JOBSHEET VI

**SORTING (BUBBLE, SELECTION, DAN INSERTION SORT)**



**From:**

AL AZHAR RIZQI RIFA’I FIRDAUS

**Class:**

1 I

**Absence:**

01

**Student Number Identity:**

2241720263

**Department:**

Information Technology

**Study Program:**

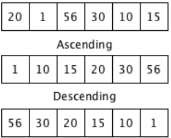
Informatics Engineering

## Learning Objective

* 1. Students are able to understand sorting algorithm
  2. Students are able create and declare the sorting algorithm
  3. Students are able to implement sorting algorithm

## Theory

Sorting is a process of ordering data from a list. There are 2 kind of sorting, mostly known as *ascending* and *descending* sorting. To get better understanding of those, kindly check on following illustration:



# Figure 1 Sorting Process

In this module, we will cover 3 method of sorting process. Such as:

## Bubble Sort

1. **Selection Sort**

## Insertion Sort Bubble Sort

This sorting method is the easiest to implement. However, this method is less effective compared to others. We sort the list by comparing one data to another respectively, each data comparison will be followed by swapping process. This swapping process depends on sorting model (*ascending* or *descending*).

Comparison and swapping process will be repeated for the second time (the 2nd data compared with 3rd data and loops until the end of the list). The same goes with comparison and swapping process for 3rd data until the end. This process loops until there is no data left to be compared. This following illustration of Bubble sort method may help you clear the algorithm

Initial Data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 50 | 25 | 1 | 3 |

Step 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 50 | 25 | 1 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 50 | 25 | 1 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 25 | 50 | 1 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 25 | 1 | 50 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 25 | 1 | 3 | 50 |

Step 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 25 | 1 | 3 | 50 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 25 | 1 | 3 | 50 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 1 | 25 | 3 | 50 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 1 | 3 | 25 | 50 |

Step 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 1 | 3 | 25 | 50 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 10 | 3 | 25 | 50 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 10 | 25 | 50 |

Step 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 10 | 25 | 50 |

swap

swap

swap

Result of Step 1

swap

swap

Result of Step 2

swap

swap

Result of Step 3

Final Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 10 | 25 | 50 |

# Figure 2 Bubble Sort Illustration

## Selection Sort

Selection sort method combines sorting and searching process at once. This method fixes the bubble sort algorithm by decreasing the amount of swapping process. This method will search for smallest value in the list and swap from there. This repeats until the last data exist in the list. Sorting process with Selection sort is illustrated below:

Initial Data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10 | 50 | 25 | 1 | 3 |

K= smallest value in the list before sorted

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | Swap | A[0] | A[1] | A[2] | A[3] | A[4] |
|  | Initial Data | 10 | 50 | 25 | 1 | 3 |
| 1 | M=A[0], k= 1 | 10 | 50 | 25 | 1 | 3 |
| 2 | M=A[1], k=3 | 1 | 30 | 25 | 10 | 3 |
| 3 | M=A[2], k=10 | 1 | 3 | 25 | 10 | 50 |
| 4 | M=A[3], k=50 | 1 | 3 | 10 | 25 | 50 |

# Figure 3 Selection Sort Illustration

## Insertion Sort

Insertion sort method is done by inserting a data in its expected position. The steps are as follows:

1. Get a data with nth index, save the value in *temp* variable (*i* starts from 2)
2. Compare the value from data *temp* with data in the left side, respectively.
3. Check if the data *temp* is smaller with data in the left side
4. If 3rd step is true, then swap the data one by one until its position is matched. The swapping process still applies the smaller and bigger value comparison.
5. Repeat step 1 – 4, so that the value of *i* is equal to the last data in the list.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Initial Data | 10 | 50 | 25 | 1 | 3 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Step | *temp* value | A[0] | A[1] | A[2] | A[3] | A[4] |
|  | Initial Data | 10 | 50 | 25 | 1 | 3 |
| 1 | temp=A[0] | 10 | 50 | 25 | 1 | 3 |
| 2 | temp=A[1] | 10 | 25 | 50 | 1 | 3 |
| 3 | temp=A[2] | 1 | 10 | 25 | 50 | 3 |
| 4 | temp=A[3] | 1 | 3 | 10 | 25 | 50 |

# Figure 4 Insertion Sort Illustration

## Practicum Steps

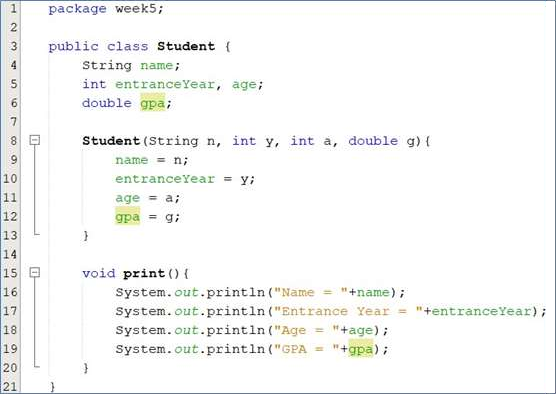
In this practicum of this jobsheet, we will create a software that can sort student’s data based on their GPA. In 1st practicum we will create a Students class as the blueprint we will use throughout this jobsheet. In 2nd practicum, we will create HighAchieverStudent class in which this class will be added the high score students as the data. Furthermore, there will be a method to sort students by their GPA in descending order. Lastly, in the 3rd practicum will be created main class to execute all the steps mentioned before.

## Practicum 1 – Create Student Class

* + 1. Pay attention in following class diagram. This will be used as our reference when creating Students Class.

|  |
| --- |
| Students |
| name: String entryYear: int age: int  gpa: double |
| Students(n: String, t: int, u: int, i: double) displayInfo(): void |

* + 1. Create **Students** class as follows!

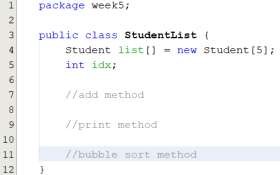


## Practicum 2 – Create HighAchieverStudent Class

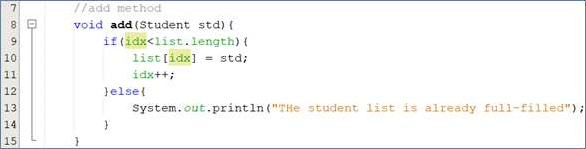
* + 1. In HighAchieverStudent Class, create a list and a method to sort the student’s data based on their GPA. In addition, create a method to display all those data and other method to insert the data to the list. Take a look at class diagram below!

|  |
| --- |
| HighAchieverStudent |
| list: Students[5] idx: int |
| add(std: Students): void display(): void  bubbleSort(): void |

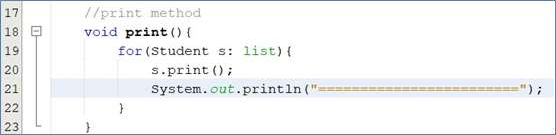
* + 1. Create **HighAchieverStudent** class as follows!



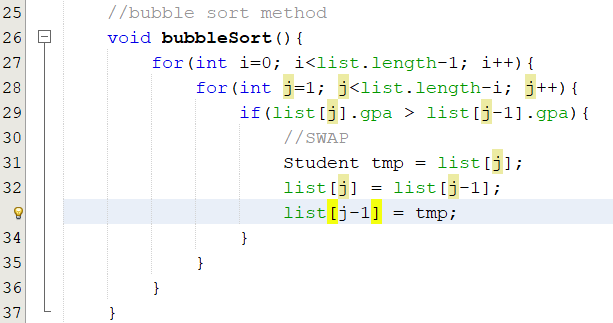
* + 1. Add method add() in that class. This method will be used to add an object from

**Students** class to listStd attribute.

* + 1. Add method display() in that class. This method will be used to display all the data that is exist in listStd. Take a look on how we use *for* loops, even though it is quite different than usual, the concept is still similar. Instead of accessing each element by its index, we just loop by each element available in the list.



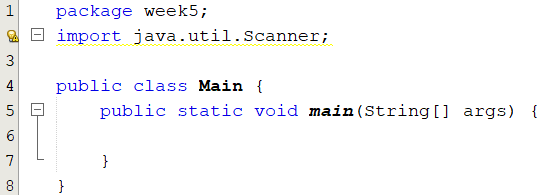
* + 1. Add method bubbleSort() in that class.



* + 1. Up to this point, the HighAchieverStudent class is finished

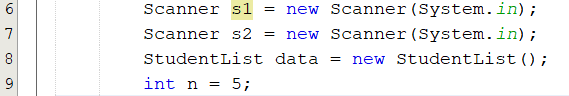
## Practicum 3 – Create Main Class

* + 1. Create main class and its main method

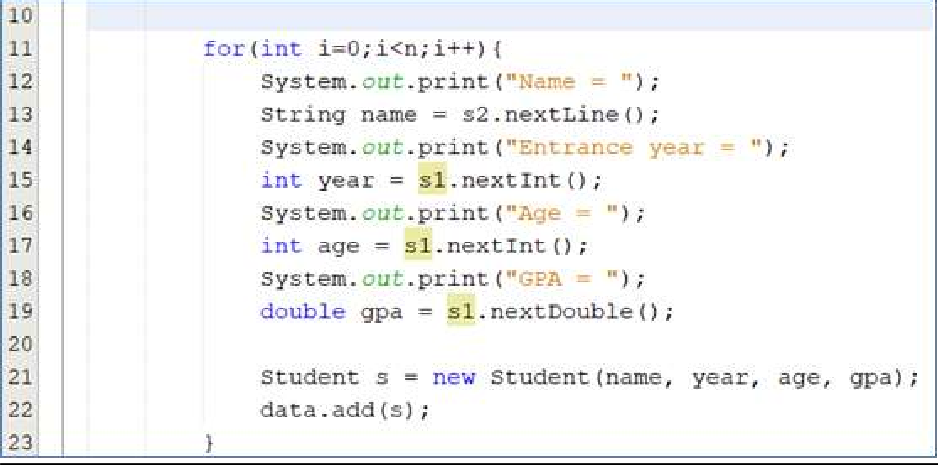


* + 1. In main method, create 2 objects from **Scanner** class, and an object from

**HighAchieverStudent** class. After that, declare the variable of amountStd to 5!



* + 1. Make loops 5 times with *for,* to insert name, age, entryYear, and GPA foreach students. Once it is done, instantiate the object from **Students** class and insert it to the list in **HighAchieverStudent** class.



* + 1. Display the student’s data that has been inserted in the list!



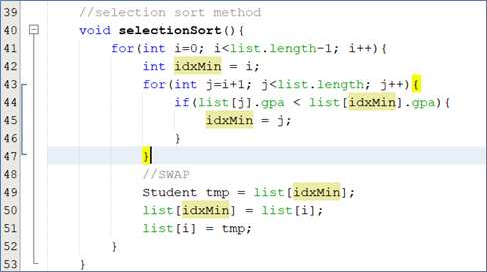
* + 1. Call method bubbleSort() and show the result!



Try to execute the program and understand the result. Are the data in the list is sorted based on GPA?

## Practicum 4 – Add Selection Sort process in HighAchieverStudent Class

* + 1. Go to **HighAchieverStudent** class, add method selectionSort() there. This method will do the sorting process in ascending order, but with selection sort approach.



* + 1. After that, open main class. In main method, add these line of code to execute selectionSort() method that we’ve just created.



Try to execute the program and understand the result. Are the data in the list is sorted based on GPA?

## Practicum 5 – Add Insertion Sort process in HighAchieverStudent Class

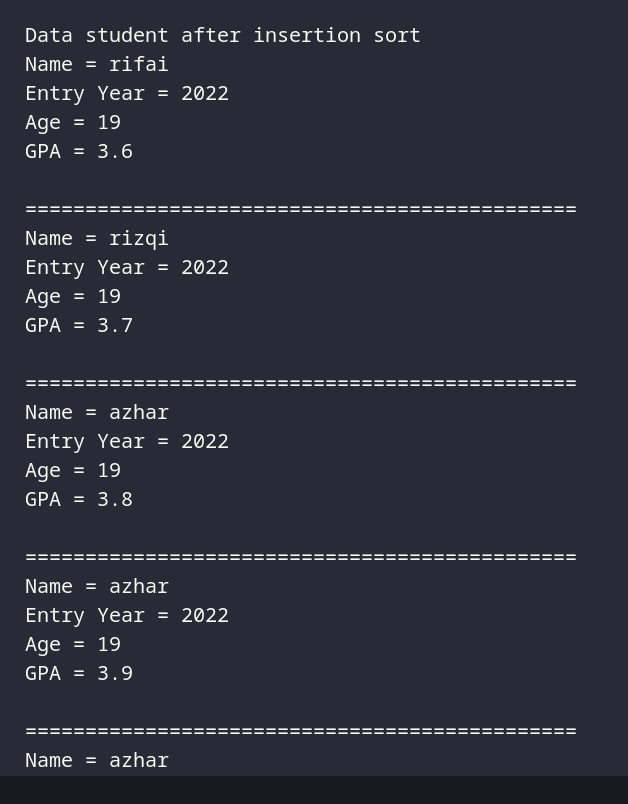
* + 1. Go to **HighAchieverStudent** class, add method insertionSort() there. This method will do the sorting process in ascending order, but with selection sort approach.



* + 1. After that, open main class. In main method, add these lines of code to execute insertionSort () method that we’ve just created.



Try to execute the program and understand the result. Are the data in the list is sorted based on GPA?



## Questions

1. In which class we have a function to do sorting with bubble sort approach?

At HighAchieverStudent class or StudentList class.

1. In which class we have a function to do sorting with insertion sort approach?

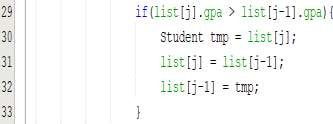
At HighAchieverStudent class or StudentList class.

1. What is the meaning of swapping process? Write the code to do the swapping process in the program above!



Swap is process of exchanging the positions of two elements in the array.

1. In bubbleSort(), there is these lines of code, what’s the function of it?



First, there is if statement that contain wheter list[j].gpa > list[j-1].gpa.then if it true, it will be swap process.

1. Look at the loops inside the bubbleSort() method:



* 1. What’s the difference of loop *i* and loop *j*?

Loop i is used to itterate the array and loop j is used to swap element inside array.

* 1. Why is the criteria of loop *i* is i<listStd.length-1?

because when the last element has been sorting, there is no need to compared again.

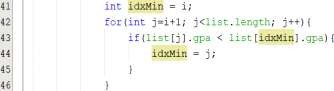
* 1. Why is the criteria of loop *j* is j<listStd.length-i?

it used -i because as reference to know that the last element has been sorted and there is no need to sorting again.

* 1. If the data in listStd is 50, how many loop *i* will happen? And how many bubble sort steps will be?

it will be 49+48+47+...+2+1 = 1225 itterations and have 49 bubble sort steps.

1. In selection sort method, there is these lines of code, what’s that for?



The code is used to find the index minimum value in the unsorted part of the array.

1. Change the insertionSort method so that the user has options to sort in either ascending or descending order. You can do it by adding a parameter, and this parameter’s value will be assigned through function calling in main class



## Assignment

* 1. There is a company that provide services in airplane ticket sales, they are developing a backend system for ticket reservation. One of its features is to display all available tickets based on filter from user. The ticket list must able to be sorted by the price in ascending and descending order. Implement these class diagrams in java program and create the sorting algorithm with **bubble sort** and **selection sort**

|  |
| --- |
| Tickets |
| airlines: String price: int  destination: String origin : String |
| Ticket(String a, String dest, String origin, int price): void |

|  |
| --- |
| TicketService |
| tickets : Ticket() |
| add(Ticket t) : void displayAll() : void bubbleSort() : void  selectionSort() : void |

|  |
| --- |
| MainTicket |
|  |
| Main(String[] abcd ): void |

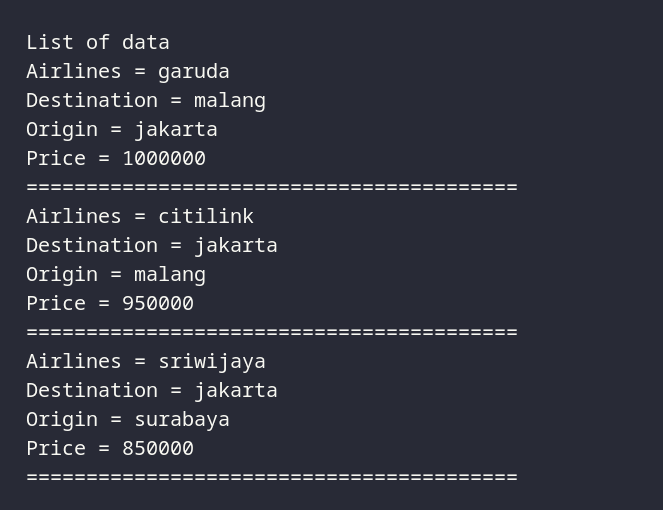
code :

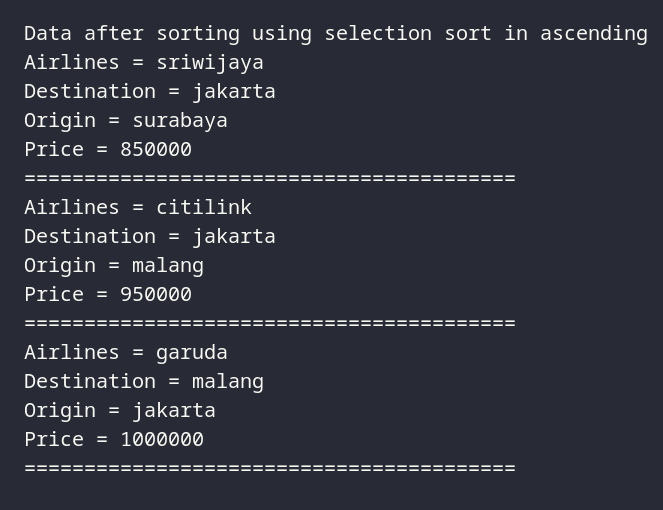


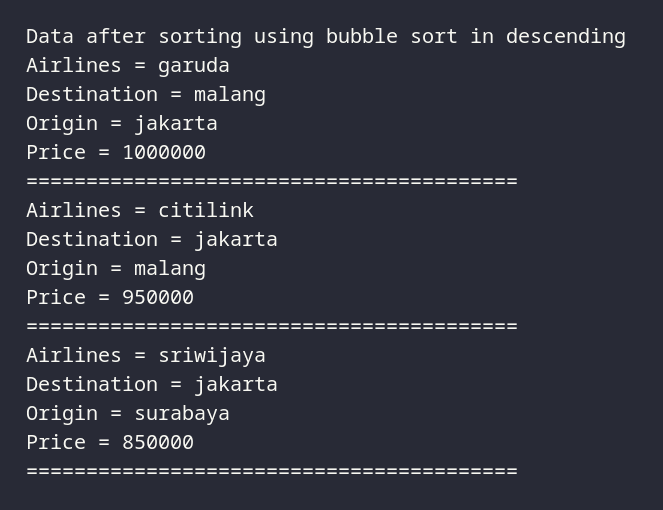




Result :







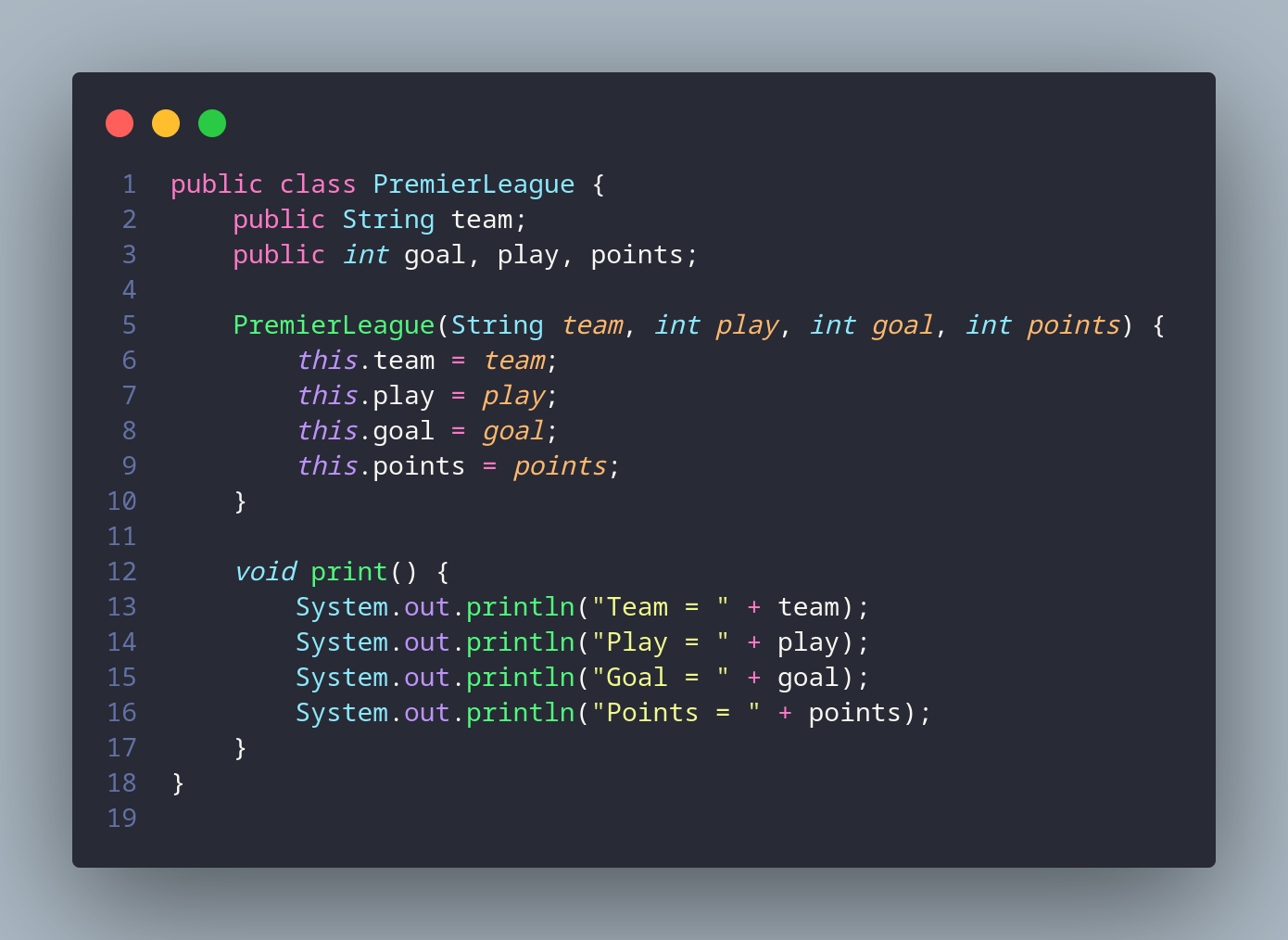
* 1. Premiere League in 2020 is already in half-season. In this season, Liverpool is the top of the list, the full list is displayed below

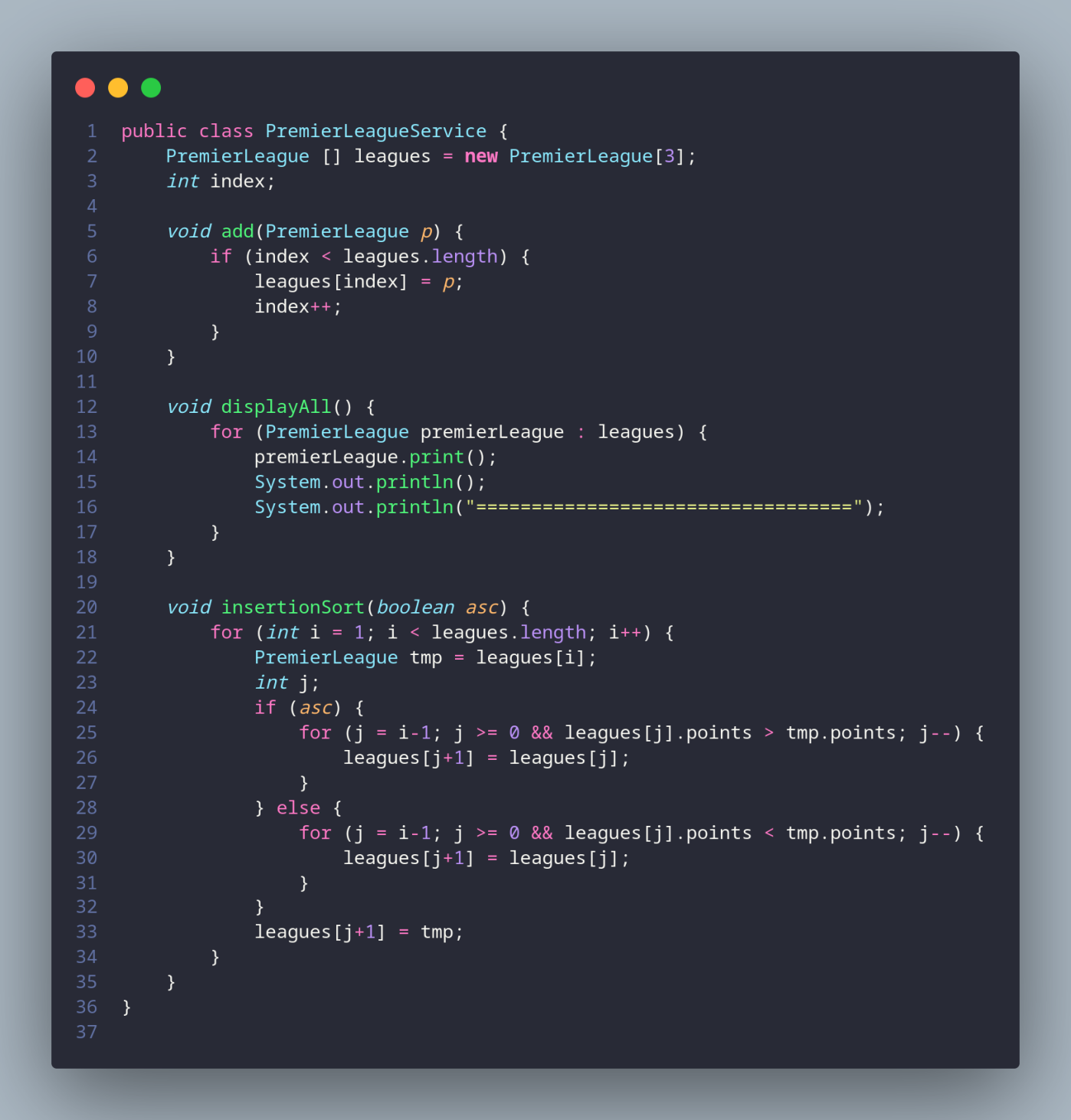


Change the standings list above to class diagram that has sorting club function based on highest to smallest points (in ascending order) with insertion sort algorithm. Take these following class diagrams as your reference:

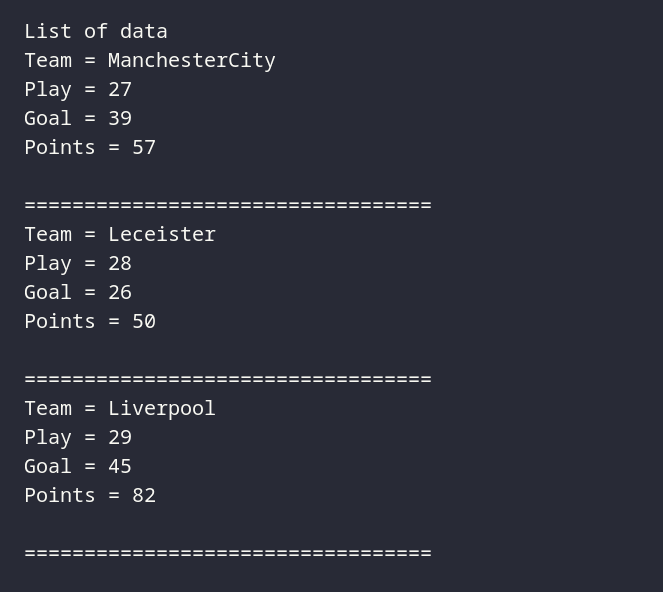
Code :

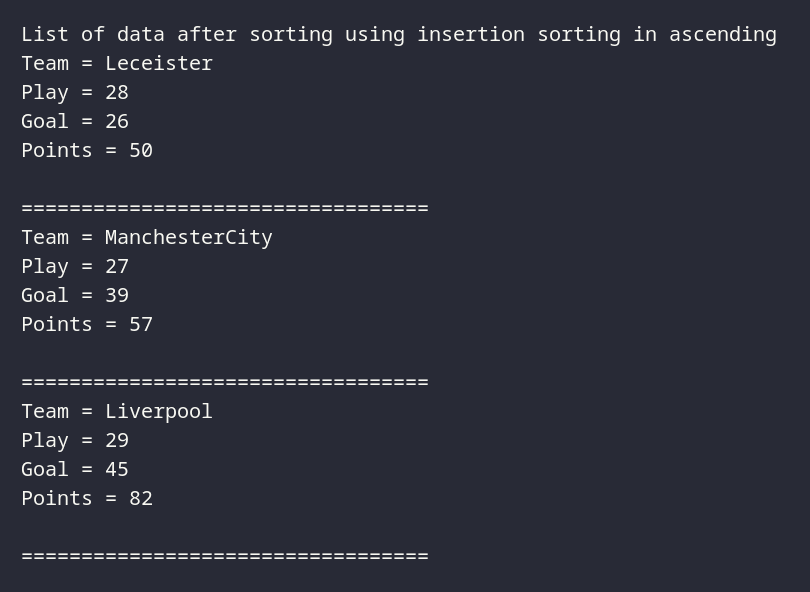


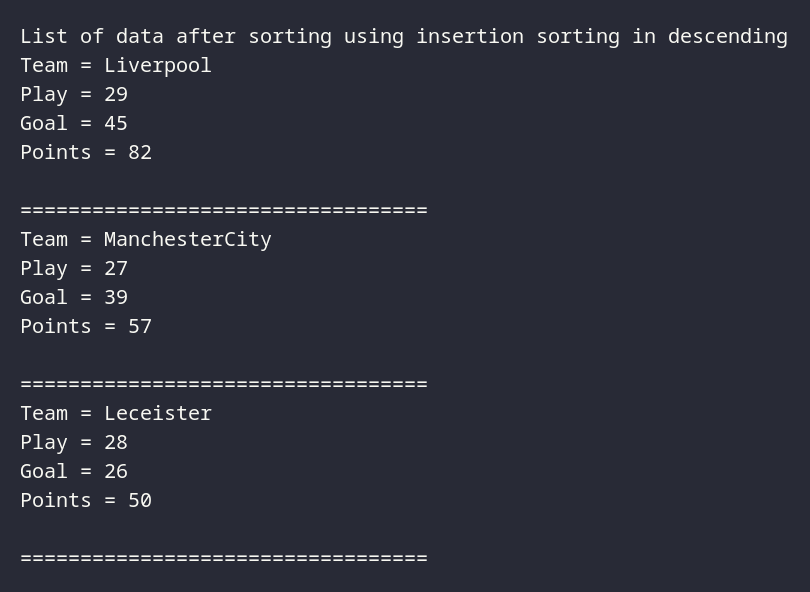




Result :







|  |  |  |
| --- | --- | --- |
| PremierLeague | |  |
| team: String play: int goal: String  points: String | |  |
| PremierLeague(String t, int p  ,int g, int pt): void | |  |
|  | MainLeague | |
|  |  | |
|  | Main(String[] abcd ): void | |

|  |
| --- |
| PremierLeagueService |
| leagues: PremierLeague() |
| add( PremierLeague p) : void displayAll() : void insertionSort(boolean asc) : void |