**JOBSHEET 5**

**Brute Force and Divide Conquer**



**Name**

Sherly Lutfi Azkiah Sulistyawati

**NIM**

2341720241

**Class**

1I

**Major**

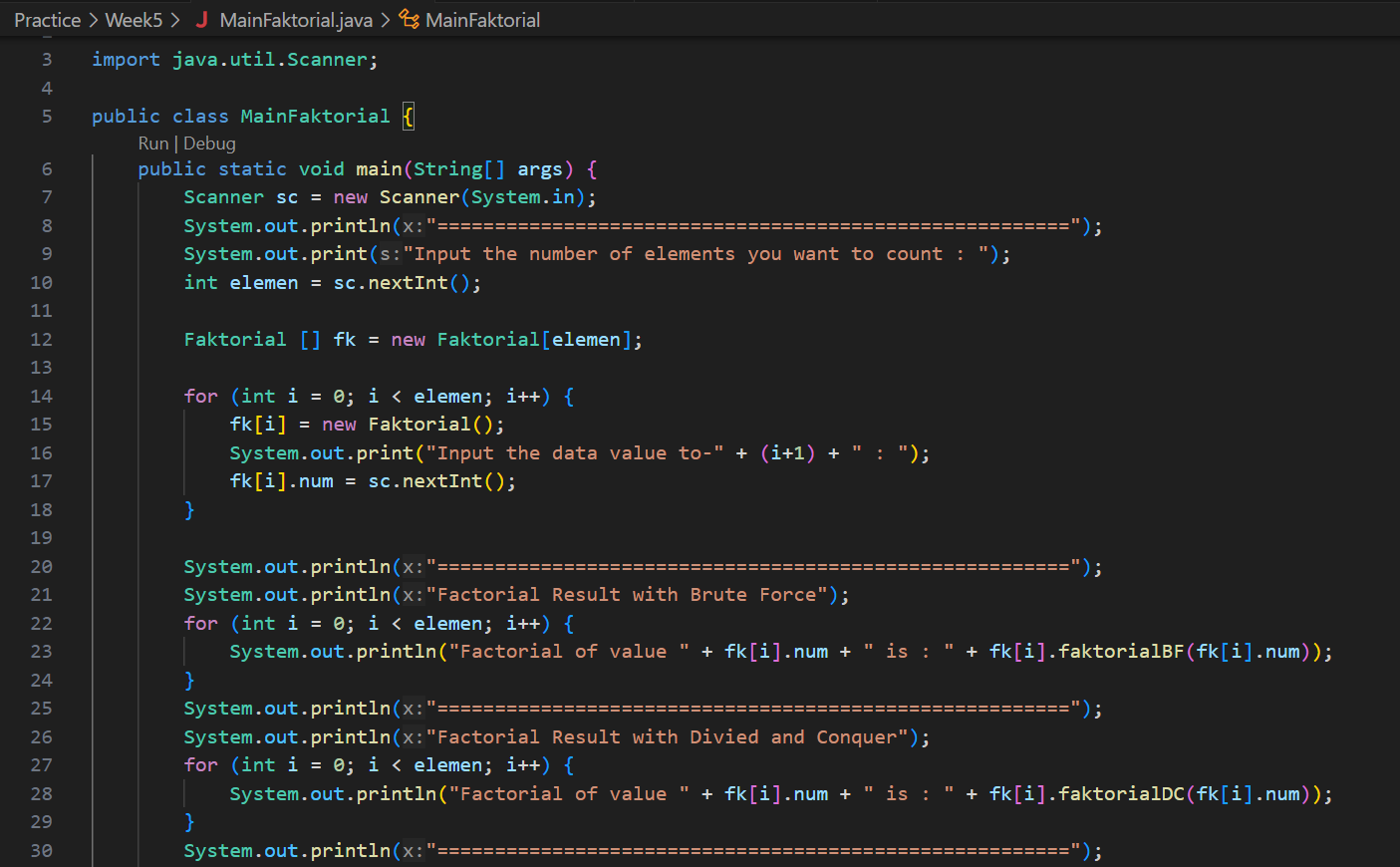
Information Technology

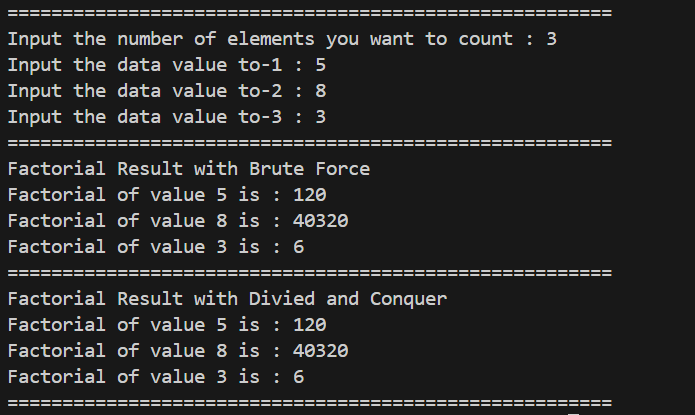
**Study Program**

D4 Informatics Engineering

**Lab Activity 1: Calculating Factorial Values with Brute Force and Divide and Conquer Algorithms**

**Result**

****

****

**Question**

1. Explain the Divide Conquer Algorithm for calculating factorial values!

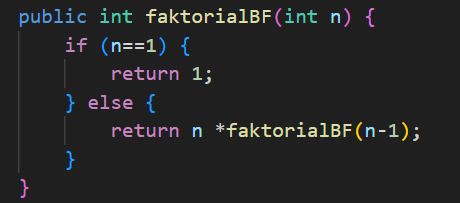
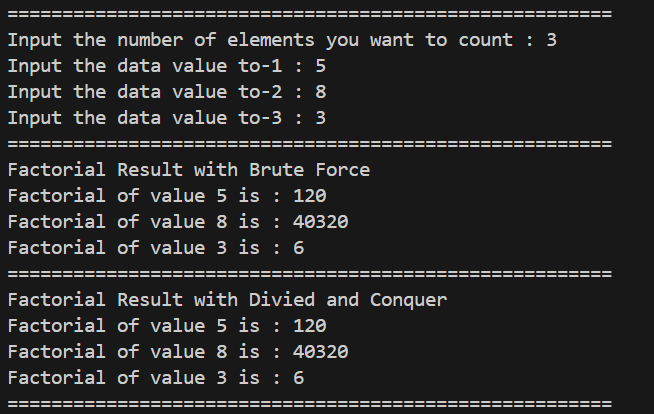
* The Divide Conquer algorithm for calculating factorial works by breaking down the problem into smaller subproblems until they become simple enough to solve directly.

1. In the implementation of Factorial Divide and Conquer Algorithm is it complete that consists of 3 stages of divide, conquer, combine? Explain each part of the program code!

* In the given implementation, the Divide and Conquer algorithm for calculating factorial is represented by the faktorialDC(int n) method. Let's break down each part:
* Divide: This part is implicit in the recursive nature of the method. Each recursive call divides the problem into a smaller subproblem by decrementing n.
* Conquer: The faktorialDC(int n) method recursively calls itself with n-1 until n becomes 1. At this point, it returns 1, which is the base case.
* Combine: There is no explicit combine step in this implementation. However, the combination of results happens implicitly as the recursive calls return and compute the final result.

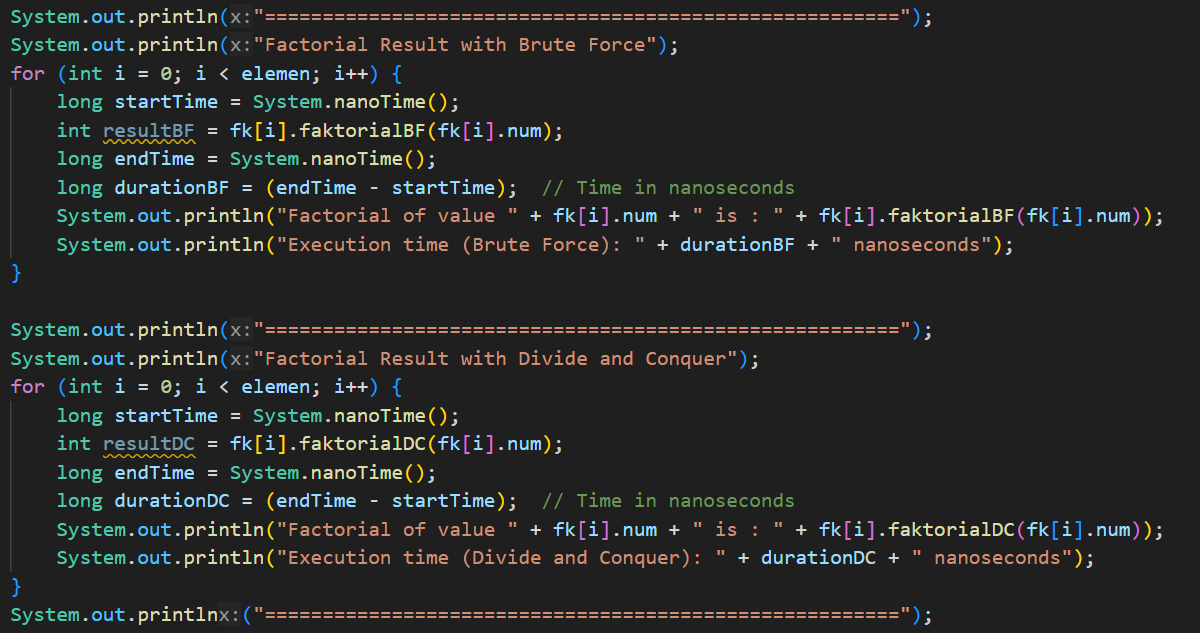
1. Is it possible to repeat the factorial BF () method instead of using for? Prove it!

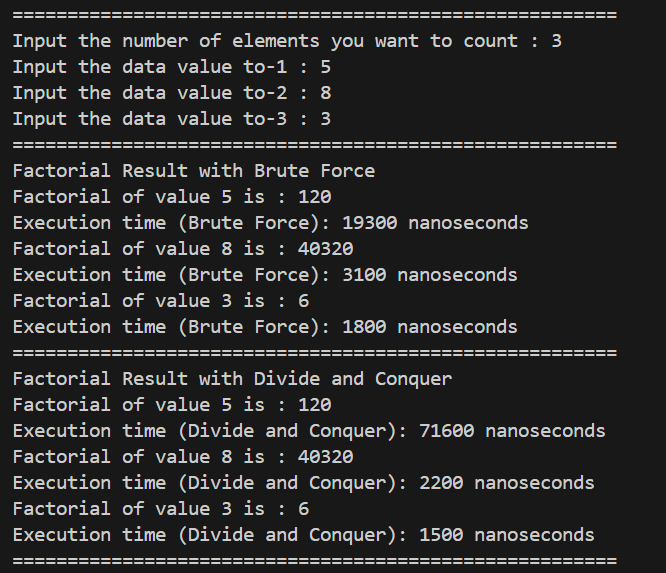
* Yes, it's possible to repeat the faktorialBF() method without using a for loop. We can achieve this by using recursion.

1. Add a check to the execution time of the two types of methods!

* Add timing checks to measure the execution time of each method using System.currentTimeMillis() or System.nanoTime().



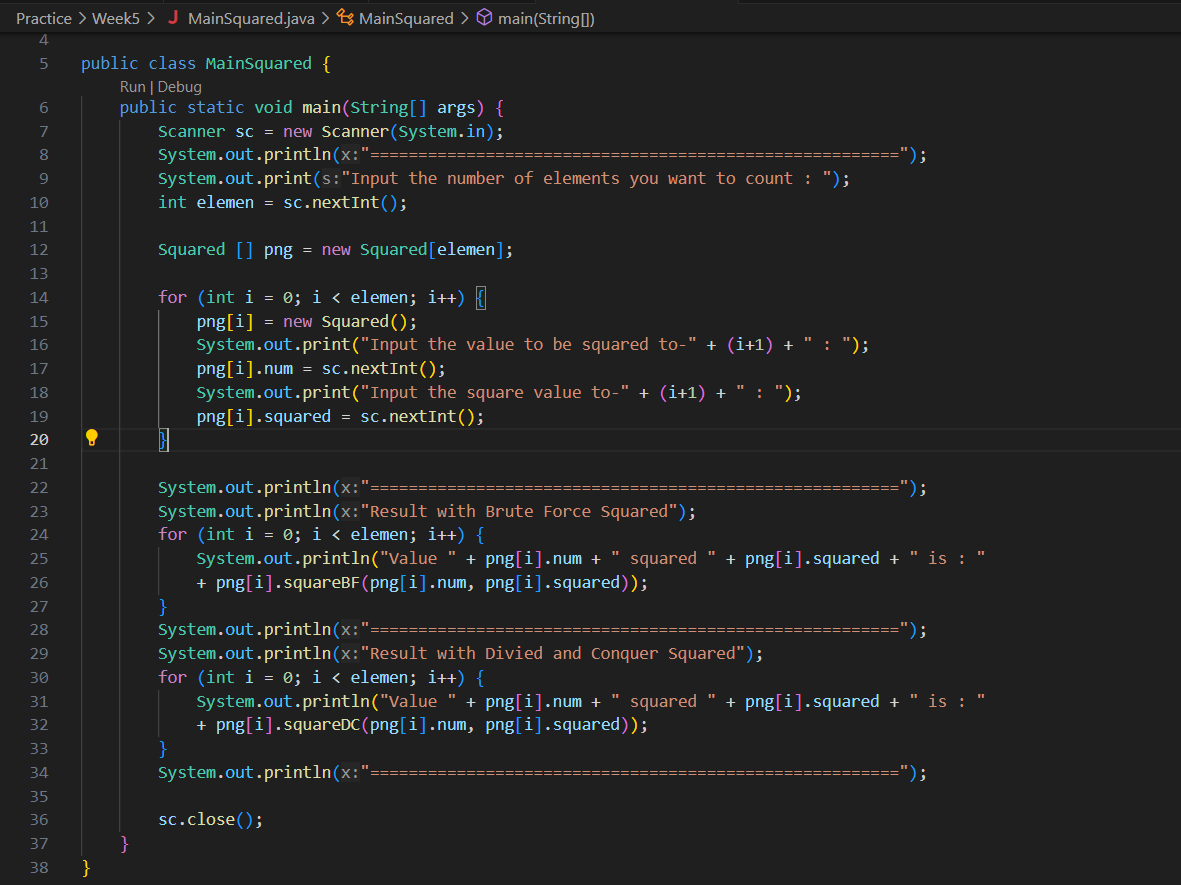


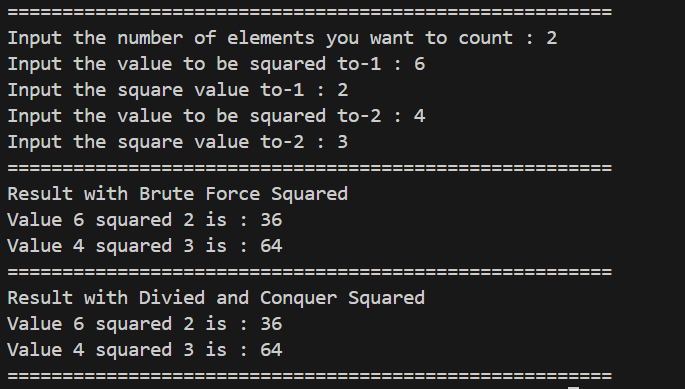
1. Prove by inputting elements that are above 20 digits, is there a difference in execution time?

* Yes, there will likely be a noticeable difference in execution time for larger input values due to the nature of the algorithms. The brute-force method (faktorialBF()) has a time complexity of O(n) because it iterates through all numbers from 1 to n. However, the divide and conquer method (faktorialDC()) has a time complexity of O(log n) due to its recursive nature. As the input value increases, the difference in execution time between the two methods will become more apparent, with the divide and conquer method being more efficient for larger inputs.

**Lab Activity 2: Calculating Squared Results with Brute Force and Divide and Conquer Algorithms**

**Result**

****

****

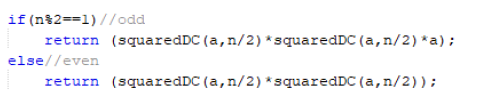
**Question**

1. Explain the differences between the 2 methods made are SquaredBF() and SquaredDC()!

* SquaredBF(): This method uses a brute-force approach to calculate the power of a number. It iteratively multiplies the base number by itself n times.

SquaredDC(): This method utilizes a divide and conquer approach. It recursively divides the power n by 2 until it reaches the base case (n == 0), and then combines the results accordingly. If n is odd, it splits the power into two equal parts and multiplies them by the base number and then multiplies the result by the base number again. If n is even, it splits the power into two equal parts and multiplies them together.

1. In the SuaredDC () method there is a program as follows:



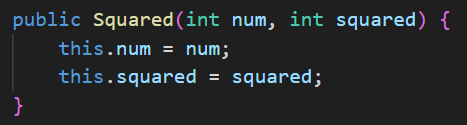
Explain the meaning of the code!

* This part of the code is handling the recursive calculation in the divide and conquer approach. It checks if the power n is odd or even.
* If n is odd, it splits the power into two equal parts (n/2), recursively calculates the square of the base number for both parts, and then multiplies them together. Finally, it multiplies the result by the base number a.
* If n is even, it splits the power into two equal parts (n/2), recursively calculates the square of the base number for both parts, and then multiplies them together without multiplying by a again.

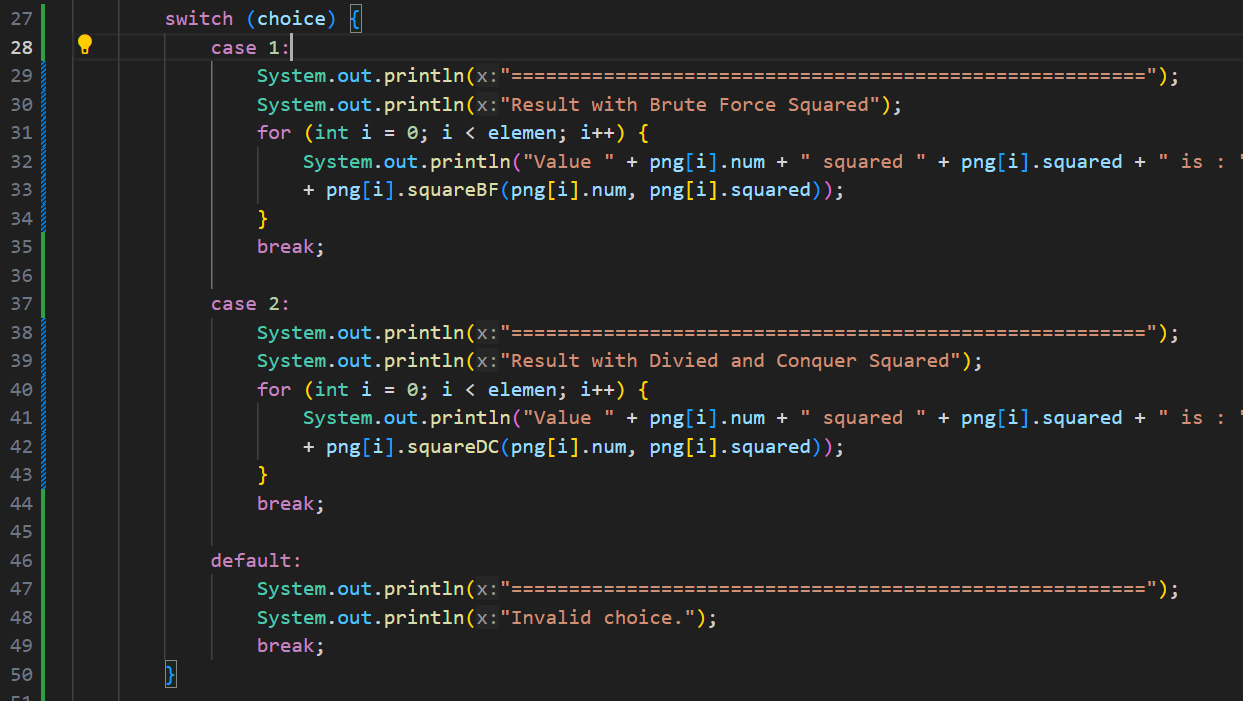
1. Explain whether the combine stage is included in the code!

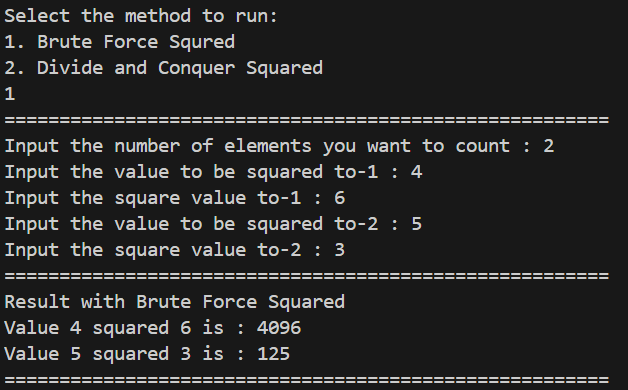
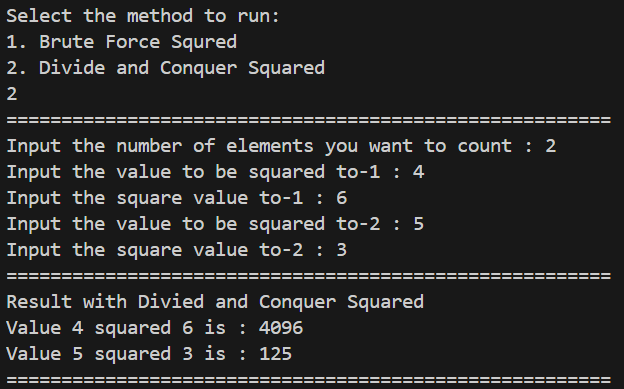
* The combine stage in a divide and conquer algorithm is essentially the part where the results of the recursive calls are combined to produce the final result. In the provided SquaredDC() method, the combine stage is implicit in the way the recursive calls are made and the results are multiplied together. Each recursive call returns a result, and these results are multiplied together in the return statement, which effectively combines them to produce the final squared value.

1. Modification of the program code, assuming the attribute filling process is done by a constructor.



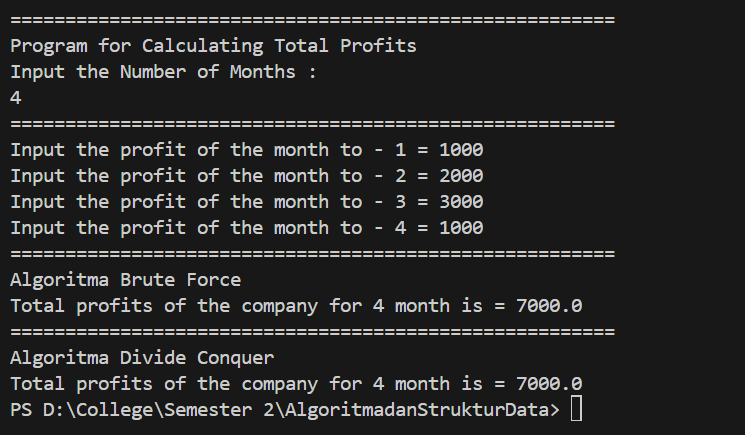
1. Add a menu so that only one of the selected methods will be run!



**Lab Activity 3: Calculating Sum Array with Brute Force and Divide and Conquer Algorithms**

**Result**

****

**Question**

1. Give an illustration of the difference in profit calculation with the TotalBF () or TotalDC () method.

* TotalBF(): This method calculates the total profit by simply iterating over the array of profits and summing them up. It has a time complexity of O(n), where n is the number of months.

TotalDC(): This method uses a divide and conquer approach to calculate the total profit. It recursively divides the array into halves until individual elements are reached, then combines the results. It has a time complexity of O(n log n) due to its divide and conquer nature.

Illustration:

* Suppose we have profits for 12 months: [1000, 2000, 1500, 3000, 2500, 1800, 4000, 3500, 2700, 2200, 3200, 2800].
* TotalBF() will simply sum all these values: 1000 + 2000 + 1500 + ... + 2800 = 29800.
* TotalDC() will recursively divide the array into halves until it calculates the total profit for each half and combines them. It's a bit more complex but achieves the same result.

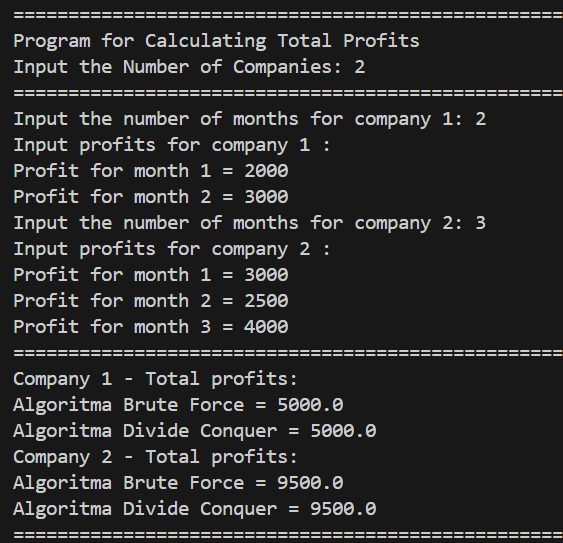
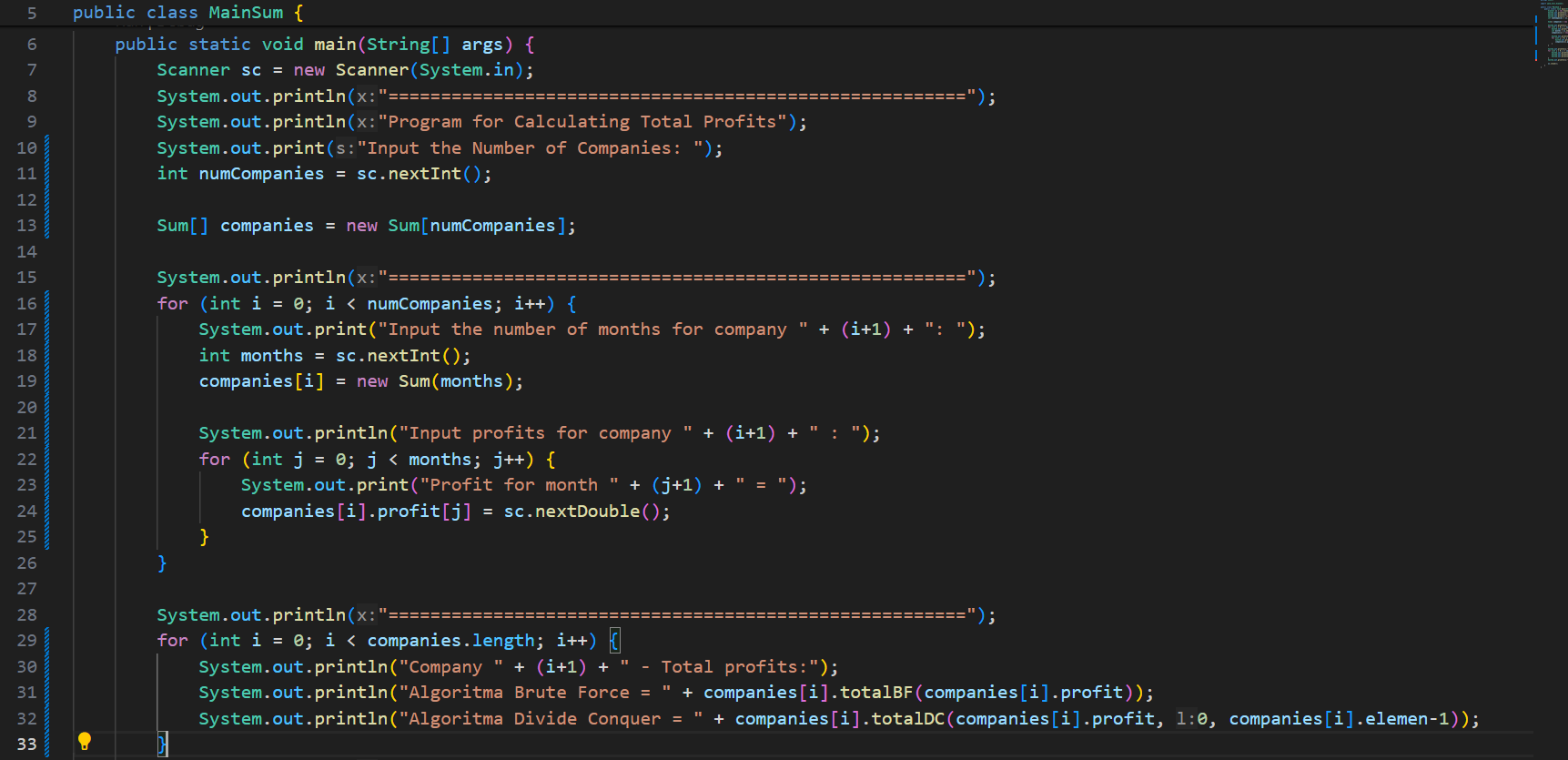
1. Why is there the following return value? Explain!

* The return statement in the totalDC() method is used to return the sum of profits calculated for the left subarray (lsum), the right subarray (rsum), and the middle element (arr[mid]). This is a crucial part of the divide and conquer approach. It ensures that the total profit is correctly calculated by summing up the profits from both halves and adding the profit from the middle element.

1. Why is the mid variable required for the TotalDC () method?

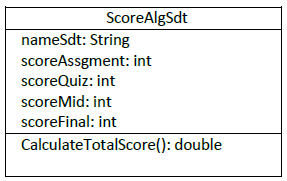
* The mid variable is required to divide the array into two halves in each recursive call. It determines the middle index of the array, which separates the array into two subarrays - one from l to mid-1, and the other from mid+1 to r. This splitting is essential for the divide and conquer approach to work correctly.

1. The profit calculation program for a company is only for one company. How do you calculate several months of profit for several companies at once (each company can have a different number of months)? Prove it with the program!

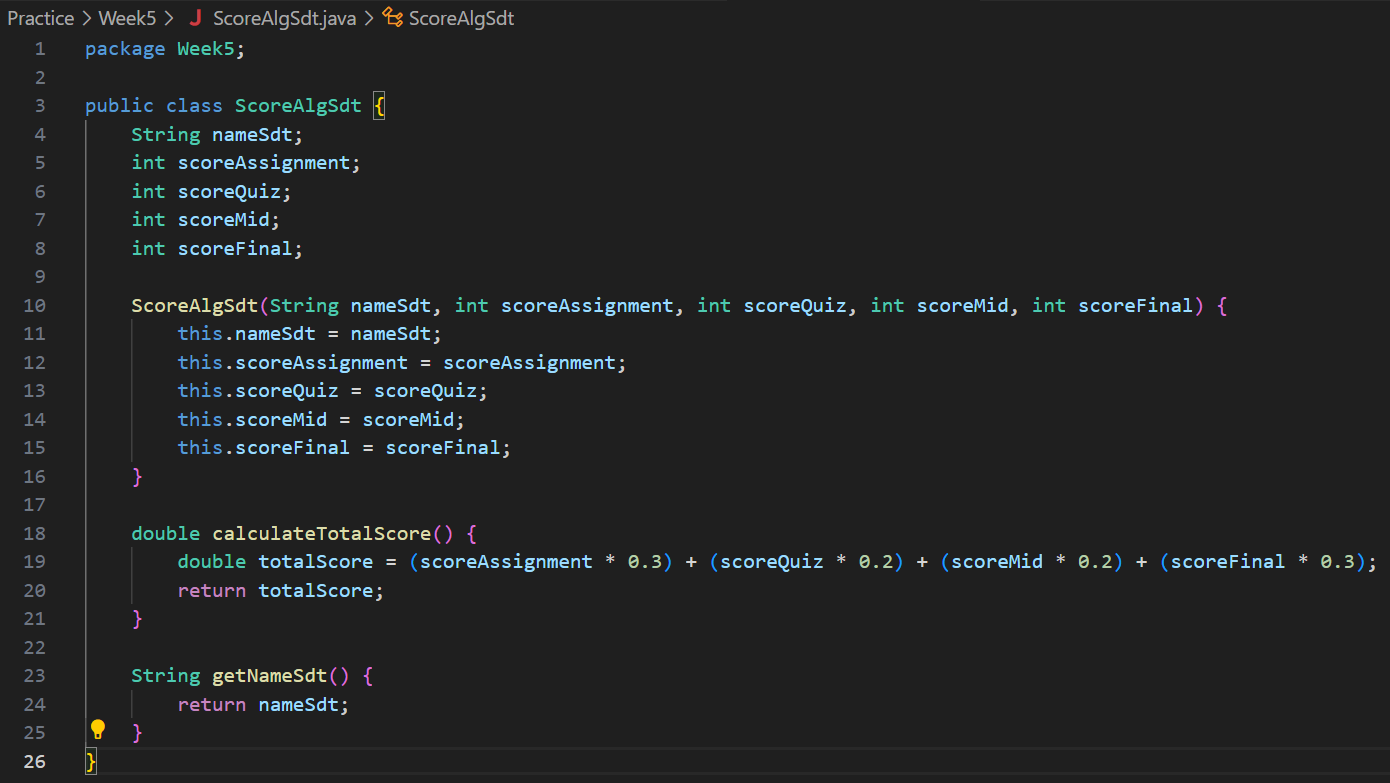


**Assignment**

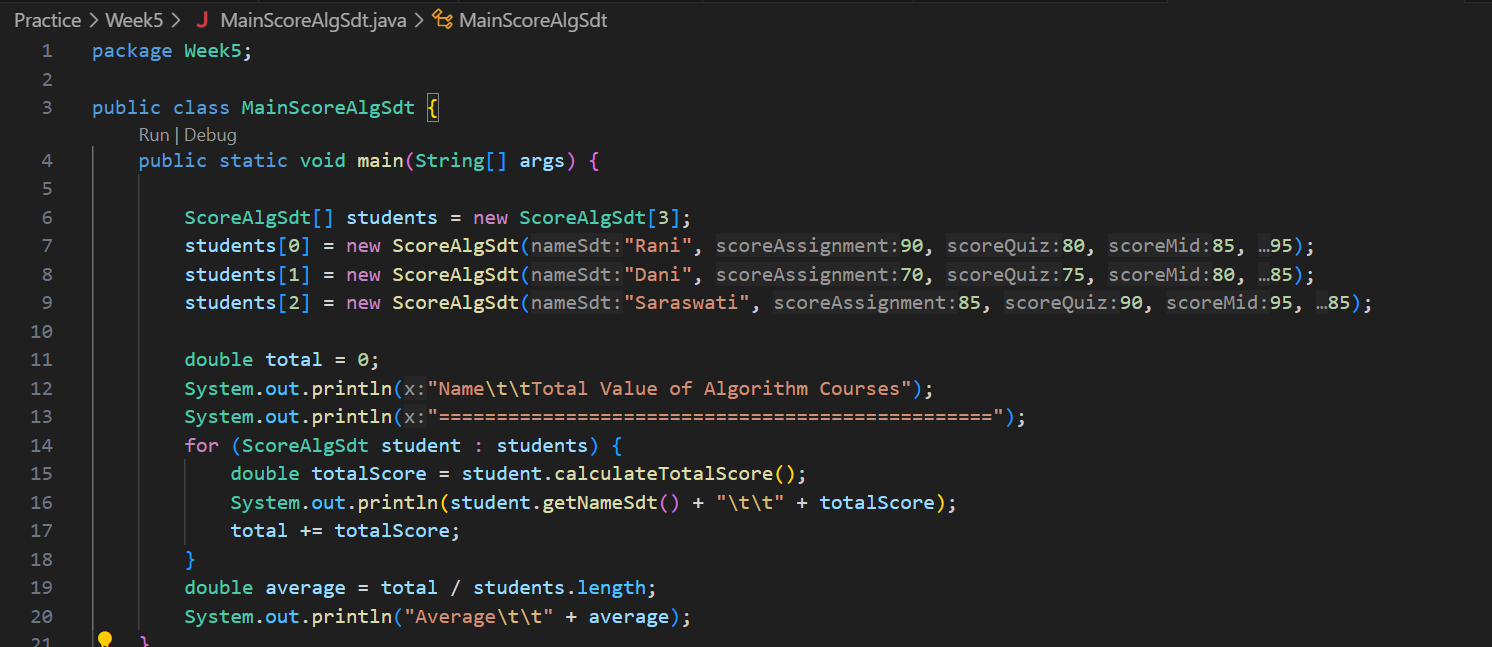
1. Create a program based on the following class diagram!



Score are calculated based on total Assignment of 30%, Quiz 20%, Mid 20%, Final 30%. Adjust the method if it must have parameters.



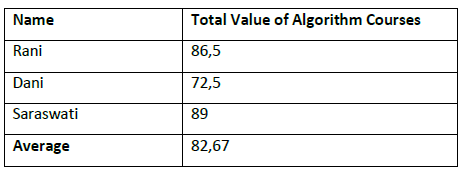
1. (continued about question no. 1). Create an array of objects in the main class to find out the value of several students in one calculation at a time!



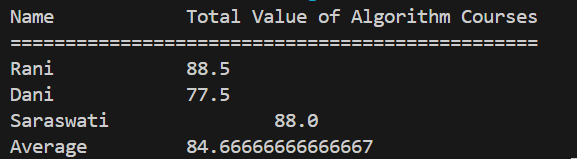
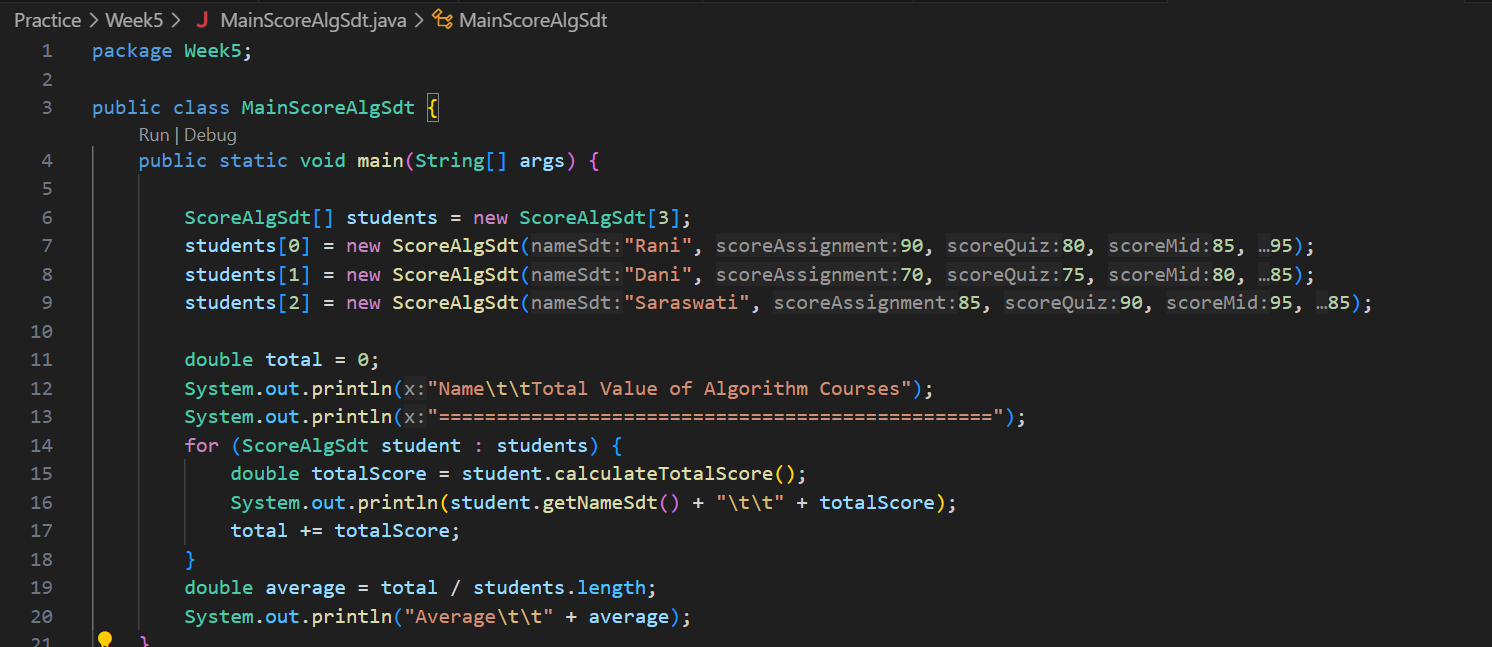
1. (Continued question no. 2) with the modified Brute Force algorithm program in order to know the average value of all students who have been inputted to the Algorithm and data structure course.

Example :

There are a total of the following student grades

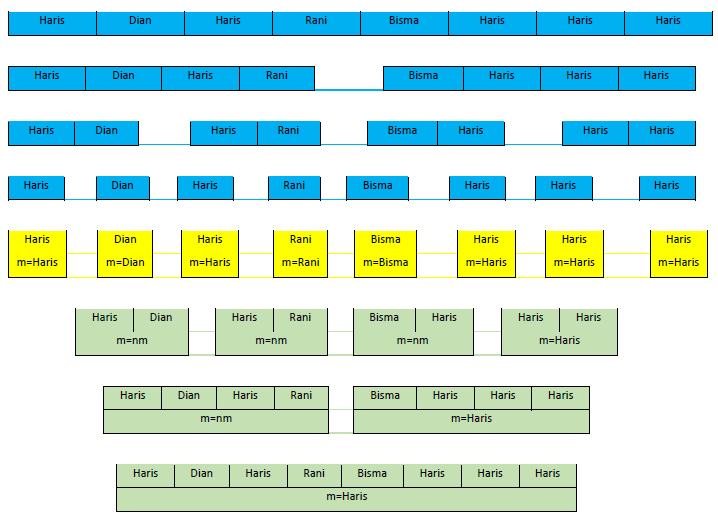


Then the average total value of all students who have taken the algorithm course is 82.67



1. A university in Malang is holding a vote to elect the BEM chairman in 2020. If the number of votes collected is always even. Then by inputting the selected candidates, count the number of votes for each candidate. Make class diagrams and programs using the Divide and Conquer algorithm from the case study! (The number of array elements and the results of the vote are user input)

Example: Voting results are as follows (m is majority, nm is not majority)



Description: Blue is the divide process, yellow starts the conquer process, green starts the combining process

1. What if the number of votes is odd? should there be a program improvement? if yes, improve the program for case study no 4. If the number of votes collected is not always even!
2. Modify the program about the average value of algorithm courses using the Divide and Conquer Algorithm!