# Methods

You can check your solutions here: <https://judge.softuni.bg/Contests/892/Classes-Constructors-Access-Modifiers-Methods>

# Declaring and Invoking Methods

## Blank Receipt

Create a method that prints a blank cash receipt. The method should invoke three other methods: one for printing the header, one for the body and one for the footer of the receipt.

|  |  |
| --- | --- |
| The header should contain the following text: | CASH RECEIPT  ------------------------------ |
| The body should contain the following text: | Charged to\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Received by\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| And the text for the footer: | ------------------------------  © SoftUni |

### Examples

|  |
| --- |
| **Output** |
| CASH RECEIPT  ------------------------------  Charged to\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Received by\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ------------------------------  © SoftUni |

### Hints

1. First create a method with no parameters for printing the header starting with **static void**. Give it a **meaningful name** like "PrintReceiptHeader" and write the code that it will execute:



1. Do the same for printing the receipt body and footer.
2. Create a **method that will call all three methods** in the necessary order. Again, give it a **meaningful and descriptive name** like "PrintReceipt" and write the code:



1. For printing **"©"** use Unicode **"\u00A9"**
2. **Call** (invoke) the PrintReceipt method from the main.



## Sign of Integer Number

Create a method that prints the sign of an integer number n.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 | The number 2 is positive. |
| -5 | The number -5 is negative. |
| 0 | The number 0 is zero. |

### Hints

1. Create a method with a **descriptive name** like "PrintSign" which should receive **one parameter** of type **int**.



1. Implement the body of the method by handling different cases:
   1. If the number is greater than zero
   2. If the number is less than zero
   3. And if the number is equal to zero
2. Call (invoke) the newly created method from the main.



## Printing Triangle

Create a method for printing triangles as shown below:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 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 |

### Hints

1. After you read the input
2. Start by creating a method **for printing a single line** from a **given start** to a **given end**. Choose a **meaningful name** for it, describing its purpose:



1. Think how you can use it to solve the problem
2. After you spent some time thinking, you should have come to the conclusion that you will need two loops
3. In the first loop you can print the first half of the triangle without the middle line:



1. Next, print the middle line:



1. Lastly, print the rest of the triangle:



# Returning Values and Overloading

## Calculate Triangle Area

Create a method that calculates and **returns** the [area](http://www.mathopenref.com/trianglearea.html) of a triangle by given base and height:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  4 | 6 |

### Hints

1. After reading the input
2. Create a method, but this time **instead** of typing **"static void"** before its name, type **"static double"** as this will make it to **return a value of type double**:



1. **Invoke** the method in the main and **save the return value in a new variable**:



## Greater of Two Values

You are given two values of the same type as input. The values can be of type int, char of string. Create a method GetMax() that returns the greater of the two values:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| int  2  16 | 16 |
| char  a  z | z |
| string  Ivan  Todor | Todor |

### Hints

1. For this method you need to create three methods with the same name and different signatures
2. Create a method which will compare integers:



1. Create a second method with the same name which will compare characters. Follow the logic of the previous method:



1. Lastly you need to create a method to compare strings. This is a bit different as strings don't allow to be compared with the operators > and <



You need to use the method "CompareTo()", which returns an integer value (greater than zero if the compared object is greater, less than zero if the compared object is lesser and zero if the two objects are equal.

1. The last step is to read the input, use appropriate variables and call the GetMax() from your Main():



## Multiply Evens by Odds

Create a program that reads an **integer number** and **multiplies the sum of all its even digits** by **the sum of all its odd digits**:

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 12345 | 54 | 12345 has **2 even digits** - 2 and 4. Even digits has **sum of 6**.  Also it has **3 odd digits** - 1, 3 and 5. Odd digits has **sum of 9**.  **Multiply 6 by 9** and you get **54**. |
| -12345 | 54 |  |

### Hints

1. Create a method with a **name describing its purpose** (like GetMultipleOfEvensAndOdds). The method should have a **single integer parameter** and an **integer return value**. Also the method will call two other methods:



1. Create two other methods each of which will sum either even or odd digits
2. Implement the logic for summing odd digits:



1. Do the same for the method that will sum even digits
2. As you test your solution you may notice that it doesn't work for negative numbers. Following the program execution line by line, find and fix the bug (**hint: you can use Math.Abs()**)

## Max Method

Create a method GetMax(int a, int b), that returns the **largest** of two numbers. Write a program that reads **three numbers** from the console and **prints** the **biggest** of them. Use the GetMax(…) method you just created.

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 1  2  3 | 3 |  | -100  -101  -102 | -100 |

1. **Prime Checker**

Write a Boolean method **IsPrime(n)** that check whether a given integer number **n** is [prime](https://en.wikipedia.org/wiki/Prime_number). Examples:

|  |  |
| --- | --- |
| **n** | **IsPrime(n)** |
| 0 | False |
| 1 | False |
| 2 | True |
| 3 | True |
| 4 | False |
| 5 | True |
| 323 | False |
| 337 | True |
| 6737626471 | True |
| 117342557809 | False |

1. **\* Primes in Given Range**

Write a method that calculates **all prime numbers in given range** and returns them as list of integers:

|  |
| --- |
| static List<int> FindPrimesInRange(startNum, endNum)  {  …  } |

Write a method to **print a list of integers**. Write a program that enters two integer numbers (each at a separate line) and prints all primes in their range, separated by a comma.

### Examples

|  |  |
| --- | --- |
| **Start and End Number** | **Output** |
| 0  10 | 2, 3, 5, 7 |
| 5  11 | 5, 7, 11 |
| 100  200 | 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199 |
| 250  950 | 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947 |
| 100  50 | *(empty list)* |

## Center Point

You are given the coordinates of two points on a [Cartesian coordinate system](https://en.wikipedia.org/wiki/Cartesian_coordinate_system) - X1, Y1, X2 and Y2. **Create a method** that prints the point that is closest to the center of the coordinate system (0, 0) in the format (X, Y). If the points are on a same distance from the center, print only the first one.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  4  -1  2 | (-1, 2) |

## Geometry Calculator

Write a program that can **calculate the area** of **four different geometry figures** - triangle, square, rectangle and circle.

**On the first line** you will get the **figure type**. Next you will get parameters for the chosen figure, **each on a different line**:

* Triangle - side and height
* Square - side
* Rectangle - width and height
* Circle - radius

The output should be rounded to the second digit after the decimal point:

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| triangle  3  6 | 9.00 |
| rectangle  4  5 | 20.00 |

## Master Numbers

A master number is an integer that holds the following properties:

* Is **symmetric** (palindrome), e.g. 5, 77, 282, 14341, 9553559.
* Its **sum of digits is divisible by 7**, e.g. 77, 313, 464, 5225, 37173.
* Holds at least **one even digit**, e.g. 232, 707, 6886, 87578.

Write a program to **print all master numbers** in the range [1…**n**].

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 600 | 232  383  464  545 |  | 5000 | 232  383  464  545  626  696  707  858  1661  2552  3443  4334 |

### Hints

1. Write 3 utility methods:

* IsPalindrome(int num)
* SumOfDigits(int num)
* ContainsEvenDigit(int num)

1. Loop through all numbers in range [1…n] and check every number with the helper methods.

## \* Factorial

Write a program that calculates and prints the n! for any n in the range [1…1000].

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 5 | 120 |  | 100 | 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864000000000000000000000000 |

### Hints

Use the class BigIntegerfrom the built-in .NET library System.Numerics.dll.

1. First add reference to System.Numerics.dll.





1. Import the namespace “System.Numerics”:



1. Use the type BigInteger instead of long or decimal to keep the factorial value:



## Factorial Trailing Zeroes

Create a program that counts the trailing zeroes of the factorial of a given number.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5 | 1 | 5! = 12**0** -> One trailing zero |
| 100 | 24 | 100! = 93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286253697920827223758251185210916864**000000000000000000000000** -> 24 trailing zeroes |

### Hints

1. You may use your solution from the previous problem. Add additional method that counts and returns the number of zeroes a number has.