

LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

BAB : COLLECTION SORTING AND SEARCHING
 NAMA : DANI ADRIAN
 NIM : 225150201111009
 ASISTEN : Tengku Muhammad Rafi Rahardiansyah
 Muhammad Bin Djafar Almasyhur
 TGL PRAKTIKUM : 24 Mei 2023

BAB 10**A. COLLECTION SORTING DAN SEARCHING****Pelaksanaan Percobaan****A. Sort****Pertanyaan**

1. Ketikkan kode program dibawah ini dan analisis output dari program tersebut!

SetExample.java

```

1. // Java program to demonstrate working of Comparator
2. // interface and Collections.sort() to sort according
3. // to user defined criteria.
4. import java.util.*;
5. import java.lang.*;
6. import java.io.*;
7.
8. // A class to represent a student.
9. class Student {
10.     int rollno;
11.     String name, address;
12.
13.     // Constructor
14.     public Student(int rollno, String name,
15.         String address) {
16.         this.rollno = rollno;
17.         this.name = name;
18.         this.address = address;
19.     }
20.
21.     // Used to print student details in main()
22.     public String toString() {
23.         return this.rollno + " " + this.name + " " +
24.         this.address;
25.     }

```

```

26.
27. class Sortbyroll implements Comparator<Student> {
28.     // Used for sorting in ascending order of
29.     // roll number
30.     public int compare(Student a, Student b) {
31.         return a.rollno - b.rollno;
32.     }
33. }
34.
35. // Driver class
36. class Main {
37.     public static void main(String[] args) {
38.         ArrayList<Student> ar = new ArrayList<Student>();
39.         ar.add(new Student(111, "bbbb", "london"));
40.         ar.add(new Student(131, "aaaa", "nyc"));
41.         ar.add(new Student(121, "cccc", "jaipur"));
42.
43.         System.out.println("Unsorted");
44.         for (int i = 0; i < ar.size(); i++)
45.             System.out.println(ar.get(i));
46.
47.         Collections.sort(ar, new Sortbyroll());
48.
49.         System.out.println("\nSorted by rollno");
50.         for (int i = 0; i < ar.size(); i++)
51.             System.out.println(ar.get(i));
52.     }
53. }

```

Source Code

- | | |
|----|---|
| 1. | Tulis source code di sini pake courier new 12 |
|----|---|

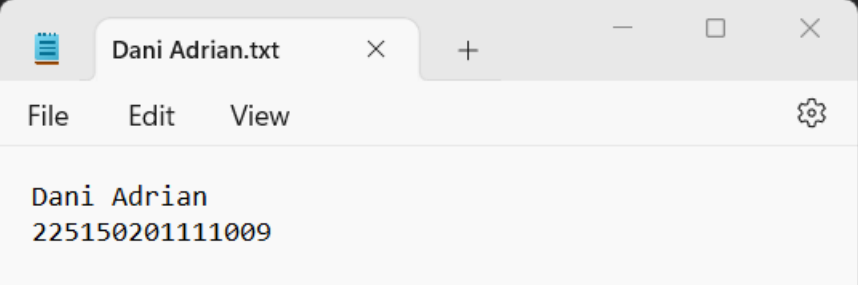
Output

```

Unsorted
111 bbbb london
131 aaaa nyc
121 cccc jaipur

Sorted by rollno
111 bbbb london
121 cccc jaipur
131 aaaa nyc
PS C:\Users\USER\JAVA VS CODE\OOP\praktikumkesebelas>

```



Penjelasan

Program ini menggunakan Comparator dan Collections.sort() untuk mengurutkan objek Student berdasarkan nomor roll (rollno). Pada awalnya, terdapat sebuah ArrayList bernama "ar" yang berisi daftar siswa yang belum diurutkan dan ditampilkan sesuai urutan penambahan. Kemudian, menggunakan metode Collections.sort() dengan mengirimkan Comparator objek bernama Sortbyroll, daftar siswa diurutkan secara naik (ascending) berdasarkan nomor roll (dari yang terkecil ke terbesar). Setelah diurutkan, daftar siswa ditampilkan kembali sesuai urutan yang baru. Dalam contoh ini, daftar siswa telah diurutkan berdasarkan nomor roll dengan urutan 111, 121, dan 131. Dengan demikian, program ini mengilustrasikan penggunaan Comparator dan Collections.sort() untuk mengurutkan objek sesuai dengan kriteria yang ditentukan pengguna, dalam hal ini berdasarkan nomor roll siswa.

B. Searching

Pertanyaan

1. Ketikkan kode program dibawah ini dan analisis output dari program tersebut!

MapExample.java	
1.	// Java Program to Demonstrate Working of binarySearch()
2.	// method of Collections Class
3.	
4.	// Importing required classes
5.	import java.util.ArrayList;
6.	import java.util.Collections;
7.	import java.util.List;

```

8.
9. // Main class
10. class GFG {
11.     // Main driver method
12.     public static void main(String[] args) {
13.         // Creating an empty ArrayList of integer type
14.         List<Integer> al = new ArrayList<Integer>();
15.
16.         // Populating the ArrayList
17.         al.add(1);
18.         al.add(2);
19.         al.add(3);
20.         al.add(10);
21.         al.add(20);
22.
23.         // 10 is present at index 3.
24.         int key = 10;
25.         int res = Collections.binarySearch(al, key);
26.
27.         if (res >= 0)
28.             System.out.println(key + " found at index =
29. " + res);
30.         else
31.             System.out.println(key + " Not found");
32.
33.         key = 15;
34.         res = Collections.binarySearch(al, key);
35.
36.         if (res >= 0)
37.             System.out.println(key + " found at index =
38. " + res);
39.         else
40.             System.out.println(key + " Not found");
41.     }
42. }

```

Source Code

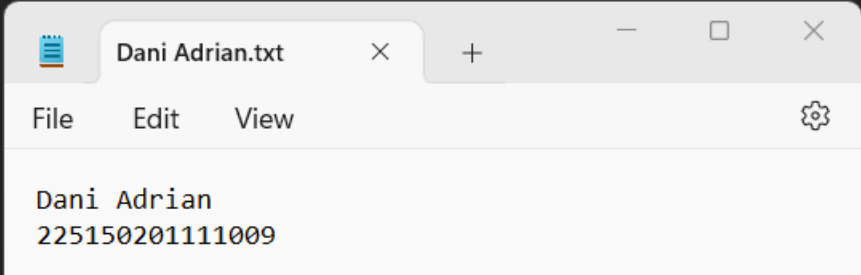
1.	Tulis source code di sini pake courier new 12
----	---

Output

```

10 found at index = 3
15 Not found
PS C:\Users\USER\JAVA VS CODE\OOP\praktikumkesebelas>

```



Penjelasan

Program ini menggunakan metode `binarySearch()` dari kelas `Collections` untuk mencari elemen tertentu dalam sebuah `ArrayList` yang sudah diurutkan. Pada awalnya, kita membuat `ArrayList` kosong bernama "al" dan mengisinya dengan elemen 1, 2, 3, 10, dan 20. Kemudian, kita mencari elemen dengan nilai 10 menggunakan `Collections.binarySearch(al, key)`. Hasil pencarian (res) adalah 3, yang menunjukkan bahwa nilai 10 ditemukan pada indeks ke-3 dalam `ArrayList`. Selanjutnya, kita mencari elemen dengan nilai 15. Karena 15 tidak ada dalam `ArrayList`, hasil pencarian (res) akan bernilai negatif (-5). Ini menunjukkan bahwa elemen dengan nilai 15 tidak ditemukan dalam `ArrayList`. Dalam kedua kasus, program mencetak pesan yang sesuai, yaitu "10 ditemukan pada indeks = 3" dan "15 Tidak ditemukan". Dengan demikian, program tersebut menjelaskan penggunaan metode `binarySearch()` dari kelas `Collections` untuk mencari elemen dalam `ArrayList` yang telah diurutkan, dan memberikan indeks elemen tersebut jika ditemukan.

B. Tugas Praktikum

No	Tim A		TIM B	
	Tinggi Badan (cm)	Berat Badan (kg)	Tinggi Badan (cm)	Berat Badan (kg)
1.	168	50	170	66
2.	170	60	167	60
3.	165	56	165	59
4.	168	55	166	58
5.	172	60	168	58
6.	170	70	175	71
7.	169	66	172	68
8.	165	56	171	68

9.	171	72	168	65
10.	166	56	169	60

Pertanyaan

1. Dengan program java, carilah data pemain diantara kedua tim tersebut :
 - a. Berdasarkan berat badannya secara *ascending* / menaik dan *descending* / menurun
 - b. Berdasarkan berat badannya secara *ascending* / menaik dan *descending* / menurun
 - c. Cari nilai maksimum dan minimum tinggi badan dan berat badan untuk pemain dari masing-masing tim
 - d. *Copy* seluruh anggota tim B ke tim C yang baru dibentuk

Source Code

```

DataSetFutsal1.java
import java.util.*;

class datasetFutsal {
    int[] dataTinggiTimA = { 168, 170, 165, 168, 172, 170, 169,
165, 171, 166 };
    int[] dataBeratBadanTimA = { 50, 60, 56, 55, 60, 70, 66,
56, 72, 56 };
    int[] dataTinggiTimB = { 170, 167, 165, 166, 168, 175, 172,
171, 168, 169 };
    int[] dataBeratBadanTimB = { 66, 60, 59, 58, 58, 71, 68,
68, 65, 60 };

    public void pengurutantinggiTimA() {
        Arrays.sort(dataTinggiTimA);
    }

    public void pengurutanberatbadantimA() {
        Arrays.sort(dataBeratBadanTimA);
    }

    public void pengurutantinggiTimB() {
        Arrays.sort(dataTinggiTimB);
    }

    public void pengurutanberatbadantimB() {
        Arrays.sort(dataBeratBadanTimB);
    }
}

```

```

    public void
pengurutantinggiabadandanberatbadangabungantimAdantimBAscending(
) {
    int[] dataTinggiGabungan = new
int[dataTinggiTimA.length + dataTinggiTimB.length];
    System.arraycopy(dataTinggiTimA, 0, dataTinggiGabungan,
0, dataTinggiTimA.length);
    System.arraycopy(dataTinggiTimB, 0, dataTinggiGabungan,
dataTinggiTimA.length, dataTinggiTimB.length);

    int[] dataBeratBadanGabungan = new
int[dataBeratBadanTimA.length + dataBeratBadanTimB.length];
    System.arraycopy(dataBeratBadanTimA, 0,
dataBeratBadanGabungan, 0, dataBeratBadanTimA.length);
    System.arraycopy(dataBeratBadanTimB, 0,
dataBeratBadanGabungan, dataBeratBadanTimA.length,
dataBeratBadanTimB.length);

    Integer[] combinedData = new
Integer[dataTinggiGabungan.length];
    for (int i = 0; i < dataTinggiGabungan.length; i++) {
        combinedData[i] = i;
    }

    // Menambahkan pembandingan untuk pengurutan
    Arrays.sort(combinedData,
Comparator.comparingInt((Integer i) ->
dataTinggiGabungan[i]).thenComparingInt(i ->
dataBeratBadanGabungan[i]));

    System.out.println("Urutan Tinggi Badan dan Berat Badan
Gabungan Tim A dan Tim B (Ascending) : ");

    System.out.println("Urutan Tinggi dan Berat Kedua Tim
(Ascending):");
    for (int i = 0; i < combinedData.length; i++) {
        int index = combinedData[i];
        System.out.println("Tinggi: " +
dataTinggiGabungan[index] + " | Berat: " +
dataBeratBadanGabungan[index]);
    }
}

```

```

    public void
    pengurutanTinggiBadanDanBeratBadanGabunganTimAdanTimBDescending
    () {
        int[] dataTinggiGabungan = new int[dataTinggiTimA.length +
        dataTinggiTimB.length];
        System.arraycopy(dataTinggiTimA, 0, dataTinggiGabungan, 0,
        dataTinggiTimA.length);
        System.arraycopy(dataTinggiTimB, 0, dataTinggiGabungan,
        dataTinggiTimA.length, dataTinggiTimB.length);

        int[] dataBeratBadanGabungan = new
        int[dataBeratBadanTimA.length + dataBeratBadanTimB.length];
        System.arraycopy(dataBeratBadanTimA, 0,
        dataBeratBadanGabungan, 0, dataBeratBadanTimA.length);
        System.arraycopy(dataBeratBadanTimB, 0,
        dataBeratBadanGabungan, dataBeratBadanTimA.length,
        dataBeratBadanTimB.length);
        Integer[] combinedData = new
        Integer[dataTinggiGabungan.length];
        for (int i = 0; i < dataTinggiGabungan.length; i++) {
            combinedData[i] = i;
        }

        Arrays.sort(combinedData, Comparator.comparingInt((Integer
        i) -> dataTinggiGabungan[i]).thenComparingInt(i ->
        dataBeratBadanGabungan[i]));
        Arrays.sort(combinedData, Collections.reverseOrder());
        System.out.println("Urutan Tinggi Badan dan Berat Badan
        Gabungan Tim A dan Tim B (Descending) : ");
        for (int i = 0; i < combinedData.length; i++) {
            int p = combinedData[i];
            System.out.println("Tinggi : " + dataTinggiGabungan[p]
            + " Berat Badan : " + dataBeratBadanGabungan[p]);
        }
    }

    public int tinggiTerkecilTimA() {
        return dataTinggiTimA[0];
    }

    public int tinngiTerbesarTimA() {
        return dataTinggiTimA[dataTinggiTimA.length - 1];
    }

```



```

    public int beratBadanTerkecilTimA() {
        return dataBeratBadanTimA[0];
    }

    public int beratBadanTerbesarTimA() {
        return dataBeratBadanTimA[dataBeratBadanTimA.length -
1];
    }

    public int tinggiTerkecilTimB() {
        return dataTinggiTimB[0];
    }

    public int tinngiTerbesarTimB() {
        return dataTinggiTimB[dataTinggiTimB.length - 1];
    }

    public int beratBadanTerkecilTimB() {
        return dataBeratBadanTimB[0];
    }

    public int beratBadanTerbesarTimB() {
        return dataBeratBadanTimB[dataBeratBadanTimB.length -
1];
    }

    public int[] copyAnggotaTimB () {
        return Arrays.copyOf(dataTinggiTimB,
dataTinggiTimB.length);
    }
}

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

        datasetFutsal data = new datasetFutsal();
        // Berdasarkan Tinggi Badan
        //TIM A Ascending
        data.pengurutantinggiTimA();
        System.out.println("Tim A: ");
        System.out.println("Tinggi Badan Tim A (Ascending): "
+ Arrays.toString(data.dataTinggiTimA));
    }
}

```

```
//TIM A Descending
Integer[] descendingTinggiTimA =
Arrays.stream(data.dataTinggiTimA).boxed().sorted(Comparator.re
verseOrder())
.toArray(Integer[]::new);
System.out.println("Tinggi Badan Tim A (Descending): "
+ Arrays.toString(descendingTinggiTimA) + "\n");

//TIM B Ascending
data.pengurutantinggiTimB();
System.out.println("Tim B: ");
System.out.println("Tinggi Badan Tim B (Ascending): " +
Arrays.toString(data.dataTinggiTimB));

//TIM B Descending
Integer[] descendingTinggiTimB =
Arrays.stream(data.dataTinggiTimB).boxed().sorted(Comparator.re
verseOrder())
.toArray(Integer[]::new);
System.out.println("Tinggi Badan Tim B (Descending): "
+ Arrays.toString(descendingTinggiTimB) + "\n");

// Berdasarkan Berat Badan
//TIM A Ascending
data.pengurutanberatbadantimA();
System.out.println("Tim A: ");
System.out.println("Berat Badan Tim A (Ascending): " +
Arrays.toString(data.dataBeratBadanTimA));

//TIM A Descending
Integer[] descendingBeratBadanTimA =
Arrays.stream(data.dataBeratBadanTimA).boxed().sorted(Comparato
r.reverseOrder())
.toArray(Integer[]::new);
System.out.println("Berat Badan Tim A (Descending): " +
Arrays.toString(descendingBeratBadanTimA) + "\n");

//TIM B Ascending
data.pengurutanberatbadantimB();
System.out.println("Tim B: ");
System.out.println("Berat Badan Tim B (Ascending): " +
Arrays.toString(data.dataBeratBadanTimB));

//TIM B Descending
```

```

        Integer[] descendingBeratBadanTimB =
Arrays.stream(data.dataBeratBadanTimB).boxed().sorted(Comparato
r.reverseOrder())
        .toArray(Integer[]::new);
        System.out.println("Berat Badan Tim B (Descending): " +
Arrays.toString(descendingBeratBadanTimB) + "\n");

data.pengurutantinggibadandanberatbadangabungantimAdantimBAscen
ding();
        System.out.println();

data.pengurutantinggibadandanberatbadangabungantimAdantimBDecen
ding();
        System.out.println();

        // Nilai maksimum dan minimum Tinggi Badan dan Berat
Badan untuk pemain
        // dari masing-masing tim.
        // Tim A
                int tinggiTerkecilTimA =
data.tinggiTerkecilTimA();
                int tinggiTerbesarTimA =
data.tinggiTerbesarTimA();
                int beratBadanTerkecilTimA =
data.beratBadanTerkecilTimA();
                int beratBadanTerbesarTimA =
data.beratBadanTerbesarTimA();
                System.out.println("Tim A:");
                System.out.println("Tinggi Terkecil Tim A: " +
tinggiTerkecilTimA);
                System.out.println("Tinggi Terbesar Tim A: " +
tinggiTerbesarTimA);
                System.out.println("Berat Terkecil Tim A: " +
beratBadanTerkecilTimA);
                System.out.println("Berat Terbesar Tim A: " +
beratBadanTerbesarTimA + "\n");

        // Tim B
                int tinggiTerkecilTimB =
data.tinggiTerkecilTimB();
                int tinggiTerbesarTimB =
data.tinggiTerbesarTimB();

```

```

        int beratBadanTerkecilTimB =
data.beratBadanTerkecilTimB();
        int beratBadanTerbesarTimB =
data.beratBadanTerbesarTimB();
        System.out.println("Tim B:");
        System.out.println("Tinggi Terkecil Tim B: " +
tinggiTerkecilTimB);
        System.out.println("Tinggi Terbesar Tim B: " +
tinggiTerbesarTimB);
        System.out.println("Berat Terkecil Tim B: " +
beratBadanTerkecilTimB);
        System.out.println("Berat Terbesar Tim B: " +
beratBadanTerbesarTimB + "\n");

        // d. Copy seluruh anggota Tim B ke Tim C yang
baru dibentuk
        System.out.println("Salinan      seluruh anggota
Tim B ke Tim C:");
        int[] dataTinggiTimC =
data.copyAnggotaTimB();
        int[] dataBeratBadanTimC =
Arrays.copyOf(data.dataBeratBadanTimB,
data.dataBeratBadanTimB.length);
        System.out.println("Data Tinggi Tim C: " +
Arrays.toString(dataTinggiTimC));
        System.out.println("Data Berat Tim C: " +
Arrays.toString(dataBeratBadanTimC));
    }
}

```

Output

Tim A:
 Tinggi Badan Tim A (Ascending): [165, 165, 166, 168, 168, 169, 170, 170, 171, 172]
 Tinggi Badan Tim A (Descending): [172, 171, 170, 170, 169, 168, 168, 166, 165, 165]

Tim B:
 Tinggi Badan Tim B (Ascending): [165, 166, 167, 168, 168, 169, 170, 171, 172, 175]
 Tinggi Badan Tim B (Descending): [175, 172, 171, 170, 169, 168, 168, 167, 166, 165]

Tim A:
 Berat Badan Tim A (Ascending): [50, 55, 56, 56, 56, 60, 60, 66, 70, 72]
 Berat Badan Tim A (Descending): [72, 70, 66, 60, 60, 56, 56, 56, 55, 50]

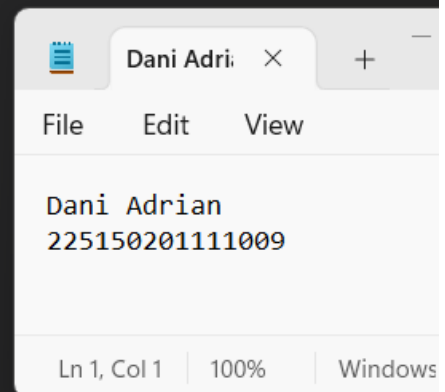
Tim B:
 Berat Badan Tim B (Ascending): [58, 58, 59, 60, 60, 65, 66, 68, 68, 71]
 Berat Badan Tim B (Descending): [71, 68, 68, 66, 65, 60, 60, 59, 58, 58]

Urutan Tinggi Badan dan Berat Badan Gabungan Tim A dan Tim B (Ascending) :
 Urutan Tinggi dan Berat Kedua Tim (Ascending):

Tinggi: 165	Berat: 50
Tinggi: 165	Berat: 55
Tinggi: 165	Berat: 58
Tinggi: 166	Berat: 56
Tinggi: 166	Berat: 58
Tinggi: 167	Berat: 59
Tinggi: 168	Berat: 56
Tinggi: 168	Berat: 56
Tinggi: 168	Berat: 60
Tinggi: 168	Berat: 60
Tinggi: 169	Berat: 60
Tinggi: 169	Berat: 65
Tinggi: 170	Berat: 60
Tinggi: 170	Berat: 66
Tinggi: 170	Berat: 66
Tinggi: 172	Berat: 72
Tinggi: 175	Berat: 71

Urutan Tinggi Badan dan Berat Badan Gabungan Tim A dan Tim B (Descending) :

Tinggi : 175	Berat Badan : 71
Tinggi : 172	Berat Badan : 68
Tinggi : 171	Berat Badan : 68
Tinggi : 170	Berat Badan : 66
Tinggi : 169	Berat Badan : 65
Tinggi : 168	Berat Badan : 60
Tinggi : 168	Berat Badan : 60
Tinggi : 167	Berat Badan : 59





```
Tinggi: 172 | Berat: 72
Tinggi: 175 | Berat: 71
```

Urutan Tinggi Badan dan Berat Badan Gabungan Tim A dan Tim B (Descending) :

```
Tinggi : 175 Berat Badan : 71
Tinggi : 172 Berat Badan : 68
Tinggi : 171 Berat Badan : 68
Tinggi : 170 Berat Badan : 66
Tinggi : 169 Berat Badan : 65
Tinggi : 168 Berat Badan : 60
Tinggi : 168 Berat Badan : 60
Tinggi : 167 Berat Badan : 59
Tinggi : 166 Berat Badan : 58
Tinggi : 165 Berat Badan : 58
Tinggi : 172 Berat Badan : 72
Tinggi : 171 Berat Badan : 70
Tinggi : 170 Berat Badan : 66
Tinggi : 170 Berat Badan : 60
Tinggi : 169 Berat Badan : 60
Tinggi : 168 Berat Badan : 56
Tinggi : 168 Berat Badan : 56
Tinggi : 166 Berat Badan : 56
Tinggi : 165 Berat Badan : 55
Tinggi : 165 Berat Badan : 50
```

Tim A:

```
Tinggi Terkecil Tim A: 165
Tinggi Terbesar Tim A: 172
Berat Terkecil Tim A: 50
Berat Terbesar Tim A: 72
```

Tim B:

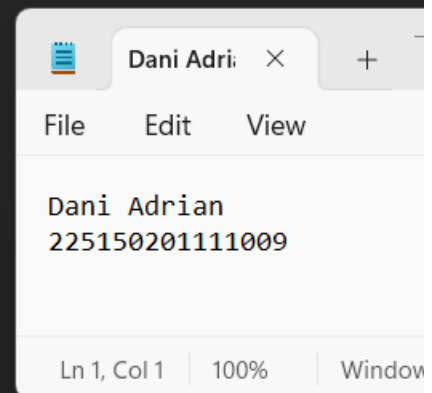
```
Tinggi Terkecil Tim B: 165
Tinggi Terbesar Tim B: 175
Berat Terkecil Tim B: 58
Berat Terbesar Tim B: 71
```

Salinan seluruh anggota Tim B ke Tim C:

```
Data Tinggi Tim C: [165, 166, 167, 168, 168, 169, 170, 171, 172, 175]
```

```
Data Berat Tim C: [58, 58, 59, 60, 60, 65, 66, 68, 68, 71]
```

```
PS C:\Users\USER\JAVA VS CODE\OOP\praktikumkeselabel>
```



Penjelasan

- Syntax `Arrays.sort(dataTinggiTimA)` dan `Arrays.sort(dataTinggiTimB)` digunakan untuk mengurutkan array `dataTinggiTimA` dan `dataTinggiTimB` secara naik (ascending) menggunakan metode `sort` dari kelas `Arrays`.
- `Integer[] descendingTinggiTimA = Arrays.stream(q.dataTinggiTimA).boxed().sorted(Comparator.reverseOrder()).toArray(In`

tinggiTimA::new) dan Integer[] descendingTinggiTimB = Arrays.stream(q.dataTinggiTimB).boxed().sorted(Comparator.reverseOrder()).toArray(Integer[]::new) digunakan untuk mengurutkan array dataTinggiTimA dan dataTinggiTimB secara turun (descending). Pertama, array primitif diubah menjadi stream menggunakan Arrays.stream(). Kemudian, menggunakan metode boxed() untuk mengubah setiap elemen menjadi tipe Integer agar dapat diurutkan. Selanjutnya, menggunakan sorted(Comparator.reverseOrder()) untuk mengurutkan secara turun. Terakhir, toArray(Integer[]::new) digunakan untuk mengonversi stream kembali menjadi array Integer.

- Pada bagian method main, objek DataSetFutsal dengan nama q dibuat. Kemudian, method urutkanTinggiTimA() dan urutkanTinggiTimB() dipanggil untuk mengurutkan tinggi badan dari Tim A dan Tim B secara naik.
- Selanjutnya, hasil pengurutan ditampilkan menggunakan System.out.println(). Array dataTinggiTimA dan dataTinggiTimB dicetak secara langsung menggunakan Arrays.toString(), sedangkan array yang sudah diurutkan secara turun (descendingTinggiTimA dan descendingTinggiTimB) juga dicetak menggunakan Arrays.toString().
- Method urutkanBeratBadanTimA() digunakan untuk mengurutkan berat badan (BeratBadan) dari Tim A secara naik menggunakan method Arrays.sort().
- Method urutkanBeratBadanTimB() digunakan untuk mengurutkan berat badan (BeratBadan) dari Tim B secara naik menggunakan method Arrays.sort().
- Setelah itu, mencetak hasil urutan berat badan Tim A dalam urutan naik dengan menggunakan Arrays.toString() untuk mengonversi array menjadi string.
- Selanjutnya, membuat array descendingBeratBadanTimA dengan menggunakan Arrays.stream() untuk membuat stream dari array dataBeratBadanTimA, kemudian menggunakan boxed() untuk mengubahnya menjadi stream of boxed Integer, lalu menggunakan sorted() dengan Comparator.reverseOrder() untuk mengurutkan dalam urutan turun, dan akhirnya menggunakan toArray() untuk mengonversi stream kembali menjadi array.
- Kemudian, mencetak hasil urutan berat badan Tim A dalam urutan turun menggunakan Arrays.toString().
- Proses serupa dilakukan untuk Tim B dengan menggunakan method urutkanBBTimB() dan mencetak hasil urutan berat badan Tim B dalam urutan naik dan turun.
- Method tinggiTerkecilTimA(), tinggiTerbesarTimA(), tinggiTerkecilTimB(), tinggiTerbesarTimB(), beratbadanTerkecilTimA(), beratbadanTerbesarTimA(), beratbadanTerkecilTimB(), dan beratbadanTerbesarTimB() digunakan untuk

mengembalikan nilai tinggi badan terkecil, tinggi badan terbesar, berat badan terkecil, dan berat badan terbesar dari masing-masing tim.

- Pada method main, nilai tinggi dan berat badan terkecil serta terbes

Pertanyaan

2. Buatlah implementasi *binary search* dalam program java berdasarkan kondisi berikut :
 - a. Implementasikan *ArrayList* untuk menyimpan data tim A dan tim B dalam bentuk *ArrayList* terpisah
 - b. Dari data tim B, dicari jumlah pemain yang mempunyai tinggi badan 168 cm dan 160 cm
 - c. Dari data tim A, dicari jumlah pemain yang mempunyai berat badan 56 kg dan 53 kg.
 - d. Ingin diketahui apakah pemain di tim A ada yang mempunyai tinggi badan atau berat badan yang sama dengan pemain di tim B ?

Source Code

```
BinarySearchFutsal.java
1.  import java.util.*;
2.  class DataSetFutsal2 {
3.      ArrayList<Integer> DataTinggiBadanTimA = new
        ArrayList<>();
4.      ArrayList<Integer> DataBeratBadanTimA = new
        ArrayList<>();
5.      ArrayList<Integer> dataTinggiTimB = new
        ArrayList<>();
6.      ArrayList<Integer> dataBBTimB = new ArrayList<>();
7.      public DataSetFutsal2() {
8.          // Data Tim A
9.          DataTinggiBadanTimA.add(168);
10.         DataTinggiBadanTimA.add(170);
11.         DataTinggiBadanTimA.add(165);
12.         DataTinggiBadanTimA.add(168);
13.         DataTinggiBadanTimA.add(172);
14.         DataTinggiBadanTimA.add(170);
15.         DataTinggiBadanTimA.add(169);
16.         DataTinggiBadanTimA.add(165);
17.         DataTinggiBadanTimA.add(171);
18.         DataTinggiBadanTimA.add(166);
19.         DataBeratBadanTimA.add(50);
20.         DataBeratBadanTimA.add(60);
21.         DataBeratBadanTimA.add(56);
22.         DataBeratBadanTimA.add(55);
```



```
23.      DataBeratBadanTimA.add(60);
24.      DataBeratBadanTimA.add(70);
25.      DataBeratBadanTimA.add(66);
26.      DataBeratBadanTimA.add(56);
27.      DataBeratBadanTimA.add(72);
28.      DataBeratBadanTimA.add(56);
29.      // Data Tim B
30.      dataTinggiTimB.add(170);
31.      dataTinggiTimB.add(167);
32.      dataTinggiTimB.add(165);
33.      dataTinggiTimB.add(166);
34.      dataTinggiTimB.add(168);
35.      dataTinggiTimB.add(175);
36.      dataTinggiTimB.add(172);
37.      dataTinggiTimB.add(171);
38.      dataTinggiTimB.add(168);
39.      dataTinggiTimB.add(169);
40.      dataBBTimB.add(66);
41.      dataBBTimB.add(60);
42.      dataBBTimB.add(59);
43.      dataBBTimB.add(58);
44.      dataBBTimB.add(58);
45.      dataBBTimB.add(71);
46.      dataBBTimB.add(68);
47.      dataBBTimB.add(68);
48.      dataBBTimB.add(65);
49.      dataBBTimB.add(60);
50.  }
51.  public int binarySearch(ArrayList<Integer> arr, int
target) {
52.      int left = 0;
53.      int right = arr.size() - 1;
54.      do {
55.          int mid = left + (right - left)
56.              / 2;
57.          int midValue = arr.get(mid);
58.          if (midValue == target) {
59.              return mid;
60.          } else if (midValue < target) {
61.              left = mid + 1;
62.          } else {
63.              right = mid - 1;
64.          }
65.      } while (left <= right);
```

```
66.         return -1;
67.     }
68.
69.     public int
countPlayersWithHeight(ArrayList<Integer> arr, int
height) {
70.         int count = 0;
71.         int i = 0;
72.         do {
73.             if (arr.get(i) == height) {
74.                 count++;
75.             }
76.             i++;
77.         } while (i < arr.size());
78.         return count;
79.     }
80.
81.     public int
countPlayersWithWeight(ArrayList<Integer> arr, int
weight) {
82.         int count = 0;
83.         int i = 0;
84.         do {
85.             if (arr.get(i) == weight) {
86.                 count++;
87.             }
88.             i++;
89.         } while (i < arr.size());
90.         return count;
91.     }
92.
93.     public boolean isSameHeightOrWeight() {
94.         int i = 0;
95.         do {
96.             int heightA = DataTinggiBadanTimA.get(i);
97.             int weightA = DataBeratBadanTimA.get(i);
98.             if (binarySearch(dataTinggiTimB, heightA)
!= -1 || binarySearch(dataBBTimB, weightA) != -1) {
99.                 return true;
100.            }
101.            i++;
102.        } while (i < DataTinggiBadanTimA.size());
103.        return false;
104.    }
```

```
105. }
106. public class BinarySearchFutsal {
107.     public static void main(String[] args) {
108.         DataSetFutsal2 q = new DataSetFutsal2();
109.         // a) Data tim A dan tim B dalam ArrayList
        terpisah
110.         ArrayList<Integer> dataTimA = new
        ArrayList<>(q.DataTinggiBadanTimA);
111.         ArrayList<Integer> data_TimA = new
        ArrayList<>(q.DataBeratBadanTimA);
112.         ArrayList<Integer> dataTimB = new
        ArrayList<>(q.dataTinggiTimB);
113.         ArrayList<Integer> data_TimB = new
        ArrayList<>(q.dataBBTimB);
114.         System.out.println("Data tinggi tim a:");
115.         System.out.println(dataTimA);
116.         System.out.println("Data berat badan tim a:");
117.         System.out.println(data_TimA);
118.         System.out.println();
119.         System.out.println("Data tinggi tim b:");
120.         System.out.println(dataTimB);
121.         System.out.println("Data berat badan tim b:");
122.         System.out.println(data_TimB);
123.         System.out.println();
124.         // b) Jumlah pemain di tim B dengan tinggi 168
        cm dan 160 cm
125.         int count168 =
        q.countPlayersWithHeight(dataTimB, 168);
126.         int count160 =
        q.countPlayersWithHeight(dataTimB, 160);
127.         System.out.println("Jumlah pemain di tim B
        dengan tinggi 168 cm: " + count168);
128.         System.out.println("Jumlah pemain di tim B
        dengan tinggi 160 cm: " + count160 + "\n");
129.         // c) Jumlah pemain di tim A dengan berat 56 kg
        dan 53 kg
130.         int count56 =
        q.countPlayersWithWeight(q.DataBeratBadanTimA, 56);
131.         int count53 =
        q.countPlayersWithWeight(q.DataBeratBadanTimA, 53);
132.         System.out.println("Jumlah pemain di tim A
        dengan berat 56 kg: " + count56);
133.         System.out.println("Jumlah pemain di tim A
        dengan berat 53 kg: " + count53 + "\n");
```

```

134.
135.         // d) Apakah ada pemain di tim A dengan tinggi
        atau berat yang sama dengan
136.         // pemain di tim B
137.         boolean sameHeightOrWeight =
        q.isSameHeightOrWeight();
138.         if (sameHeightOrWeight == true) {
139.             System.out.println(
140.                 "Apakah ada pemain di tim A dengan
        tinggi atau berat yang sama dengan pemain di tim B: Ada
        ");
141.         } else {
142.             System.out.println(
143.                 "Apakah ada pemain di tim A dengan
        tinggi atau berat yang sama dengan pemain di tim B:
        Tidak ada ");
144.         }
145.     }
146. }

```

Output

```

Data tinggi tim a:
[168, 170, 165, 168, 172, 170, 169, 165, 171, 166]
Data berat badan tim a:
[50, 60, 56, 55, 60, 70, 66, 56, 72, 56]

```

```

Data tinggi tim b:
[170, 167, 165, 166, 168, 175, 172, 171, 168, 169]
Data berat badan tim b:
[66, 60, 59, 58, 58, 71, 68, 68, 65, 60]

```

```

Jumlah pemain di tim B dengan tinggi 168 cm: 2
Jumlah pemain di tim B dengan tinggi 160 cm: 0

```

```

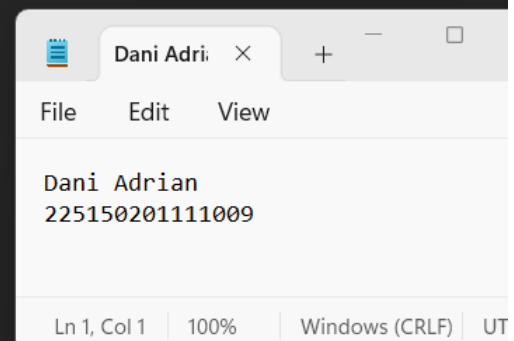
Jumlah pemain di tim A dengan berat 56 kg: 3
Jumlah pemain di tim A dengan berat 53 kg: 0

```

```

Apakah ada pemain di tim A dengan tinggi atau berat yang sama dengan pemain di tim B: Ada

```



Penjelasan

- Kelas `Data_Set_Futsal` digunakan untuk menyimpan data tinggi badan dan berat badan pemain dalam `ArrayList` terpisah untuk tim A dan tim B.
- Dalam konstruktor `Data_Set_Futsal()`, data tinggi badan dan berat badan pemain ditambahkan ke `ArrayList` `dataTinggiTimA`, `dataBBTimA`, `dataTinggiTimB`, dan `dataBBTimB` sesuai dengan data yang diberikan.
- Dalam kelas `BinarySearchFutsal`, objek `q` dari kelas `Data_Set_Futsal` dibuat.

- Bagian yang dijelaskan adalah bagian yang mencetak data tinggi badan dan berat badan untuk tim A dan tim B secara terpisah menggunakan ArrayList yang telah dibuat sebelumnya.
- Data tinggi badan dan berat badan tim A dicetak terlebih dahulu, diikuti oleh data tinggi badan dan berat badan tim B. Setiap ArrayList dicetak menggunakan metode `println()` dari kelas `System.out`.

Method `binarySearch`

- Mengimplementasikan algoritma binary search. Method ini menerima ArrayList `arr` dan nilai target yang ingin dicari. Dalam method ini, digunakan `do-while` loop untuk melakukan pencarian secara berulang dengan membagi array menjadi setengah pada setiap iterasi. Jika ditemukan nilai yang sesuai dengan target, method mengembalikan indeks dari nilai tersebut. Jika tidak ditemukan, method mengembalikan -1.
- Method `countPlayersWithHeight` digunakan untuk menghitung jumlah pemain dengan tinggi tertentu dalam ArrayList `arr`. Jumlah pemain dihitung dengan menggunakan `do-while` loop. Setiap kali elemen dengan tinggi yang sesuai ditemukan, variabel `count` akan bertambah. Method ini mengembalikan jumlah pemain yang ditemukan.
- Pada method `main` dalam kelas `BinarySearchFutsal`, objek `Data_Set_Futsal` dengan nama `q` dibuat. Kemudian, menggunakan objek `q`, dilakukan pemanggilan method `countPlayersWithHeight` untuk menghitung jumlah pemain di Tim B dengan tinggi 168 cm dan 160 cm. Hasilnya kemudian dicetak ke layar.

Method `countPlayersWithWeight`:

- Menerima dua parameter, yaitu `arr` (ArrayList) dan `weight` (berat yang ingin dicari).
- Membuat variabel `count` dengan nilai awal 0 untuk menyimpan jumlah pemain dengan berat yang sesuai.
- Menggunakan `do-while` loop untuk mengiterasi melalui elemen-elemen ArrayList `arr`.
- Di dalam loop, setiap elemen diuji apakah sama dengan `weight`. Jika benar, `count` akan bertambah.
- Variabel `i` ditingkatkan setelah setiap iterasi.
- Loop akan berlanjut selama `i` kurang dari ukuran ArrayList `arr`.
- Mengembalikan nilai `count` setelah loop selesai.

Kelas `BinarySearchFutsal`:

- Pada method `main`, membuat objek `q` dari kelas `Data_Set_Futsal`.
- Menggunakan objek `q` untuk memanggil metode `countPlayersWithWeight` dengan memberikan `q.dataBBTimA` dan berat 56 sebagai argumen.
- Hasilnya dalam variabel `count56`.

- Juga, memanggil metode yang sama dengan berat 53 dan menyimpan hasilnya dalam variabel `count53`.
- Menggunakan `System.out.println` untuk menampilkan jumlah pemain dengan berat 56 kg dan 53 kg di tim A.

Method `isSameHeightOrWeight()`:

- Menggunakan do-while loop untuk mengiterasi melalui elemen-elemen dalam `ArrayList dataTinggiTimA`.
- Pada setiap iterasi, nilai tinggi dan berat pemain dari Tim A pada indeks saat ini diambil.
- Dilakukan pencarian menggunakan binary search dengan menggunakan metode `binarySearch()` untuk mencari apakah ada pemain di Tim B yang memiliki tinggi atau berat yang sama.
- Jika ditemukan, fungsi akan mengembalikan `true`, menandakan bahwa ada pemain di Tim A dengan tinggi atau berat yang sama dengan pemain di Tim B.
- Jika tidak ditemukan setelah iterasi selesai, fungsi akan mengembalikan `false`, menandakan bahwa tidak ada pemain di Tim A dengan tinggi atau berat yang sama dengan pemain di Tim B.

Kelas `BinarySearchFutsal`:

- Membuat objek `Data_Set_Futsal` dengan nama `q`.
- Memanggil metode `isSameHeightOrWeight()` untuk mencari apakah ada pemain di Tim A dengan tinggi atau berat yang sama dengan pemain di Tim B.
- Menampilkan output sesuai dengan hasil pencarian.