Invoke – извикване на метод

Метода е именувано парче код. Име което кара програмата да изпълнява някакъв код. Чрез метод можем да преизползваме код по-късно.

Име на метод (параметри) – static double GetSquare(double num)

{

Return num \* num; - return връща стойността на num \* num в променлива double

}

Методите трябва да са в клас (в C#).

Стойностите вътре в метода са local!!!! Ако дефинираме променлива, тя си остава там.

Скобите разделят метода от променлива! PrintHeader() – метод

Invoke става само с написване на името на метода.

Метода трябва да се извиква вътре в мейна.

Може самия метод да извика себе си – recursion. StackOverflowException – препълване на стека.

Метода трябва да се направи ИЗВЪН мейн метода – преди или след. Засега няма значение дали преди или след, защото компилатора първо чете целия код, създава това, което му трябва и почва да чете.

PrintNumbers(int start, int end) –дефинираме 2 променливи цели числа

**for (int i = start; i <= end; i++)**

**{**

**Console.Write("{0} ", i);**

**}**

След това в мейн метода - PrintNumbers(5, 10)

За изпита – можем да си дежинираме конкретни методи, които да използваме по-късно.

Параметрите могат да бъдат нула, могат да бъдат от различен тип.

Int n = int.Parse(Console.ReadLine());

PrintHeader(n);

Void – не връща стойност, просто изпълнва код

static int PlusOne(int n) - връща някаква стойност!!!!

{

return n + 1;

}

Връща числото плюс едно. След return приключва работа. Освен че break-ва и изпълнява някаква стойност – в случая n+1

Метода може да връща и дата.

Void методите могат да бъдат прекратени с return;

Можем Console.WriteLine(PlusOne(7));

Можем да ползваме метод директно в променлива или израз.

int max = GetMax(5, 10);

Parse е метод от класа int.

WriteLine е метод от класа Console.

Static int Getmax() – връща по-голямото от две числа

Int first=….readline

Int second=….readline

Return Math.Max(first,second);

Math.Pow – същото като задачата за степен

Overload – използваме един и същи метод по няколко различни начина. Създаваме метод, който е преизползваем по различни начини. Примерно метод, който показва много типове данни, да има вариант да покаже и когато няма някой тип данни.

Return type-а не е част от сигнатурата. Тоест overload се ползва само за методи с ЕДИН ТИП ДАННИ!!! Или void или int или string….

Сравняване на string-ове!   
 first.CompareTo(second) >= 0

CompareTo връща числова стойност, >0 ако числото е по-голямо, = 0 ако са равни, < 0 ако е по-малко

Стека е буферното място, където се зарежда кода, за да се компилира. Стека помни точно къде, на кой ред сме.

F9 слага и маха breakpoint. F10 следваща стъпка. F11 следи какво прави даден метод?

Ctrl + . ->показва какво трябва да се импортира. Ползва го при System.Globalization в примера с DateTime.ParseExact.

\*\*\* for цикъл с дата – AddDays(1) накрая

Метода трябва да казва какво прави – чете Read, връща return. Наименоват се с Pascal Case – всички думи почват с главна буква.

Параметрите на методите са наименоват като променливи с camelCase.

Всеки метод трябва да прави точно едно нещо. Ако прави няколко неща – най-добре да разделим метода.

Самодокументиране – името на метода и това, което се случва вътре са ясни от пръв поглед.

Хубаво е един метод да се събира на 1 екран, около 30 реда.