Assignment #10

Rikiya Takehi

1W21CS04-6

Algorithms and Data Structures

04/07/2022

**Problem Statement**

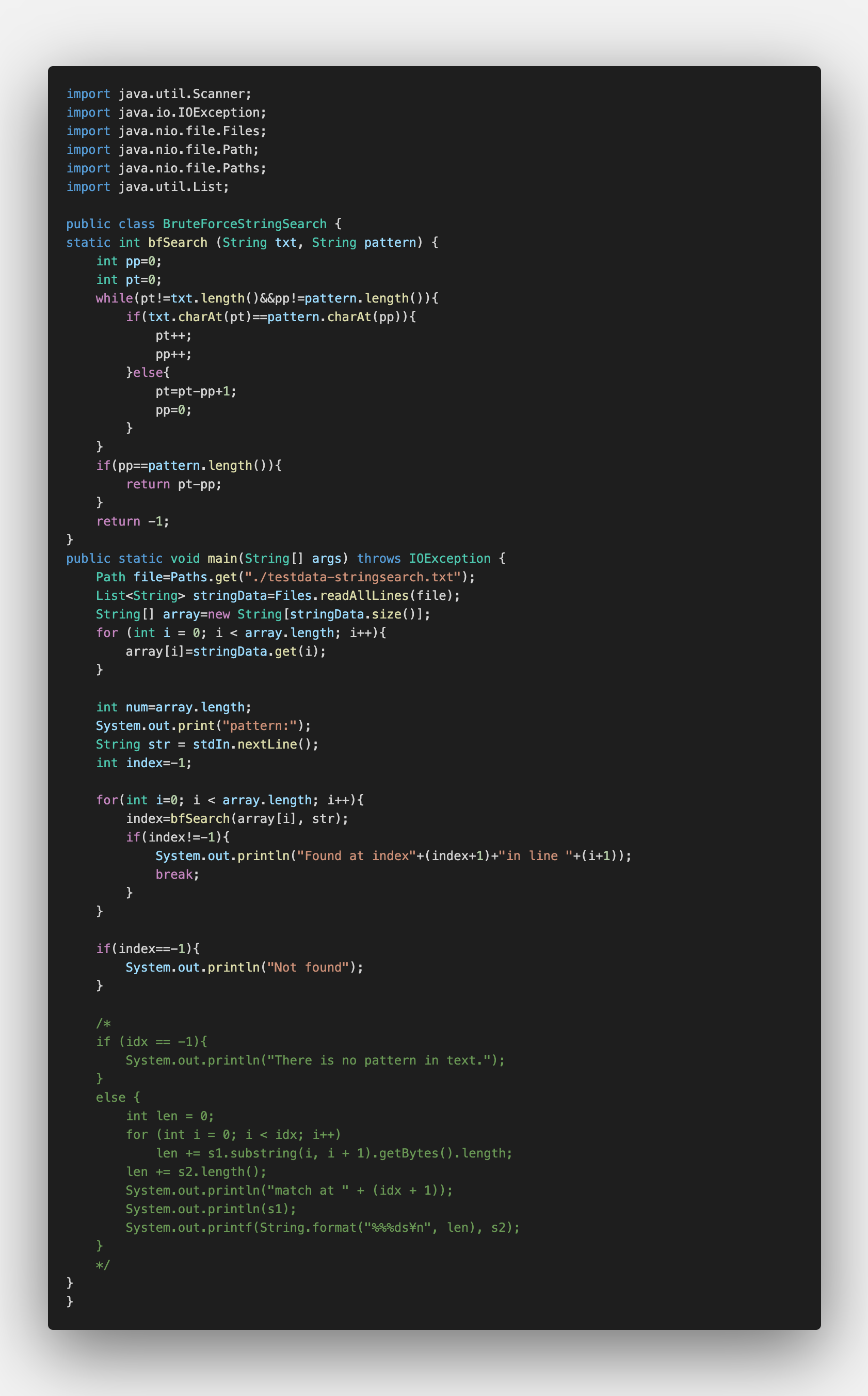
Write a quick sort program, binary search program, and brute force string search program. For the quick sort program, use testdata-sort-{1~4}.txt and measure its execution time for each of the test data. For the binary search program, testdata-search.txt is given and key 799,829 is searched. For brute force string search program, testdata-tringsearch.txt is used, and the pattern “trophy” is searched.

**Programs**

quick sort program:

binary search program:



brute force string search program:

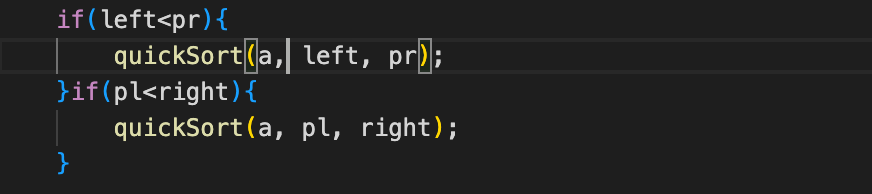
**Program explanation**

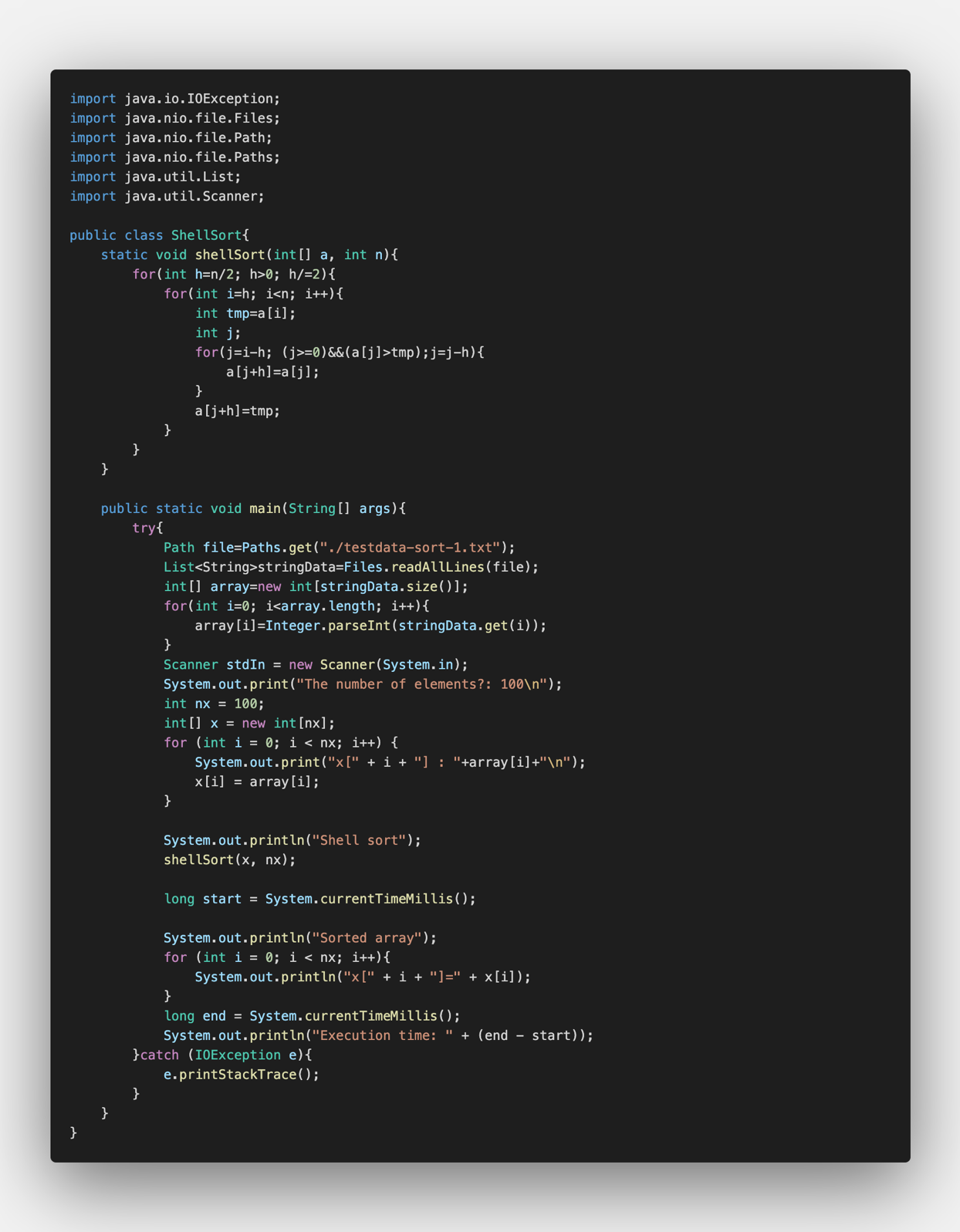
Quick Sort Program

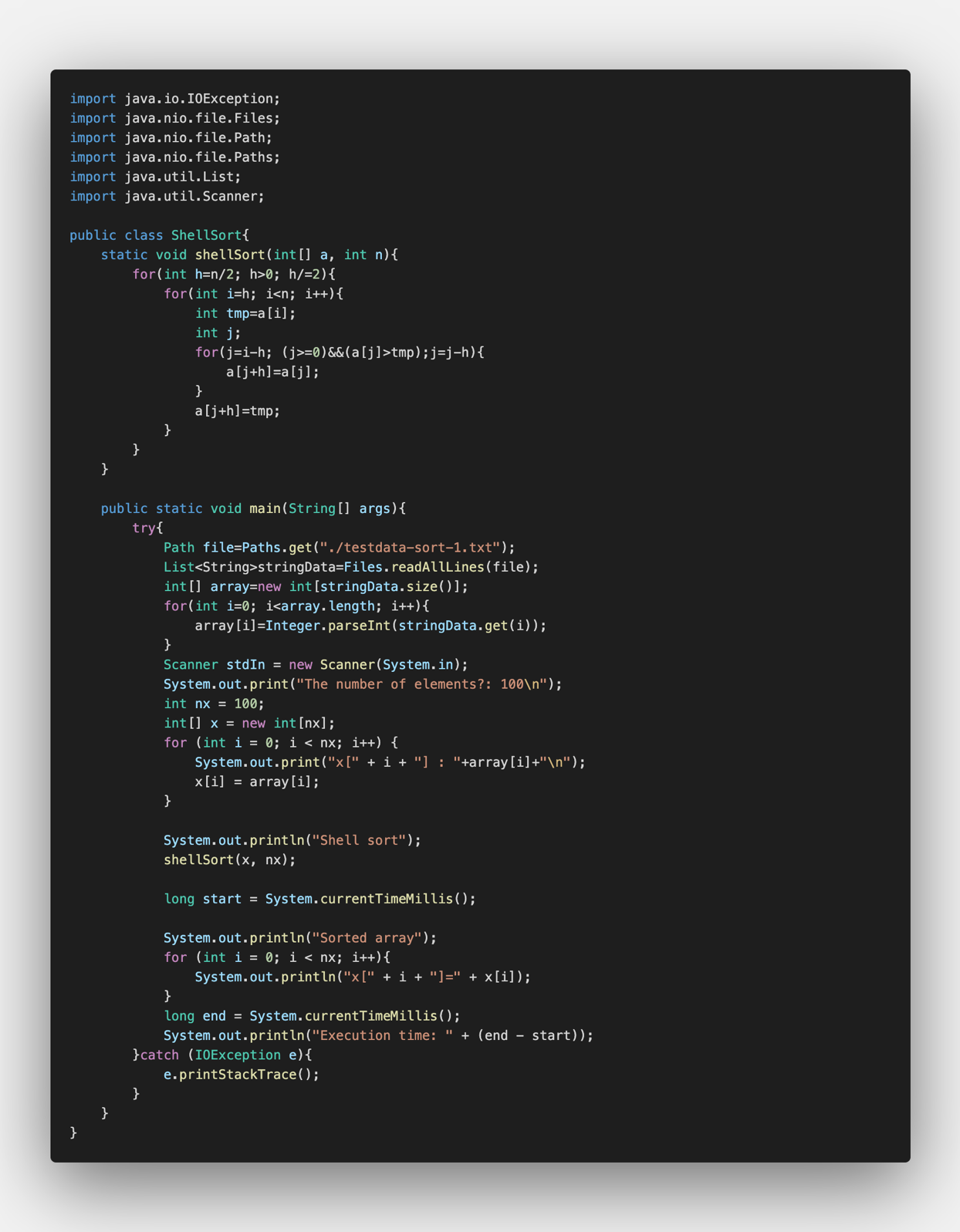
Lines 9-11 are codes where two numbers (idx1 and idx2) can be swapped. By using an integer d, it replaces the numbers. In the later codes, it only requires to recall “swap” in order to swap two specific numbers.

A quick sort array makes two arrays and calls itself in a recursive way to sort the two subarrays.

In lines 14-28, it identifies the pivot in line 17, and divides it into the left section (pl) and the right section (pr). While the left section is smaller than the right section, the following procedures are done. If the number on the array is smaller than the pivot value, it adds 1 to pl, and if the number is bigger than the pivot value it subtracts 1 from pr. This way, it makes two subarrays.

Then, in lines 29-33, it says to quick sort the two subarrays recursively until it is completely sorted.

In line 47, it records the exact time using the following code and states “start”.

In line 53, it records the time again using the following code and states “end”. Then in line 54, it shows the time taken to complete the sorting by subtracting the time recorded as “end” by time recorded at “start”.

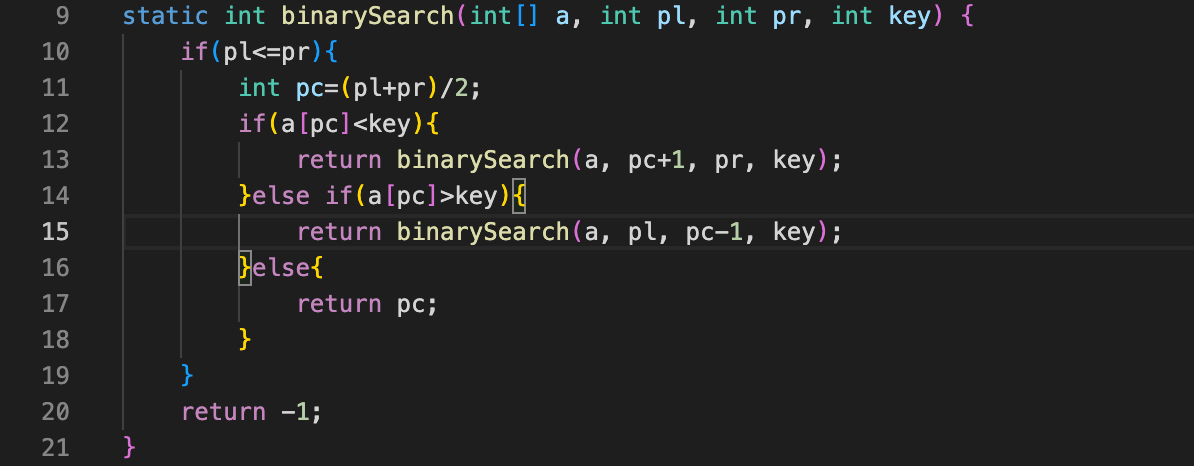
When switching the data and the sorting method, change the names in line 37.



Binary Search Program:

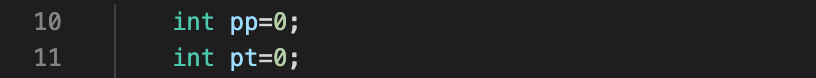
In this program, the binary search was done using recursion.

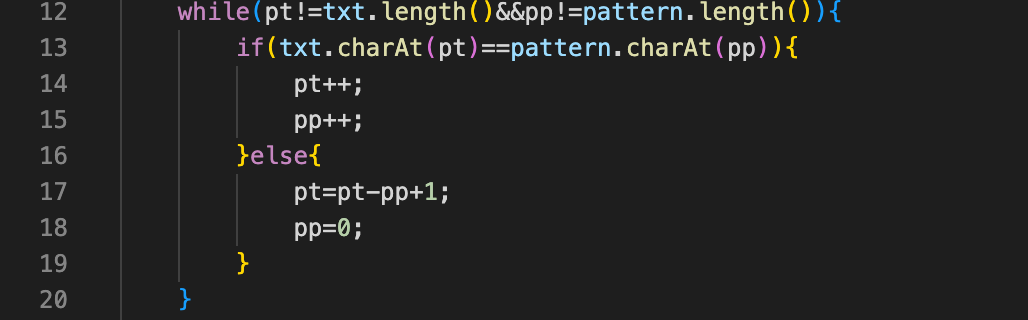
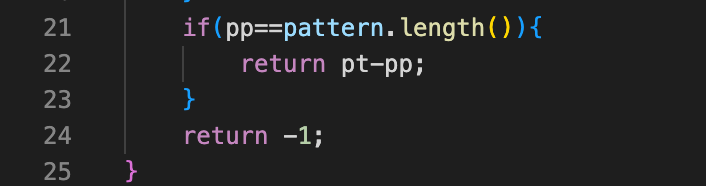
In lines 9 to 21, it shows the step in which the next actions are decided. In line 11, it identifies the center of the array. In line 12 and 13, if the key is bigger than the center number, it returns to do the binary search again but with the latter half of the array. In lines 14 and 15, if the key is smaller than the center number, it returns to do the binary search again but with the first half of the array. If the number is the center, it returns the center number as the key (line 16 and 17). If the procedure is repeated until the end and number is not found, it returns “-1” which shows that it was unsuccessful.



brute force string search program:

Brute force string search program is a string program that searches each of the characters one by one until it has a mismatch. In lines 10, 11, it names pp and pt each for the cursor of txt (string array) and the cursor of pp (pattern).

If the current character of txt is the current character of the pattern, if compares the next character of the pattern and the text is compared (lines 13-15). If the characters are not the same, it shifts the pattern down the text by one character (lines 16-18).

If pp is the length of the pattern, it shows that the entire length is found, so it returns pt-pp (the place it was found) and ends the procedure (lines 21-25).

**Experimental settings**

Input files:

“testdata-sort-1.txt”, “testdata-sort-2.txt”, “testdata-sort-3.txt”, “testdata-sort-4.txt” for quick sort program.

“testdata-search.txt” for binary search program.

“testdata-stringsearch.txt” for brute force string search program.

Tested programs:

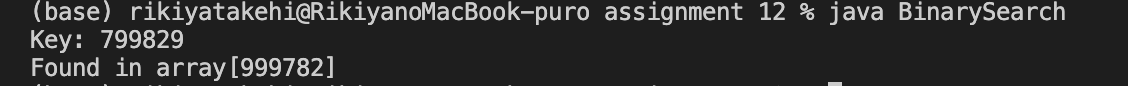
“QuickSort”, “BinarySearch”, “BruteForceStringSearchProgram”

**Results**

Quick Sort Program:

Table showing the time taken for quick sort program to sort each test data in milliseconds compared with other sorting methods

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | testdata-sort-1.txt | testdata-sort-2.txt | testdata-sort-3.txt | testdata-sort-4.txt |
| Quick sort | 21 | 543 | 2089 | 2451 |
| Bubble sort | 28 | 16096 | 6350 |  |
| Bubble sort 1 | 21 | 17052 | 496 |  |
| Bubble sort 2 | 23 | 16682 | 384 |  |
| Shell sort | 2 | 239 | 1764 | 1943 |
| Insertion sort | 2 | 1365 | 98083 | 1420 |
| Selection Sort | 2 | 7437 | 524848 | 558898 |

Binary Search Program:

Brute Force String Search Program:

The pattern trophy was found at index 25 in line 5896

**Discussion of the Results**

Quick Sort Program:

When comparing the quick sort program with others, it can be seen that it is fairly fast overall. The execution time is fairly slow for the test data sort 1, but are fairly fast for the rest of the test data. When compared with shell sort, it can be seen that the time taken for quick sort is slower by a little for every test data. This would show that the it takes time for quick sort to deal with the test data 1 because no matter the size of the test data, it is necessary to find the pivot multiple times. On the other hand, it is faster than many others in long arrays because it does not have to compare every single value in the array.

Binary Search:n