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:

1. **Learning Objective**
   a. Students are able to understand sorting algorithm
   b. Students are able create and declare the sorting algorithm
   c. Students are able to implement sorting algorithm
2. **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:

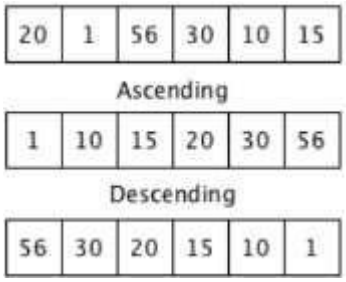

Figure 1 Sorting Process

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

1. **Bubble Sort**
2. **Selection Sort**
3. **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

10	50	25	1	3

Initial Data

Step 1

10	50	25	1	3

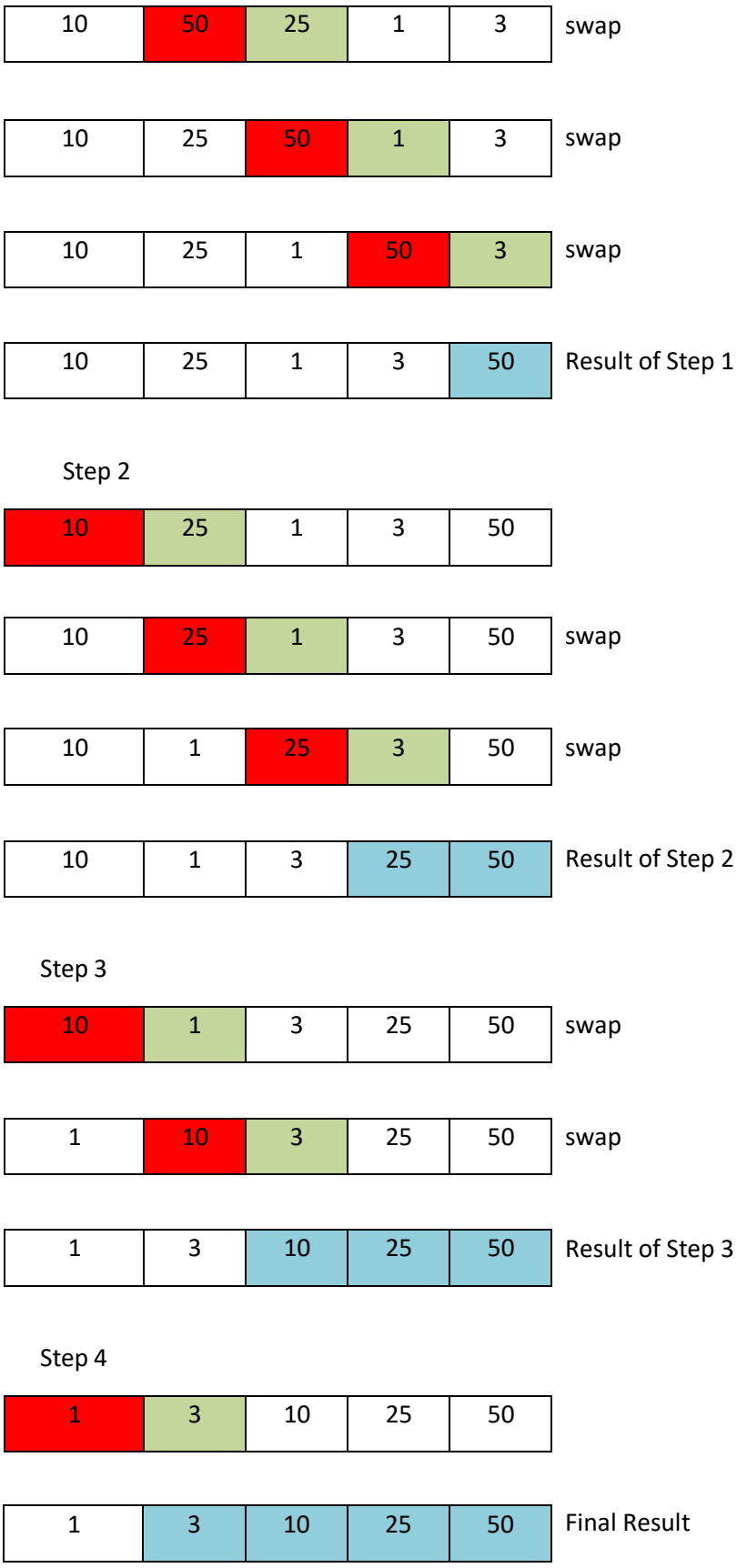

Figure 2 Bubble Sort Illustration

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

3. 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.

a. 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

2. Create **Students** class as follows!

```
1  package week5;
2
3  public class Student {
4      String name;
5      int entranceYear, age;
6      double gpa;
7
8      Student(String n, int y, int a, double g){
9          name = n;
10         entranceYear = y;
11         age = a;
12         gpa = g;
13     }
14
15     void print(){
16         System.out.println("Name = "+name);
17         System.out.println("Entrance Year = "+entranceYear);
18         System.out.println("Age = "+age);
19         System.out.println("GPA = "+gpa);
20     }
21 }
```

b. 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

2. Create **HighAchieverStudent** class as follows!

```
1  package week5;
2
3  public class StudentList {
4      Student list[] = new Student[5];
5      int idx;
6
7      //add method
8
9      //print method
10
11     //bubble sort method
12 }
```

3. Add method add() in that class. This method will be used to add an object from **Students** class to listStd attribute.

```
7      //add method
8      void add(Student std){
9          if(idx<list.length){
10             list[idx] = std;
11             idx++;
12         }else{
13             System.out.println("THe student list is already full-filled");
14         }
15     }
```

4. 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.

```
17     //print method
18     void print(){
19         for(Student s: list){
20             s.print();
21             System.out.println("==========================");
22         }
23     }
```

5. Add method bubbleSort() in that class.

```
25      //bubble sort method
26      void bubbleSort(){
27          for(int i=0; i<list.length-1; i++){
28              for(int j=1; j<list.length-i; j++){
29                  if(list[j].gpa > list[j-1].gpa){
30                      //SWAP
31                      Student tmp = list[j];
32                      list[j] = list[j-1];
                        list[j-1] = tmp;
34                  }
35              }
36          }
37      }
```

6. Up to this point, the HighAchieverStudent class is finished

c. Practicum 3 – Create Main Class

1. Create main class and its main method

```
1   package week5;
    import java.util.Scanner;
3
4   public class Main {
5       public static void main(String[] args) {
6
7       }
8   }
```

2. In main method, create 2 objects from **Scanner** class, and an object from **HighAchieverStudent** class. After that, declare the variable of amountStd to 5!

```
6               Scanner s1 = new Scanner(System.in);
7               Scanner s2 = new Scanner(System.in);
8               StudentList data = new StudentList();
9               int n = 5;
```

3. 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.

```
10
11          for(int i=0;i<n;i++){
12              System.out.print("Name = ");
13              String name = s2.nextLine();
14              System.out.print("Entrance year = ");
15              int year = s1.nextInt();
16              System.out.print("Age = ");
17              int age = s1.nextInt();
18              System.out.print("GPA = ");
19              double gpa = s1.nextDouble();
20
21              Student s = new Student(name, year, age, gpa);
22              data.add(s);
23          }
```

4. Display the student's data that has been inserted in the list!

```
24
25          System.out.println("Unsorted student list:");
26          data.print();
```

5. Call method bubbleSort() and show the result!

```
27          System.out.println("Data mahasiswa setelah sorting desc berdasar ipk= ");
28          data.bubbleSort();
29          data.tampil();
```

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

d. <u>Practicum 4 – Add Selection Sort process in HighAchieverStudent Class</u>

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

```
39      //selection sort method
40      void selectionSort(){
41          for(int i=0; i<list.length-1; i++){
42              int idxMin = i;
43              for(int j=i+1; j<list.length; j++){
44                  if(list[j].gpa < list[idxMin].gpa){
45                      idxMin = j;
46                  }
47              }
48              //SWAP
49              Student tmp = list[idxMin];
50              list[idxMin] = list[i];
51              list[i] = tmp;
52          }
53      }
```

2. After that, open main class. In main method, add these line of code to execute selectionSort() method that we've just created.

```
30          System.out.println("Ascending Sorted student list");
31          data.selectionSort();
32          data.print();
```

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

e. <u>Practicum 5 – Add Insertion Sort process in HighAchieverStudent Class</u>

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

```
void insertionSort() {
    for (int i = 1; i < list.length; i++) {
        Student tmp = list[i];
        int j;
        for (j = i-1; j >= 0 && list[j].gpa > tmp.gpa; j--) {
            list[j+1] = list[j];
        }
        list[j+1] = tmp;
    }
}
```

2. After that, open main class. In main method, add these lines of code to execute insertionSort () method that we've just created.

```
System.out.println("Data student after insertion sort");
data.insertionSort();
data.print();
```

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

```
Data student after insertion sort
Name = rifai
Entry Year = 2022
Age = 19
GPA = 3.6

================================================
Name = rizqi
Entry Year = 2022
Age = 19
GPA = 3.7

================================================
Name = azhar
Entry Year = 2022
Age = 19
GPA = 3.8

================================================
Name = azhar
Entry Year = 2022
Age = 19
GPA = 3.9

================================================
Name = azhar
```

4. Questions

1. In which class we have a function to do sorting with bubble sort approach?
   At HighAchieverStudent class or StudentList class.
2. In which class we have a function to do sorting with insertion sort approach?
   At HighAchieverStudent class or StudentList class.
3. What is the meaning of swapping process? Write the code to do the swapping process in the program above!

```
1  // swap
2  Student tmp = list[j];
3  list[j] = list[j-1];
4  list[j-1] = tmp;
```

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

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

```
29                  if(list[j].gpa > list[j-1].gpa){
30                      Student tmp = list[j];
31                      list[j] = list[j-1];
32                      list[j-1] = tmp;
33                  }
```

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

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

```
27          for(int i=0; i<list.length-1; i++){
28              for(int j=1; j<list.length-i; j++){
```

   a. 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.
   b. 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.
   c. 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.
   d. 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.

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

```
41              int idxMin = i;
42              for(int j=i+1; j<list.length; j++){
43                  if(list[j].gpa < list[idxMin].gpa){
44                      idxMin = j;
45                  }
46              }
```

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

7. 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

```java
void insertionSort(boolean asc) {
    for (int i = 1; i < list.length; i++) {
        Student tmp = list[i];
        int j;
        if (asc) {
            for (j = i-1; j >= 0 && list[j].gpa > tmp.gpa; j--) {
                list[j+1] = list[j];
            }
        } else {
            for (j = i-1; j >= 0 && list[j].gpa < tmp.gpa; j--) {
                list[j+1] = list[j];
            }
        }
        list[j+1] = tmp;
    }
}
```

5. 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 :

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        TicketService data = new TicketService();
        int number = 3;
        for (int i = 0; i < number; i++) {
            System.out.print("Insert name of Airlines : ");
            String airlines = scanner.next();
            System.out.print("Insert Destination : ");
            String destination = scanner.next();
            System.out.print("Insert Origin : ");
            String origin = scanner.next();
            System.out.print("Insert Price : ");
            int price = scanner.nextInt();

            Tickets tickets = new Tickets(airlines, destination, origin, price);
            data.add(tickets);
        }
        System.out.println();

        System.out.println("List of data");
        data.displayAll();
        System.out.println();

        System.out.println("Data after sorting using bubble sort in ascending");
        data.bubbleSort();
        data.displayAll();
        System.out.println();

        System.out.println("Data after sorting using selection sort in descending");
        data.selectionSort();
        data.displayAll();

        scanner.close();
    }
}

```

```java
public class Tickets {
    public String airlines, destination, origin;
    public int price;

    Tickets(String airlines, String destination, String origin, int price) {
        this.airlines = airlines;
        this.destination = destination;
        this.origin = origin;
        this.price = price;
    }

    void print() {
        System.out.println("Airlines = " + airlines);
        System.out.println("Destination = " + destination);
        System.out.println("Origin = " + origin);
        System.out.println("Price = " + price);
    }
}

```

```java
public class TicketService {
    Tickets [] tickets = new Tickets[3];
    int index;

    void add(Tickets ticket) {
        if (index < tickets.length) {
            tickets[index] = ticket;
            index++;
        }
    }

    void displayAll() {
        for (Tickets ticket : tickets) {
        ticket.print();
        System.out.println("========================================");
        }
    }

    void bubbleSort() {
        for (int i = 0; i < tickets.length-1; i++) {
            for (int j = 1; j < tickets.length-i; j++) {
                if (tickets[j].price > tickets[j-1].price) {
                    Tickets tmp = tickets[j];
                    tickets[j] = tickets[j-1];
                    tickets[j-1] = tmp;
                }
            }
        }
    }

    void selectionSort() {
        for (int i = 0; i < tickets.length-1; i++) {
            int indexMin = i;
            for (int j = i+1; j < tickets.length; j++) {
                if (tickets[j].price < tickets[indexMin].price) {
                    indexMin = j;
                }
                Tickets tmp = tickets[indexMin];
                tickets[indexMin] = tickets[i];
                tickets[i] = tmp;
            }
        }
    }
}

```

Result :

```
List of data
Airlines = garuda
Destination = malang
Origin = jakarta
Price = 1000000
========================================
Airlines = citilink
Destination = jakarta
Origin = malang
Price = 950000
========================================
Airlines = sriwijaya
Destination = jakarta
Origin = surabaya
Price = 850000
========================================

Data after sorting using selection sort in ascending
Airlines = sriwijaya
Destination = jakarta
Origin = surabaya
Price = 850000
========================================
Airlines = citilink
Destination = jakarta
Origin = malang
Price = 950000
========================================
Airlines = garuda
Destination = malang
Origin = jakarta
Price = 1000000
========================================
```

```
Data after sorting using bubble sort in descending
Airlines = garuda
Destination = malang
Origin = jakarta
Price = 1000000
========================================
Airlines = citilink
Destination = jakarta
Origin = malang
Price = 950000
========================================
Airlines = sriwijaya
Destination = jakarta
Origin = surabaya
Price = 850000
========================================
```

2. 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

	Premier League	P	GD	PTS
1	Liverpool	29	45	82
2	Manchester City	27	39	57
3	Leicester	28	26	50
4	Chelsea	29	9	48
5	Wolverhampton Wanderers	29	7	43
6	Sheffield United	28	5	43
7	Manchester United	28	12	42
8	Tottenham Hotspur	29	7	41
9	Arsenal	28	4	40
10	Burnley	29	-6	39
11	Crystal Palace	29	-8	39
12	Everton	29	-6	37
13	Newcastle United	29	-16	35
14	Southampton	29	-17	34
15	Brighton & Hove Albion	29	-8	29
16	West Ham United	29	-15	27
17	Watford	29	-17	27
18	AFC Bournemouth	29	-18	27
19	Aston Villa	27	-18	25
20	Norwich City	29	-27	21

Manchester City

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 :

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        PremierLeagueService data = new PremierLeagueService();
        int number = 3;

        for (int i = 0; i < number; i++) {
            System.out.print("Insert team : ");
            String team = scanner.next();
            System.out.print("Insert Play : ");
            int play = scanner.nextInt();
            System.out.print("Insert goal : ");
            int goal = scanner.nextInt();
            System.out.print("Insert points : ");
            int points = scanner.nextInt();

            PremierLeague premierLeague = new PremierLeague(team, play, goal, points);
            data.add(premierLeague);
        }
        System.out.println();

        System.out.println("List of data");
        data.displayAll();
        System.out.println();

        System.out.println("List of data after sorting using insertion sorting in ascending");
        data.insertionSort(true);
        data.displayAll();
        System.out.println();

        System.out.println("List of data after sorting using insertion sorting in descending");
        data.insertionSort(false);
        data.displayAll();

        scanner.close();
    }
}

```

```java
public class PremierLeague {
    public String team;
    public int goal, play, points;

    PremierLeague(String team, int play, int goal, int points) {
        this.team = team;
        this.play = play;
        this.goal = goal;
        this.points = points;
    }

    void print() {
        System.out.println("Team = " + team);
        System.out.println("Play = " + play);
        System.out.println("Goal = " + goal);
        System.out.println("Points = " + points);
    }
}

```

```java
public class PremierLeagueService {
    PremierLeague [] leagues = new PremierLeague[3];
    int index;

    void add(PremierLeague p) {
        if (index < leagues.length) {
            leagues[index] = p;
            index++;
        }
    }

    void displayAll() {
        for (PremierLeague premierLeague : leagues) {
            premierLeague.print();
            System.out.println();
            System.out.println("==================================");
        }
    }

    void insertionSort(boolean asc) {
        for (int i = 1; i < leagues.length; i++) {
            PremierLeague tmp = leagues[i];
            int j;
            if (asc) {
                for (j = i-1; j >= 0 && leagues[j].points > tmp.points; j--) {
                    leagues[j+1] = leagues[j];
                }
            } else {
                for (j = i-1; j >= 0 && leagues[j].points < tmp.points; j--) {
                    leagues[j+1] = leagues[j];
                }
            }
            leagues[j+1] = tmp;
        }
    }
}

```

Result :

```
List of data
Team = ManchesterCity
Play = 27
Goal = 39
Points = 57

====================================
Team = Leceister
Play = 28
Goal = 26
Points = 50

====================================
Team = Liverpool
Play = 29
Goal = 45
Points = 82

====================================
```

```
List of data after sorting using insertion sorting in ascending
Team = Leceister
Play = 28
Goal = 26
Points = 50

================================
Team = ManchesterCity
Play = 27
Goal = 39
Points = 57

================================
Team = Liverpool
Play = 29
Goal = 45
Points = 82

================================
```

```
List of data after sorting using insertion sorting in descending
Team = Liverpool
Play = 29
Goal = 45
Points = 82

==================================
Team = ManchesterCity
Play = 27
Goal = 39
Points = 57

==================================
Team = Leceister
Play = 28
Goal = 26
Points = 50

==================================
```

PremierLeague
team: String play: int goal: String points: String
PremierLeague(String t, int p ,int g, int pt): void

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

MainLeague
Main(String[] abcd): void