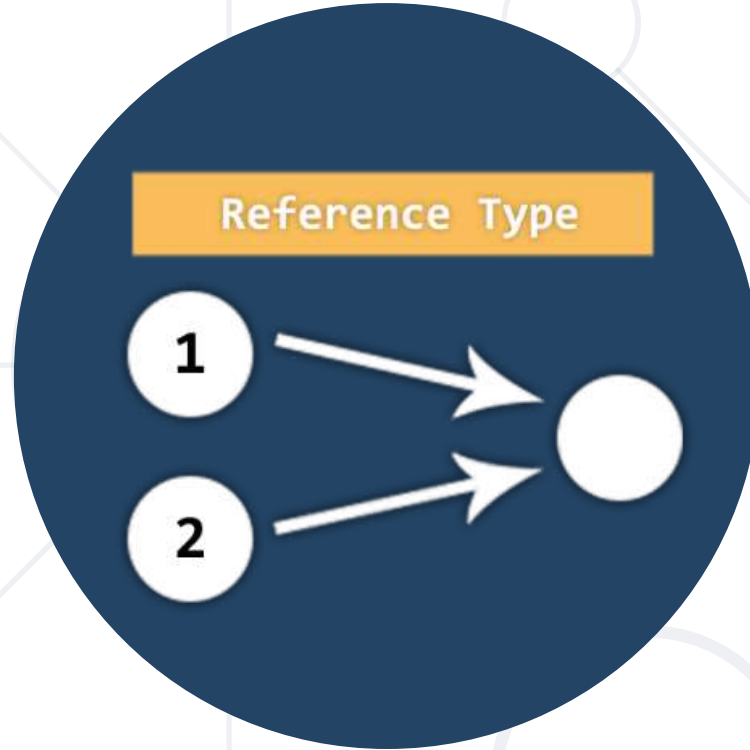
```
static void PrintTriangle(int n)
{
    for (int line = 1; line <= n; line++)
        PrintLine(1, line);

    for (int line = n - 1; line >= 1; line--)
        PrintLine(1, line);
}
```

Метод с
параметър n

Редове 1...n

Редове n-1...1

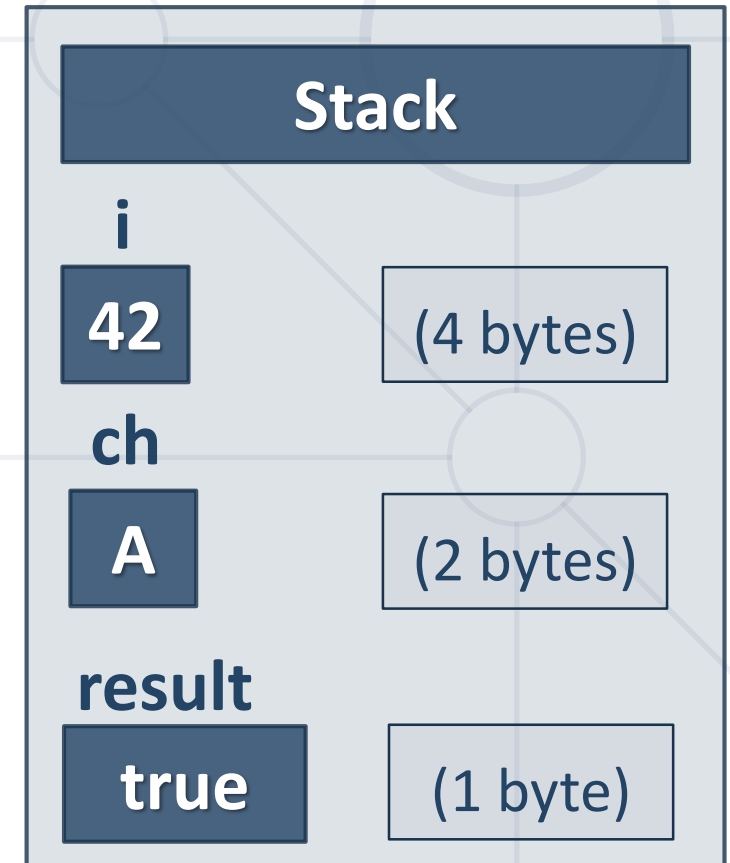


Стек и динамична памет


Стойностни типове

- Променливите от **стойностен тип** пазят директно своята стойност в стека
 - `int`, `float`, `double`, `bool`, `char`, `BigInteger`, ...
- Всяка променлива има свое **копие** на **стойността**

```
int i = 42;  
char ch = 'A';  
bool result = true;
```

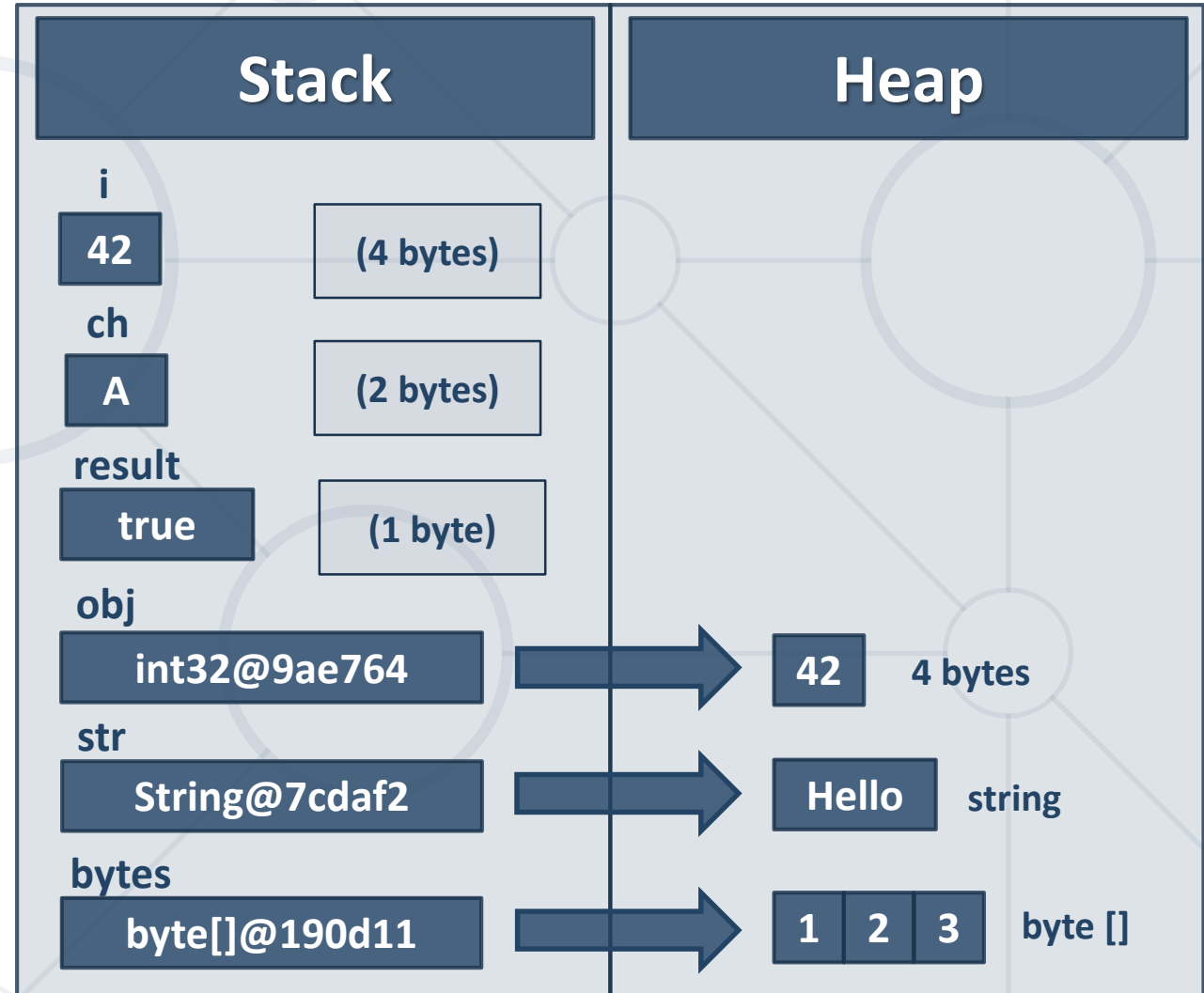


Референтни типове

- 
- Променливите от **референтен тип** (**string**, **int[]**, **char[]**, **string[]**, **Random**) пазят адрес от динамичната памет (heap), където е записана стойността им.
 - Две променливи от референтен тип може да **реферират** към **един и същ обект**
 - Операции върху която и да е от двете променливи достъпват/модифицират **едни и същи данни**

Разлика между стойности и референтни типове

```
int i = 42;  
char ch = 'A';  
bool result = true;  
object obj = 42;  
string str = "Hello";  
byte[] bytes = { 1, 2, 3 };
```



Пример: Стойностни типове

```
public static void Main() {  
    int number = 5;  
    Increment(number, 15);  
    Console.WriteLine(number);  
}
```

number == 5

```
public static void Increment(int num, int value)  
{  
    num += value;  
}
```

num == 20

Пример: Референтни типове

```
public static void Main() {  
    int[] nums = { 5 };  
    Increment(nums, 15);  
    Console.WriteLine(nums[0]);  
}
```


nums[0] == 20

```
public static void Increment(int[] nums, int value)  
{  
    nums[0] += value;  
}
```

nums[0] == 20


Стойностни vs. Референтни типове

pass by reference

cup = 

fillCup()

pass by value

cup = 

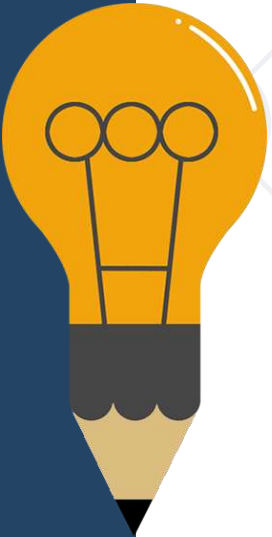
fillCup()



Връщане на стойности в метода

Ключовата дума return

- Ключовата дума **return** незабавно прекратява изпълнението на метода.
- Връща се конкретна стойност



```
static string ReadFullName()
{
    string firstName = Console.ReadLine();
    string lastName = Console.ReadLine();
    return firstName + " " + lastName;
}
```

Връща
стринг

- Void методите могат да бъдат **прекратени** само с ключовата дума **return** (те не връщат стойност)

Употреба на върнатите стойности

- Върнатите стойности мога да бъдат:

- **Присвоени** на променлива:

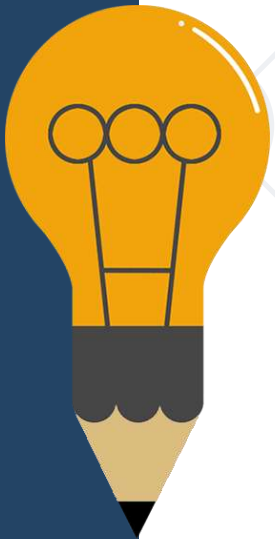
```
int max = GetMax(5, 10);
```

- **Използвани** в израз:

```
decimal total = GetPrice() * quantity * 1.20m;
```

- **Подадени** на друг метод:

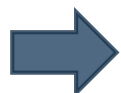
```
int age = int.Parse(Console.ReadLine());
```



Задача: Лице на правоъгълник

- Създайте метод, който връща **лицето на правоъгълник** при зададени **дължина** и **ширина**.

3
4



12

6
8



48

5
10



50

7
8



56

Решение: Лице на правоъгълник

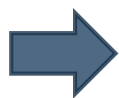
```
static void Main()
{
    double width = double.Parse(Console.ReadLine());
    double height = double.Parse(Console.ReadLine());
    double area = CalcRectangleArea(width, height);
    Console.WriteLine(area);
}
```

```
static double CalcRectangleArea(double width, double height)
{
    return width * height;
}
```

Задача: Повторение на стринг

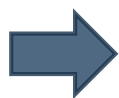
- Напишете метод, който получава **стринг** и цяло число **n**, което означава броя повторения.
 - Методът трябва да връща **нов стринг**, който представлява **въведения стринг**, повторен **n пъти**.

abc
3



abccabccabcc

String
2



StringString

Решение: Повторение на стринг (1)

```
static void Main()
{
    string inputStr = Console.ReadLine();
    int count = int.Parse(Console.ReadLine());

    string result = RepeatString(inputStr, count);
    Console.WriteLine(result);
}
```

Решение: Повторение на стринг (2)

```
private static string RepeatString(string str, int count)
{
    StringBuilder result = new StringBuilder();
    for (int i = 0; i < count; i++)
    {
        result.Append(str);
    }
    return result.ToString();
}
```

- Създайте метод, който изчислява и връща стойността на дадено число (база), повдигнато на определена степен:

$$2^8 \Rightarrow 256$$

$$3^4 \Rightarrow 81$$

```
static double MathPower(double number, int power)
{
    double result = 1;
    for (int i = 0; i < power; i++)
        result *= number;
    return result;
}
```

Проверете решението си тук: <https://judge.softuni.bg/Contests/Practice/Index/3160#6>



Варианти на методи (Overloading Methods)

- Комбинацията от **името** и **параметрите** на метода се нарича **сигнатура**.

```
static void Print(string text)
{
    Console.WriteLine(text);
}
```

Сигнатура
на метода

- Сигнатурата **различава** методи с еднакви имена
- Когато методи с **еднакви имена** имат **различна сигнатура**, методът се нарича "**overloading**".

Варианти на методи (Overloading Methods)

```
static void Print(string text)
{
    Console.WriteLine(text);
}
```

```
static void Print(int number)
{
    Console.WriteLine(number);
}
```

```
static void Print(string text, int number)
{
    Console.WriteLine(text + ' ' + number);
}
```

Различни
сигнатури
на един метод

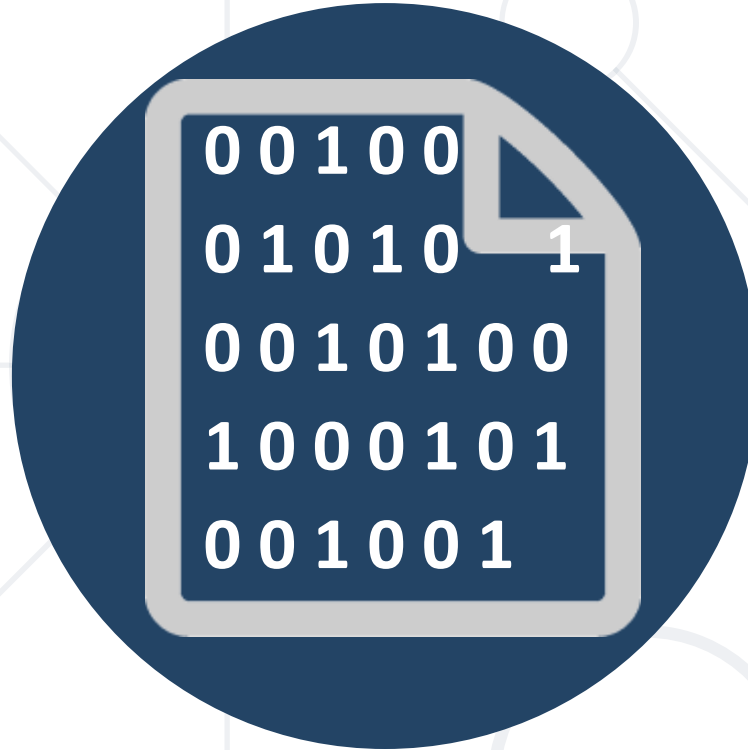
- Типът на върнатата от метода стойност **не е част** от неговата сигнатура

```
static void Print(string text)
{
    Console.WriteLine(text);
}

static string Print(string text)
{
    return text;
}
```

Грешка при
компилация

- Как може компилаторът да знае **кой метод да извика?**



Ред на изпълнение в програмата

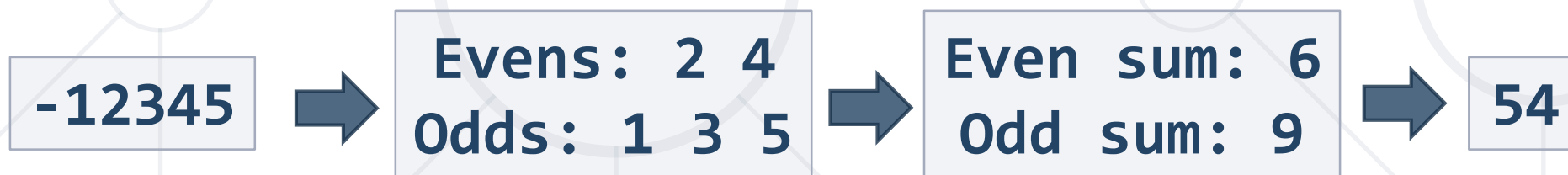
- Програмата **продължава** след като **изпълнението на метода приключи**:

```
static void Main()
{
    Console.WriteLine("before method executes");
    PrintLogo();
    Console.WriteLine("after method executes");
}

static void PrintLogo()
{
    Console.WriteLine("Company Logo");
    Console.WriteLine("http://www.companywebsite.com");
}
```

Задача: Произведение от четни и нечетни цифри

- Създайте програма, която **умножава сумата** на **всички четни цифри** от дадено число по **сумата** на **всички нечетни цифри** от същото число:



- Създайте метод **GetSumOfEvenDigits()**
- Създайте метод **GetSumOfOddDigits()**
- Създайте метод **GetMultipleOfEvensAndOdds()**
- Можете да ползвате **Math.Abs()** за отрицателни числа

Решение: Произведение от четни и нечетни цифри (1)

```
static int GetSumOfEvenDigits(int number)
{
    int evenSum = 0;
    while (number >= 1)
    {
        int digit = number % 10;
        if (digit % 2 == 0)
            evenSum += digit;
        number /= 10;
    }
    return evenSum;
}
```

```
static int GetSumOfOddDigits(int number)
{
    // Use the same logic ...
}
```

Решение: Произведение от четни и нечетни цифри (2)

```
static int GetMultipliedEvensAndOdds(int number)
{
    int evenSum = GetSumOfEvenDigits(number);
    int oddSum = GetSumOfOddDigits(number);
    int result = evenSum * oddSum;
    return result;
}
```

```
static void Main(string[] args)
{
    int num = int.Parse(Console.ReadLine());
    int number = Math.Abs(num);
    int result = GetMultipliedEvensAndOdds(number);
    Console.WriteLine(result);
}
```

Проверете решението си тук: <https://judge.softuni.bg/Contests/Practice/Index/3160#8>



Именуване на методи

Методи за именуване

- Задавайте **смислени** имена на методи - използвайте **глаголи**
- Имената на методите трябва да отговарят на въпроса:
 - **Какво прави този метод?**



FindStudent, LoadReport, Save

- Ако не можете да се сетите за добро име за метод, помислете за нещо, което **отличава метода** от останалите



Method1, DoSomething, HandleStuff, SampleMethod, DirtyHack



Именуване на параметрите на метод

- Предпочитана форма: [съществително име] или [прилагателно име] + [съществително име]
- Трябва да бъде в **camelCase**
- Трябва да бъде **смислено**



```
firstName, report, speedKmH,  
usersList, fontSizeInPixels, font
```


- Всеки метод трябва да изпълнява **една** добре дефинирана задача
 - Името на метода трябва винаги **да описва неговата задача** по **ясен** начин
- **Избягвайте** методи, **по-дълги от един екран**
 - **Можете да ги разделите** на няколко по-кратки метода

```
private static void PrintReceipt()  
{  
    PrintHeader();  
    PrintBody();  
    PrintFooter();  
}
```


**Описателни и
лесни за тестване
методи**

- Уверете се, че ползвате правилна **индентация**

```
static void Main()  
{  
    ➡ // some code...  
    ➡ // some more code...  
}
```



```
static void Main()  
    ➡ {  
        ➡ // some code...  
➡ // some more code...  
}
```



- Оставяйте **празен ред** между **методи**, след **цикли** и **условни конструкции**
- Използвайте **къдрави скоби { }** за тялото на циклите и условните конструкции
- Избягвайте **дълги редове** и **сложни изрази**

- Можем да разделяме дълги програми на по-кратки **методи**, които решават конкретни задачи
- Методите се състоят от **декларация** и **тяло**
- Методите се извикват с тяхното **име** + **()**
- Методите могат да приемат **параметри**
- Методите могат да **връщат стойност** или **да не връщат нищо (void)**

Въпроси?



SoftUni



Software
University



SoftUni
Svetlina



SoftUni
Creative



SoftUni
Digital



SoftUni
Foundation



SoftUni
Kids

- 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>

