JOBSHEET VII
SEARCHING

7.1 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 algorithms

7.2 Sequential Search Method

Look at the 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 an 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, adding a student, displaying all student's information, to search by NIM, and to display searched student's data later on

7.1 Steps

1. Open your project
2. Then, create a new package named **week7.**
3. Create new **Students** class, then declare following attributes:

```
public class Students {
    int nim, age;
    String name;
    double gpa;
```

4. Create a constructor in **Students** class with parameters (int ni, String nm, int age, double gpa), as follows:

```
public Students(int nim, int age, String name, double gpa) {
    this.nim = nim;
    this.age = age;
    this.name = name;
    this.gpa = gpa;
}
```

5. Create **display()** method with void as its return type

```
public void display(){
    System.out.println("NIM = " + nim);
    System.out.println("Name = " + name);
    System.out.println("Age = " + age);
    System.out.println("GPA= " + gpa);
}
```

6. Create a new **SearchStudent** class as follows.

```
public class SearchStudent {
    Students[] listStd = new Students[5];
    int idx;
```

7. Create method **add()** at that class! This will be used for adding objects from **Students** class to **listStd** attribute

```
public void add(Students std){
    if(idx < listStd.length){
        listStd[idx] = std;
        idx++;
    }else{
        System.out.println("Data is already full");
    }
}
```

8. Create method **display()** in class **SearchStudent!** This **display()** method will be used to print all student data available in this class. Pay attention on how we use **for-loops** differently (popularly known as **foreach**). Even so, the concepts are still the same

```
public void display(){
    for (Students students : listStd) {
        students.display();
        System.out.println("----------------------------");
    }
}
```

9. Create method **findSeqSearch()** with integer as its return type. Then fill in the function with sequential search algorithm.

```
public int findSeqSearch(int search){
    int position = -1;
    for (int i = 0; i < listStd.length; i++) {
        if(listStd[i].nim == search){
            position = i;
            break;
        }
    }
    return position;
}
```

10. Create method **showPosition()** with void as its return type. And write the following code

```
public void showPosition(int x, int pos){
    if(pos != -1){
        System.out.println("Data : "+ x + " is found in index-"+pos);
    }else{
        System.out.println("Data : " + x + " is not found");
    }
}
```

11. Create method **showData()** with void as its return type. And write the following code

```
public void showData(int x, int pos){
    if(pos != -1){
        System.out.println("NIM \t : " + x);
        System.out.println("Name \t : " + listStd[pos].name);
        System.out.println("Age \t : " + listStd[pos].age);
        System.out.println("IPK \t : " + listStd[pos].gpa);
    }else{
        System.out.println("Data " + x + " is not found");
    }
}
```

12. Create a main class named **MainStudent** and add main method as follows

```
public class MainStudent {
    public static void main(String[] args) {

    }
}
```

13. In main method, instantiate an object in **SearchStudent** that consists of 5 **Students,** then add all students object by calling **add()** function in object **SearchStudent**

```
Scanner s = new Scanner(System.in);
Scanner sl = new Scanner(System.in);

SearchStudent data = new SearchStudent();
int amountStudent = 5;

System.out.println("---------------------------------");
System.out.println("Input student data accordingly from smallest NIM")
for (int i = 0; i < amountStudent; i++) {
    System.out.println("----------");
    System.out.print("NIM\t:");
    int nim = s.nextInt();
    System.out.print("Name\t:");
    String name = sl.nextLine();
    System.out.print("Age\t:");
    int age = s.nextInt();
    System.out.print("GPA\t:");
    double gpa = s.nextDouble();

    Students std = new Students(nim, age, name, gpa);
    data.add(std);
}
```

14. Add method **display()** to print all inserted data

```
System.out.println("-------------------------");
System.out.println("Entire Student Data");
data.display();
```

15. 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

```
System.out.println("_________________________");
System.out.println("_________________________");
System.out.print("Search student by NIM: ");
int search = s.nextInt();
System.out.println("Using Sequential Search");
int position = data.findSeqSearch(search);
```

16. Call method **showPosition()** from class **SearchStudent.**

```
data.showPosition(search, position);
```

17. Call method **showData()** from class **SearchStudent**

```
data.showData(search, position);
```

18. Run the program and see the result

7.2 Result

Match the output of your program code with following image

```
run:
--------------------------------
Input student data accordingly from smallest NIM
----------
NIM     :2017
Name    :Dewi Lestari
Age     :23
GPA     :3.5
----------
NIM     :2018
Name    :Sinta Sanjaya
Age     :22
GPA     :4
----------
NIM     :2019
Name    :Danang Adi
Age     :22
GPA     :3.7
----------
NIM     :2020
Name    :Budi Prakarsa
Age     :20
GPA     :2.9
NIM     :2021
Name    :Vania Siti
Age     :20
GPA     :3.0
--------------------------
Entire Student Data
NIM = 2017
Name = Dewi Lestari
Age = 23
GPA= 3.5
------------------------------
NIM = 2018
Name = Sinta Sanjaya
Age = 22
GPA= 4.0
------------------------------
NIM = 2019
Name = Danang Adi
Age = 22
GPA= 3.7
------------------------------
```

```
NIM = 2020
Name = Budi Prakarsa
Age = 20
GPA= 2.9
-----------------------------
NIM = 2021
Name = Vania Siti
Age = 20
GPA= 3.0
-----------------------------

-------------------------
-------------------------
Search student by NIM: 2018
Using Sequential Search
Data : 2018 is found in index-1
NIM       : 2018
Name      : Sinta Sanjaya
Age       : 22
IPK       : 4.0
BUILD SUCCESSFUL (total time: 1 minute 50 seconds)
```

7.3 Question

1. What is the difference between the methods **displayData** and **displayPosition** in **StudentSearch** class?
2. What is the function of break in this following program code?

```
if(listStd[i].nim == search){
    position = i;
    break;
}
```

3. If the NIM data inserted is not sorted from smallest to biggest value, will the program encounter an error? Is the result still correct? Why is that?
4. Look at **findSeqSearch** method, why position is initialized by -1 instead of 0?

7.3 Binary Search Method

7.3.1. Steps

1. Still in **SearchStudent** class, add a new method **findBinarySearch()** with integer as its data type**.** Then declare the content of method **findBinarySearch** with using binary search as its searching algorithm

```
public int FindBinarySearch(int cari, int left, int right) {
    int mid;
    if (right >= left) {
        mid = (left + right) / 2;
        if (cari == listMHs[mid].nim) {
            return (mid);
        } else if (listMHs[mid].nim > cari) {
            return FindBinarySearch(cari, left, mid - 1);
        } else {
            return FindBinarySearch(cari, mid + 1, right);
        }
    }
    return -1;
}
```

2. Call method **findBinarySearch()** from **SearchStudent** class in **StudentsMain**. Then call method **showPosition()** and **showData()**

```
System.out.println("======================");
System.out.print("Search student by NIM: ");
System.out.println("Using binary Search");
int position1 = data.findBinarySearch(search,0, amountStudent -1);

data.showPosition(search, position1);
data.showData(search, position1);
```

3. Run and see the result

7.2 Result

Match the output of your program code with following image

```
run:
--------------------------------
Input student data accordingly from smallest NIM
----------
NIM      :2017
Name     :Dewi Lestari
Age      :23
GPA      :3.5
----------
NIM      :2018
Name     :Sinta Sanjaya
Age      :22
GPA      :4
----------
NIM      :2019
Name     :Danang Adi
Age      :22
GPA      :3.7
----------
NIM      :2020
Name     :Budi Prakarsa
Age      :20
GPA      :2.9
NIM      :2021
Name     :Vania Siti
Age      :20
GPA      :3.0
--------------------------
Entire Student Data
NIM = 2017
Name = Dewi Lestari
Age = 23
GPA= 3.5
------------------------------
NIM = 2018
Name = Sinta Sanjaya
Age = 22
GPA= 4.0
------------------------------
NIM = 2019
Name = Danang Adi
Age = 22
GPA= 3.7
------------------------------
Search student by NIM: Using binary Search
Data : 2018 is found in index-1
NIM       : 2018
Name      : Sinta Sanjaya
Age       : 22
IPK       : 4.0
BUILD SUCCESSFUL (total time: 1 minute 21 seconds)
```

7.3 Question

1. Show the program code in which runs the divide process!
2. Show the program code in which runs the conquer process!
3. If inserted NIM data is not sorted, will the program give the correct result? 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
4. Modify program above so that the students amount inserted is matched with user input

7.4 Review Divide and Conquer

7.4.1. Steps

1. Add a new class **MergeSort**
2. In this class, we will create method **mergeSort()** that receives an array in its parameter

```
public void mergeSort(int[] data){

}
```

3. Create **merge()** method to do data merging process from left side to the right

```
private void merge(int data[], int left, int mid, int right){

}
```

4. Implement **merge()** process as follows:

```
public void merge(int data[], int left, int middle, int right) {
    int[] temp = new int[data.length];
    for (int i = left; i <= right; i++) {
        temp[i] = data[i];
    }
    int a = left;
    int b = middle + 1;
    int c = left;
      //membandingkan setiap bagian
      while (a <= middle && b <= right) {
          if (temp[a] <= temp[b]) {
              data[c] = temp[a];
              a++;
          } else {
              data[c] = temp[b];
              b++;
          }
          c++;
      }
      int s = middle - a;
      for (int i = 0; i <= s; i++) {
          data[c + i] = temp[a + i];
      }
  }
```

5. Create sort method

```
private void sort(int data[], int left, int right){

}
```

6. Implement these following codes in sort method

```
// DIvide into 2 parts and divide it again until no more thing to be divided
private void sort(int data[], int left, int right){
    if(left < right){
        int mid = (left + right) /2;
        sort(data,left,mid);
        sort(data, mid+1, right);
        merge(data, left, mid, right);
    }
}
```

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

```
public void printArray(int arr[]){
    int n = arr.length;
    for (int i = 0; i < n; i++) {
        System.out.println(arr[i]+" ");
    }
    System.out.println();
}
```

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

7.2 Result

Match the output of your program code with following image

```
run:
Sorting with merge sort
Initial Data
10 40 30 50 70 20 100 90
Sorted Data
10 40 30 50 70 20 100 90
BUILD SUCCESSFUL (total time: 0 seconds)
```

7.5 Assignments

1. In the previous practical session on Jobsheet 5, which included three classes—**Lecturer<attendance no>, LecturerData< attendance no>,** and **LecturerDemo< attendance no >**—add the following methods:
    a. **sequentialSearch()**: Used to search for lecturer data by name using the Sequential Search algorithm.
    b. **binarySearch()**: Used to search for lecturer data by age using the Binary Search algorithm.

    Establish a rule to detect and display a warning message if the search returns multiple results! Ensure that the implemented algorithm aligns with the given case!