  
Name : Azaria Cindy Sahasika

Number Id : 2341760169 / 06

Class : 1G – Business Information System

Lesson : Algorithm and Data Structure

Material : Jobsheet 6

Github Link : <https://github.com/azariacindy/algorithm-ds>

JOBSHEET VI SEARCHING

# Learning Objective

After learning this practicum course, students will be able to:

1. Define Searching algorithm
2. Create and declare searching algorithm structure
3. Implement searching algorithm

# Sequential Search Method

Take a look on following class diagram! Use this class diagram as blueprint of program code in

**Students** class

|  |
| --- |
| Students |
| Nim: int name: String age: int  gpa: double |
| Students(ni:int, nm: String, age: int, gpa: double)  display(): void |

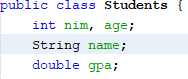
Create a **Students** class to make instantiation process of **Students** class which will be added in an array. There is a constructor with parameter and display() method to print all attributes available in Students class

|  |
| --- |
| SearchStudent |
| listStd: Student[5]  idx: int |
| add(mhs: Mahasiswa): void display(): void FindSeqSearch(int cari): int showPosition(int x,int pos): void  showData(int x,int pos) :void |

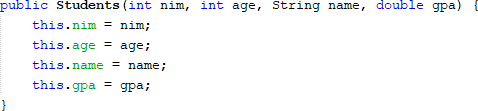
Next, above class diagram will represents a class to manipulate array of objects instantiated from **Students** class. For example, to add a student, display all student’s information, to search by NIM, and to display searched student’s data later on

## Steps

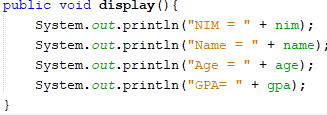
* + - 1. Create a new project in NetBeans called **TestSearching**
      2. Then, create a new package **week7.**
      3. Create new **Students** class, then declare following attributes:



* + - 1. Create a constructor in **Students** class with parameters (int ni, String nm, int age, double gpa). Convert it to program code as follows:



* + - 1. Create display() method with void as its return type

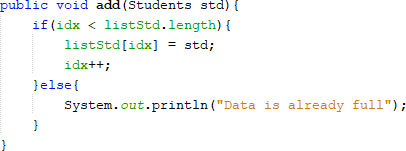


* + - 1. Create a new **SearchStudent** class as follows.

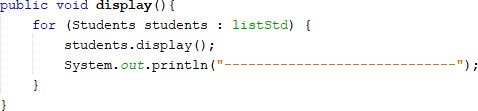


* + - 1. Create method **add()** at that class! This will be used for adding objects from **Students**

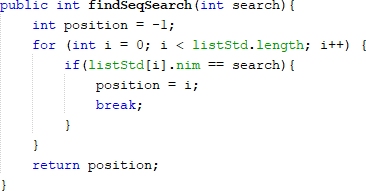
class to listStd attribute



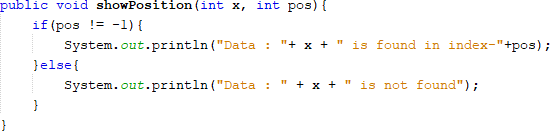
* + - 1. Create method **display()** in class **SearchStudent!** This display() method will be used to print all students data available in this class. Pay attention on how we use **for loops** differently. Even so, the concepts is still the same



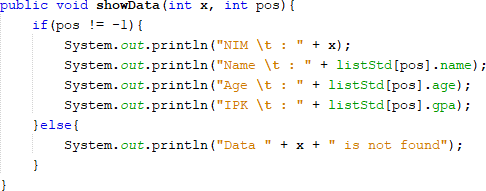
* + - 1. Create method **FindSeqSearch** with integer as its return type. Then fill in the function with sequential search algorithm.



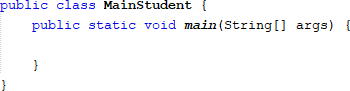
* + - 1. Create method **displayPosition** with void as its return type. And write these following code as follows



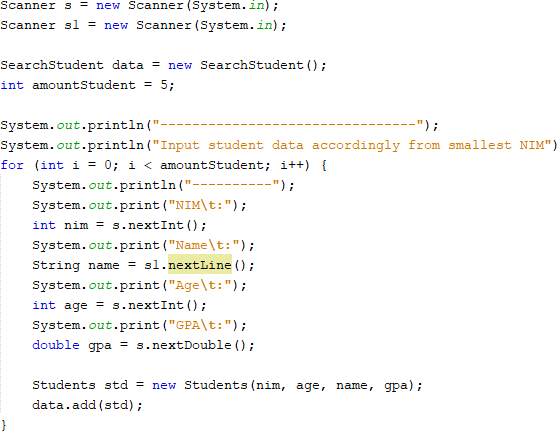
* + - 1. Create method **displayData** with void as its return type. And write these following code as follows



* + - 1. Create a main class named **StudentsMain** and add main method as follows



* + - 1. In main method, instantiate an object in **SearchStudent** that consist of 5 **Students,** then add all students object by calling **add** function in object **SearchStudent**



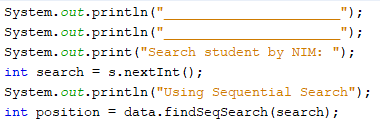
* + - 1. Add method **display** to print all inserted data



* + - 1. To search students by their NIM, create a **search** variable to hold input from user. Then call method **FindSeqSearch** with its parameter is the search variable we’ve declared before



* + - 1. Call method **displayPosition** from class **SearchStudent.**



* + - 1. Call method **displayData** from class **SearchStudent**

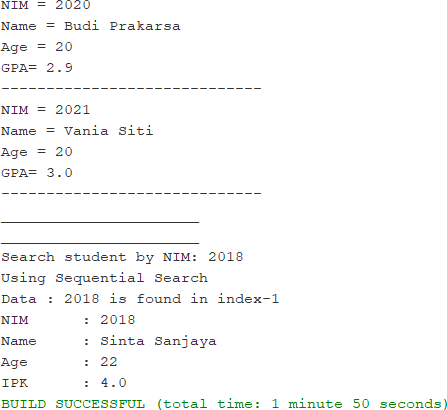
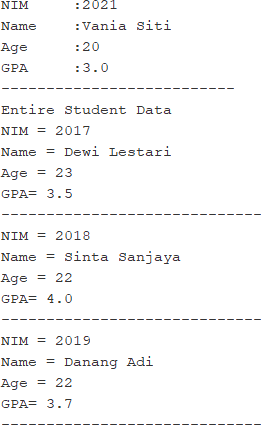
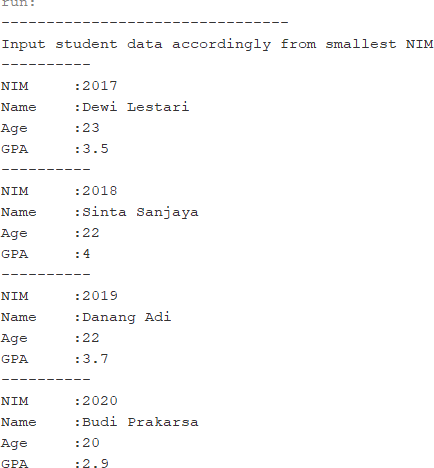


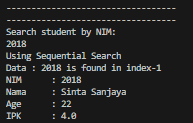
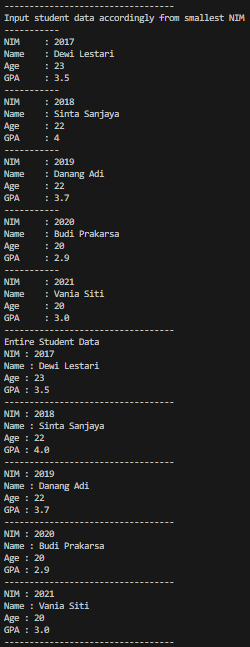
* + - 1. Run the program and see the result



## Result

Match the output of your program code with following image



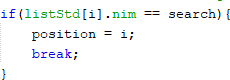


## Question

* + - 1. What is the difference of method **displayData** and **displayPosition** in **StudentSearch**

class?

* 'displayData' to display detailed information (nim, name, age, ipk) of a student at a specific position in the array.
* 'displayPosition' to display only the position of the student in the array (index).
  + - 1. What is the function of break in this following program code?

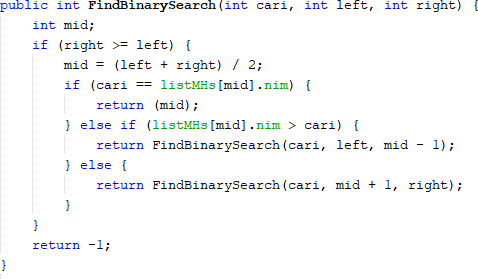


* The 'break' function is to exit the loop immediately when the student with the desired nim is found.
  + - 1. If inserted NIM data is not sorted from smallest to biggest value, will the program encounter an error? Is the result still correct? Why is that?
* If the nim data entered is not sorted, the program will not experience an error. However, the result of the search operation may not be correct.

# Binary Search Method

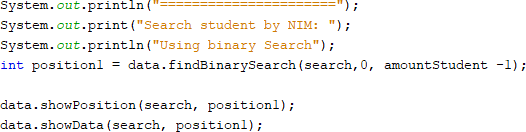
## Steps

* + - 1. in step 1.2.1 (Sequential search), create method **FindBinarySearch** with integer as its data type in class **SearchStudent.** Then declare the content of method **FindBinarySearch** with using binary search as its searching algorithm



* + - 1. Call method **FindBinarySearch** from **SearchStudent** class in **StudentsMain**. Then call method

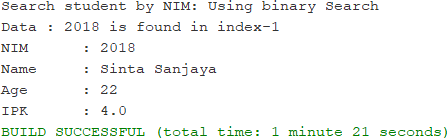
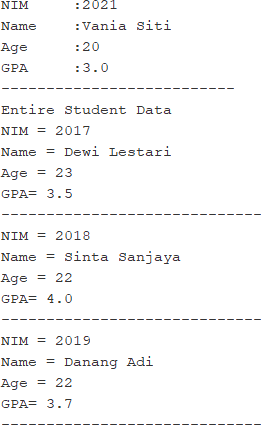
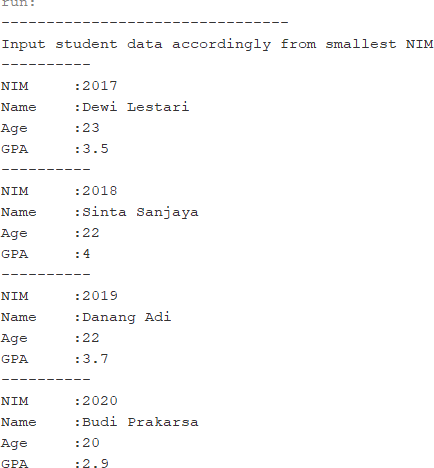
## displayPosition and displayData

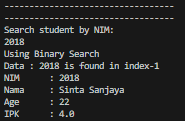
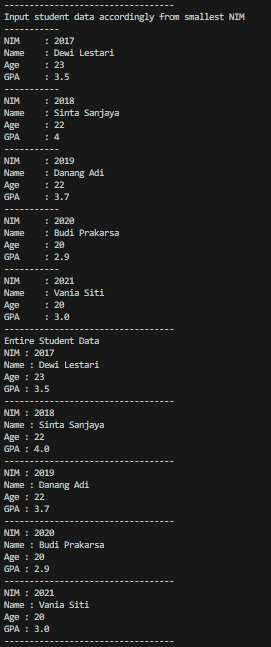


* + - 1. Run and see the result

## Result

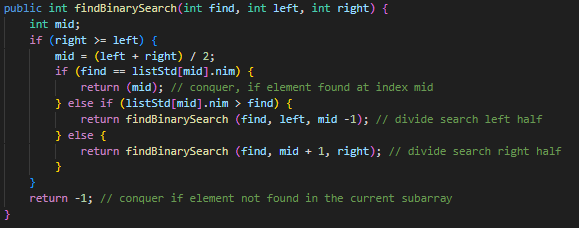
Match the output of your program code with following image



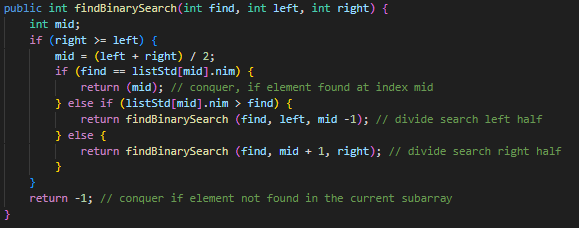


## Question

* + - 1. Show the program code in which runs the divide process

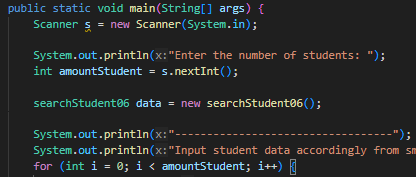


* + - 1. Show the program code in which runs the conquer process



* + - 1. If inserted NIM data is not sorted, will the program crash? Why?

If inserted NIM data is sorted from largest to smallest value (e.g 20215, 20214 20212, 20211,20210) and element being searched is 20210. How is the result of binary search? does it return the correct one? if not, then change the code so that the binary search executed properly

* If the NIM data is sorted from the largest value to the smallest and the searched element is 2040, the binary search may not give the correct result because it assumes the array is sorted in ascending order.
  + - 1. Modify program above so that the students amount inserted is matched with user input

# Review Divide and Conquer

## Steps

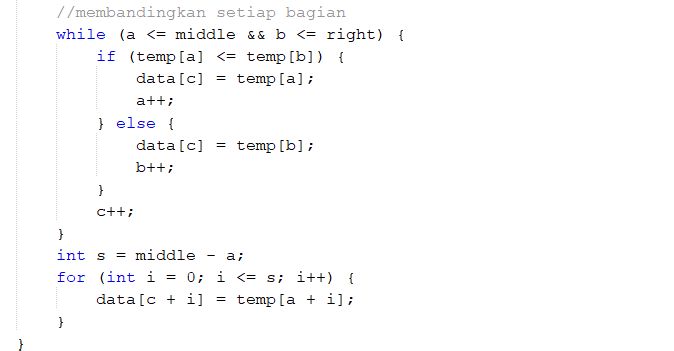
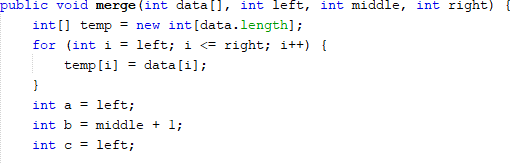
* + - 1. Create a new package in NetBeans named **MergeSortTest**
      2. Add class **MergeSorting** in this package
      3. In this class, create method mergeSort that receives an array in its parameter



* + - 1. Create **merge** method to do data merging process from left side to the right



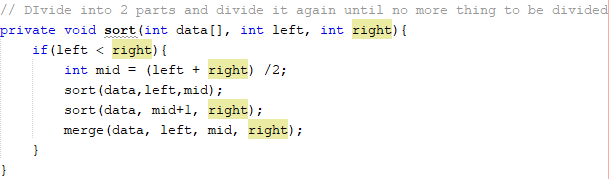
* + - 1. Implement **merge** process as follows:



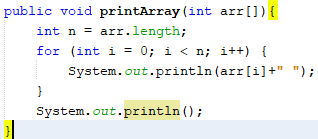
* + - 1. Create sort method



* + - 1. Implement these following codes in sort method



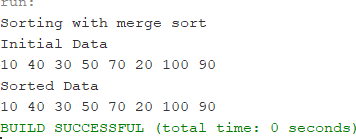
* + - 1. In method **mergeSort**, call method sort with the data that wants to be sorted and initial data range as its parameter
      2. Add method printArray

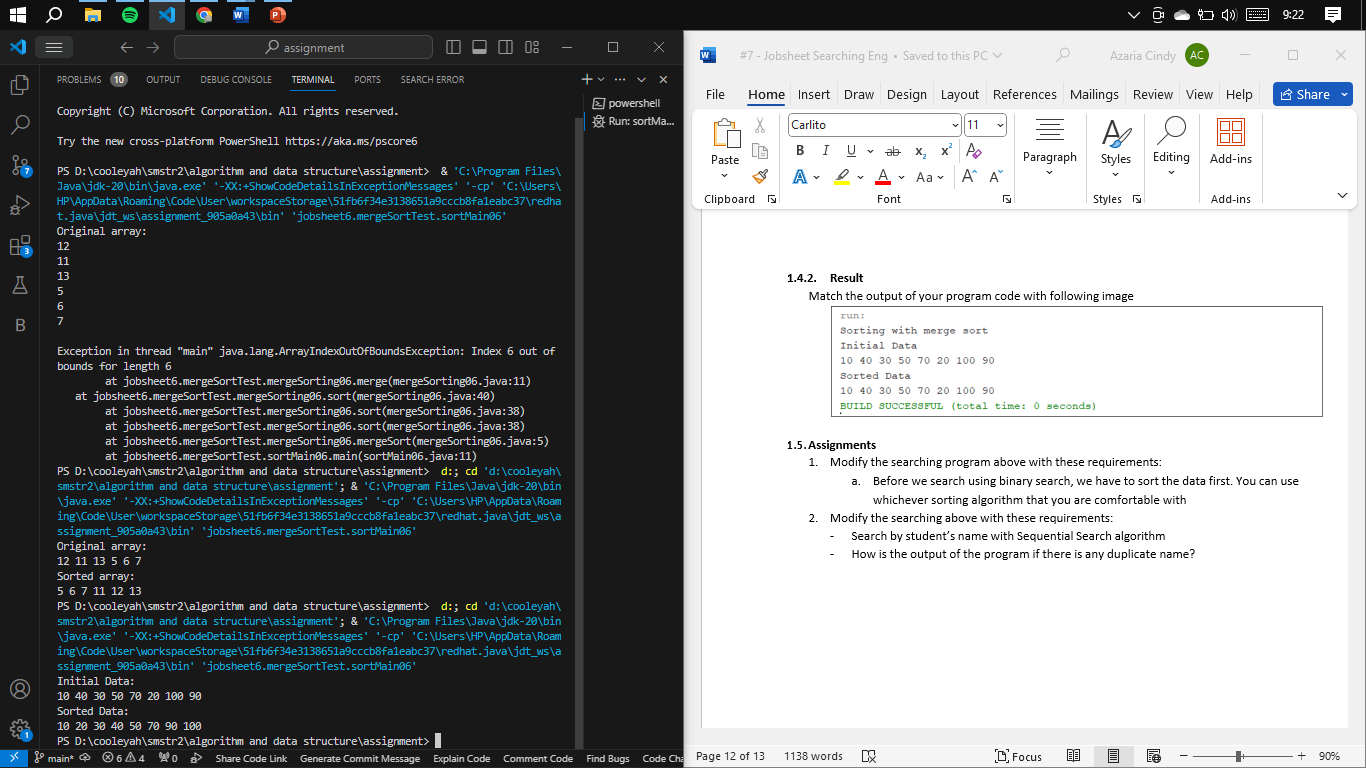
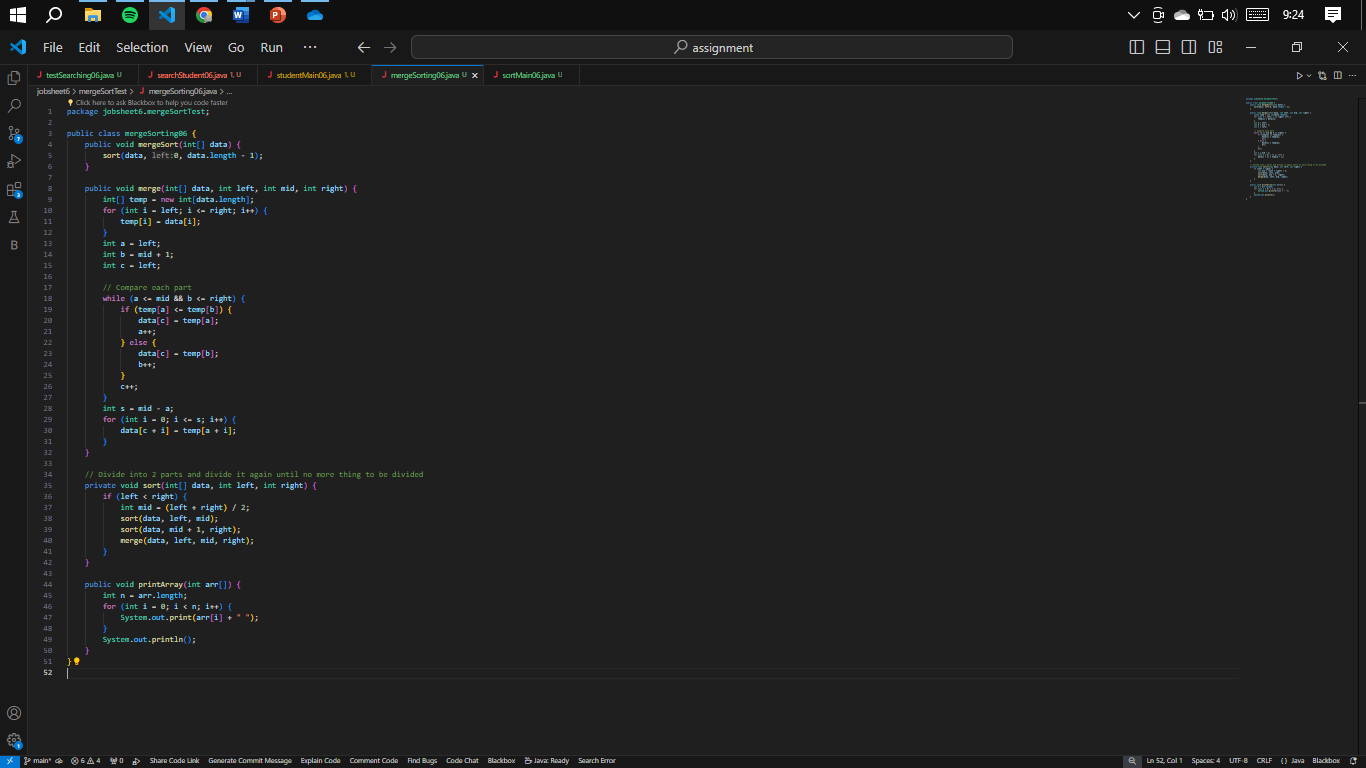
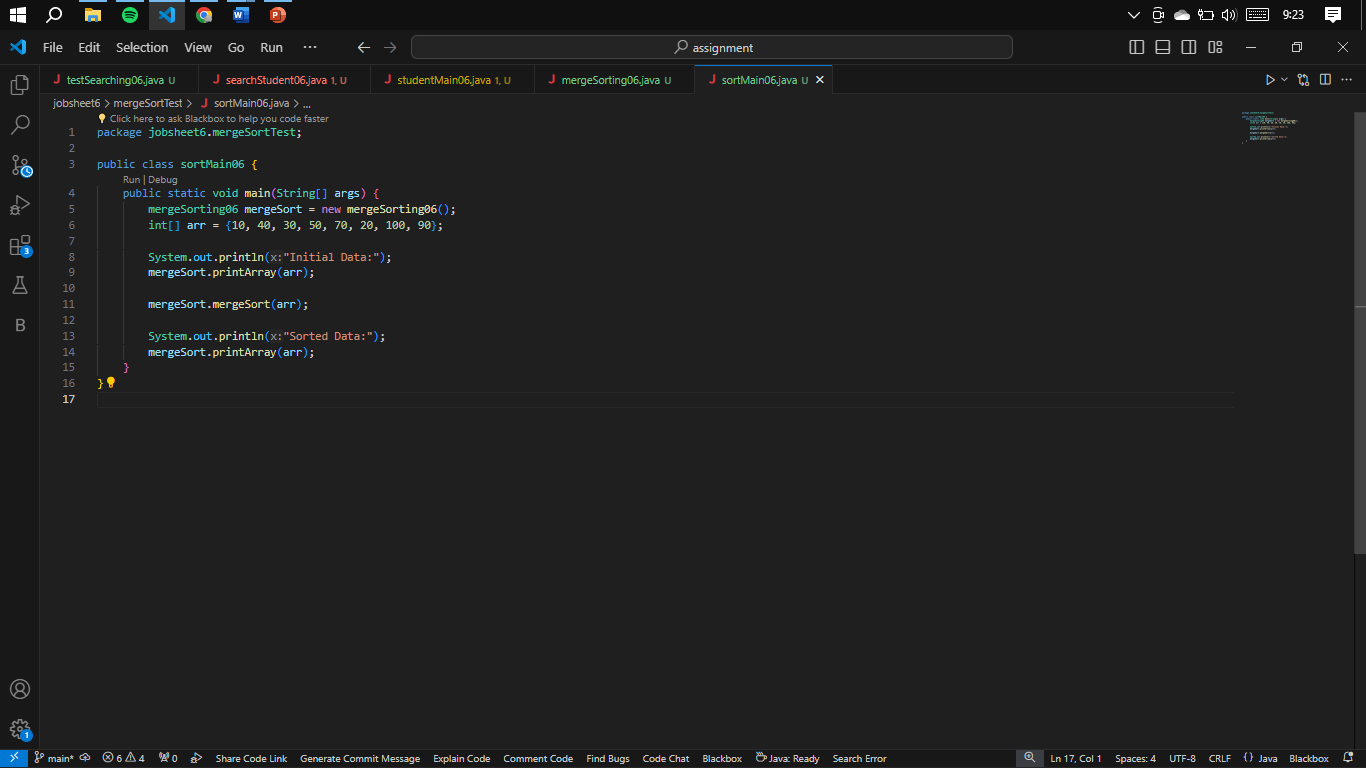


* + - 1. Finally, declare the data to be sorted by using sorting process in **SortMain** class

## Result

Match the output of your program code with following image



## Assignments

1. Modify the searching program above with these requirements:
   1. Before we search using binary search, we have to sort the data first. You can use whichever sorting algorithm that you are comfortable with
2. Modify the searching above with these requirements:

* Search by student’s name with Sequential Search algorithm
* How is the output of the program if there is any duplicate name?

1. There is 2d array as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Index** | **0** | **1** | **2** | **3** | **4** |
| **0** | 45 | 78 | 7 | 200 | 80 |
| **1** | 90 | 1 | 17 | 100 | 50 |
| **2** | 21 | 2 | 40 | 18 | 65 |

Based on data above, create a program to search data in 2d array, which the data to be searched is defined by user input (using sequential search)

1. There is a 1D array as follows:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **5. 0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| 12 | 17 | 2 | 1 | 70 | 50 | 90 | 17 | 2 | 90 |

Create a program to sort the array, search & display the biggest value, and print the amount of biggest value available alongside with its position.

