**Requirements:**

All tasks should be performed by separate methods (task1/task2/task2\_1/...) in the Main class, accept arguments and returns value from the condition.

Implement each additional task in a separate method, and, if possible, reuse the method of the main task.

In the main method, for each task:

* Separately declare all incoming arguments and expected results as variables;
* Call task1/task2/..., save the value to a new variable;
* For each task, output the variables to the console.

Optionally:

• Display in table format (StringBuilder and Format may be useful to you), converted to strings: expected result, actual result, and true/false to match.

**Example:**

Task 0. Write a method that converts a numeric value to a string.

public static void main(String[] args) {  
 int input = 454;  
 String task0Result = *task0*(input);  
 System.*out*.println("Result of task 0: " + task0Result);  
}  
  
public static String task0(int input) {  
 return String.*valueOf*(input);  
}

**Mandatory tasks:**

**Task 1.** Write a method that takes an integer and returns its sum of digits.

For example: task1 (1985) = 1 + 9 + 8 + 5 = 23

**Task 2.** Write a method that takes an empty array of 100 elements and fills it with the first 100 primes.

For example task2 (new int[100]) = {2, 3, 5, 7, 11, 13, ..., 541}

**2.1 (Optional)** Add a method that accepts an arbitrary size of the array, creates it and reuses task2 to solve.

**Task 3.** Write a method that takes an array of integers of arbitrary size and returns the arithmetic mean of these numbers (note that it may not be an integer).

For example task3 ({12, 22, 32, 42, 52, 62}) = (12 + 22 + 32 + 42 + 52 + 62) /6=37.0

**3.1 (Optional)** Add a method for calculating the geometric mean

**3.2 (Optional)** Add the ability to enter numbers from the keyboard using Scanner (System.in))

**Task 4.** Write a method that takes an array of arbitrary length integers and returns sorted in ascending order. Select implementation of one/several/or all optional sorting methods.

For example task4 ({55, 42, 16, 877, 24, 62, 56}) = {16, 24, 42, 55, 56, 62, 877}

Sorting methods:

• Bubble Sort: Easiest. Use no more than 1 intermediate variable. Use double loop for implementation. In a double loop, different (a [i]> a [j]) array elements are compared, and swapped if they are not in ascending order relative to their indices.

• Bogosort: The most meaningless. Use recursion for implementation. The algorithm randomly shuffles the elements of the array, and checks if they are in sorted form, if not, it continues the recursion. Be careful to limit yourself to small arrays as the execution time is factorially difficult.

• Fast sorting: one of the most efficient for discrete finite state machines (computers on transistors and not only, on the contrary, there are quantum sorting algorithms for quantum computers, they are even more efficient). Use recursion for implementation. The algorithm divides the incoming array in half, and is called recursively on its halves. Upon reaching the lower level of recursion, the algorithm compares 2 numbers (at this moment these are arrays of unit length), and swaps them at the higher level, thus obtaining a sorted array of 2 numbers.

**Task 5.** Write a method that takes 2 strings: a substring and a main string. Returns the number of substring inclusions in the main string (ignoring the case, i.e. the uppercase and lowercase letters are considered the same).

For example task5 ("Java", "Java School allows you to try java programming language on practice. Java one of the most popular programming languages. I love Java !!!") = 4

**5.1** Add a method that will not just read, but return an array of indices of the beginning of each of the inclusions

For example task5\_1 ("Java", "Java School allows you to try java programming language on practice. Java one of the most popular programming languages. I love Java !!!") = {0, 30, 69, 128}

**5.2 (Optional)** implement the ability to enter 2 lines from the keyboard using Scanner (System.in)

**Optional:**

Write a method that will generate an arbitrary weather forecast and display it and recommendations on how to dress, depending on the forecast

**Requirements:**

**Optional task 1.** Write a generator method that will create a one-dimensional array representing a specific day and its temperature:

• The array index will be the date (int[] = new int[29])

• The array element must contain int values: temperature int temp from -100 to +100.

• Data should be filled with random numbers

For instance:

optionalTask1 ()

{-8, -5, -3, 2, 3, ...}

1 day, -8 degrees;

2 day, -5 degrees;

...

**Optional task 2.** Create an ENUM, which is an enumeration of weather types (Frost, Cool, Warm, Hot) and a method that accepts an array of temperatures and returns an associated ENUM (switch-case will help you):

It is necessary to take into account the maximum and minimum values ​​of the temperature, to separate which ENUM to use.

Write a method that (the input data is at your discretion) will return an array of 2 strings for outerwear and underwear suitable for the corresponding weather.

For example

optionalTask2 (Frost) = {"Fur coat", "Warm boots"}

or

optionalTask2 (23) = {"T-shirt", "Shorts"}

**Optional task 3.** Create a method that will use the generation method (from optional task 1) and generate a forecast for a given number of days, go through it and display the temperature and recommendations for what to take with you based on the temperature on each day.

public static void optionalTask3(2) {  
//some realisation  
}

Expected console output:

Day 1: -2, Fur Coat + Warm Boots

Day 2: +40, T-shirt + Shorts