MODUL

“HashSet, LinkedHashSet dan TreeSet”

PEMOGRAMAN BERORIENTASI OBJEK

Dosen Pengampu: Jefril Rahmadoni, M.Kom

Disusun Oleh :

Citra Aulia	2111521022	Suci Rahmadhani	2111522014
Fikran Shadiq El Yafit	2111521024	Alvino Albas	2111522016
Gilang Kharisma	2111522002	Putra Ilham	2111522018
Ariq Abdurrahman Hakim	2111522006	Syahnia Putri Hendry	2111522020
M. Satria Gemilang Lubis	2111522008	Briana Firsta	2111522024
Athifa Rifda Andra	2111522010	M. Yamin	
Annisa Hasifah Cantika	2111522012		

Kelas 02

JURUSAN SISTEM INFORMASI

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2022

A. Pokok Pembahasan

1. Konsep dari HashSet, LinkedHashSet, dan TreeSet.
2. Metode dan konstruktor dari HashSet, LinkedHashSet, dan TreeSet.
3. Persamaan dan perbedaan dari HashSet, LinkedHashSet, dan TreeSet.

B. Tujuan Belajar

Dengan adanya pembuatan modul ini penulis dan pembaca diharapkan dapat:

1. Memahami konsep dari HashSet, LinkedHashSet, dan TreeSet.
2. Mampu untuk menerapkan metode serta konstruktor dari HashSet, LinkedHashSet, dan TreeSet pada pemograman.
3. Memahami persamaan dan perbedaan dari HashSet, LinkedHashSet, dan TreeSet.

C. Dasar Teori

1. HashSet
   a. Deskripsi

Kelas HashSet mengimplementasikan Interface Set, dengan menggunakan Hash Table yang merupakan instance HashMap. HashSet tidak mengeluarkan elemen yang ada di dalamnya dengan iteration order, namun, HashSet merupakan metode tercepat untuk menemukan elemen dalam array, sebab, digunakan modulus dan nilai ASCII yang dapat menemukan langsung elemen yang dicari.

Singkatnya, HashSet adalah sebuah koleksi elemen yang mana setiap elemen dalam HashSet bernilai unik dan elemen yang sama akan selalu memiliki nilai yang sama pula, hal ini yang juga memudahkan pencarian elemen dalam array menggunakan HashSet.

Struktur yang mendasari HashSet ialah Hashtable sebagai tempat penyimpanan setiap elemen pada HashSet. Agar kinerja waktu yang digunakan konstan, iterasi pada HashSet memerlukan waku yang sebanding dengan jumlah elemen ditambah kapasitas dari instance HashMap. Maka, sangat penting untuk tidak menetapkan kapasitas awal terlalu tinggi atau Load Factor teralu rendah jika dibutuhkan kinerja iterasi yang konstan.

Kapasitas awal adalah jumlah awal bucket saat HashTable dibuat. Jumlah bucket akan otomatis bertambah jika jumlah awal sudah penuh. Sedangkan Load Factor adalah ukuran awal HashSet dapat penuh sebelum kapasitasnya ditingkatkan

secara otomatis. Ketika jumlah entri dalam HashTable melebihi Load Factor dan kapasitas awal, maka, dibangun kembali struktur data internal yang menghasilkan jumlah table hash 2 kali lipat dari jumlah awal.

b. Algoritma Hashing

Dilakukan perhitungan pada nilai ASCII dari elemen dengan ukuran kelas untuk mentransformasikannya ke sebuah alamat yang langsung mengarahkan perhitungan ke posisi elemen berada. Perhitungannya ialah:

Index number = sum ASCII codes Mod Size of Array

Contoh:

Misal, ada sebuah array sebagai berikut:

Bea	Tim	Leo	Sam	Mia	Zoe	Jan	Lou	Max	Ada	Ted
0	1	2	3	4	5	6	7	8	9	10

Akan dicari elemen “Ada” pada array diatas, algoritma Hashing akan melakukan penjumlahan nilai ASCII dari “Ada” lalu dioperasikan dengan modulus 11 yang didapat dari size array tersebut.

Penjumlahannya adalah sebagai berikut:

- Nilai ASCII “Ada”

  A = 65

  d = 100

  a = 97

  Ada = (65 + 100 + 97) = 262

- Ukuran array: 11

Maka, 262 Mod 11 = 9. Disini, sudah ditemukan bahwa elemen “Ada” berada pada urutan ke-9 pada array.

c. Metode HashSet

Ada beberapa metode yang dapat digunakan pada HashSet:

1) contains()

Metode Contains digunakan untuk memeriksa apakah sebuah elemen ada pada sebuah HashSet.

Contoh: cars.contains("Mazda");

Maka, akan dicari apakah pada kelas “cars” ada elemen “Mazda”. Akan dikeluarkan output “true” jika ada, dan “false” jika tidak ada.

2) remove()

Metode remove() digunakan untuk menghapus elemen yang ada pada class.

Contoh: cars.remove("Volvo");

Perintah ini akan menghapus elemen “Volvo” pada kelas “cars”.

3) clear()

Metode clear digunakan untuk menghapus semua elemen pada class.

Contoh: cars.clear();

Dengan perintah ini, maka, setiap elemen pada kelas “cars” akan dihapus dan kelas akan kosong.

4) size()

Metode ini akan menghitung jumlah elemen yang ada pada HashSet dan untuk elemen yang diduplikasi akan tetap terhitung 1 sebab ia menempati 1 tempat pada array.

Contoh: cars.size();

Metode ini akan menampilkan jumah elemen pada array.

d. Konstruktor HashSet

Dalam pembuatan HashSet, dibutuhkan objek dari kelas HashSet. Kelas HashSet terdiri dari beberapa konstruktor, diantaranya:

1) HashSet

Fungsi dari konstruktor ini digunakan untuk membuat objek HashSet kosong dengan kapasitas awal default 16 dan faktor beda default 0,75. Untuk membuat HashSet kosong dengan nama hs, maka code-nya ialah:

HashSet<E> hs = HashSet baru<E>();

2) HashSet(int InitialCapacity)

Konstruktor ini digunakan untuk membangun pembuatan objek HashSet kosong dengan kapasitas awal ditentukan saat pembuatan objek dan Load Factor tetap 0,75. Code-nya ialah:

HashSet<E> hs = new HashSet<E>(int initialCapacity);

3) HashSet(int initialCapacity, float loadFactor)

Konstruktor ini berfungsi untuk membangun objek HashSet kosong dimana initialCapacity dan Load Factor ditentukan pada saat pembuatan objek.

HashSet<E> hs = new HashSet<E>(int kapasitas awal, float faktor muat);

4) HashSet(Collection)

Fungsi konstruktor ini ialah untuk mengkonversi koleksi elemen isi dari sebuah objek ke objek HashSet. Jika ingin membuat HashSet dengan nama hs, maka dapat code-nya ialah:

HashSet<E> hs = HashSet baru<E>(Koleksi C);

2. LinkedHashSet

a. Deskripsi

Linked Hash Set adalah versi terurut dari Hash Set yang mempertahankan daftar tertaut ganda di semua elemen. Ketika urutan iterasi diperlukan untuk dipertahankan, kelas ini digunakan. Saat melakukan iterasi melalui HashSet, urutannya tidak dapat diprediksi, sedangkan LinkedHashSet memungkinkan kita melakukan iterasi melalui elemen sesuai urutan penyisipannya. Saat beredar melalui LinkedHashSet menggunakan iterator, elemen akan dikembalikan sesuai urutan penyisipannya.

Perbedaannya dari HashSet ialah:

- Urutan penyisipan elemen dipertahankan selama pembuatan LinkedHashSet.
- Struktur data yang mendasarinya adalah gabungan dari Tabel Hash (dalam HashSet) & Daftar Tertaut.
- Duplikat tidak diperbolehkan di LinkedHashSet.

b. Metode LinkedHashSet

Semua metode sama dengan metode seperti di kelas HashSet. Ini menyiratkan bahwa operasi apa pun yang dapat kita lakukan dengan koleksi Hash Set juga dimungkinkan dengan kelas LinkedHashSet. Oleh karena itu, manipulasi koleksi LinkedHashSet tidak diilustrasikan secara eksplisit

c. Konstruktor LinkedHashSet

Konstruktor di kelas LinkedHashSet memiliki bentuk yang sama dengan konstruktor di kelas Hashset. Kelas LinkedHashSet memperluas kelas HashSet dan mengimplementasikan antarmuka Set. Kelas LinkedHashSet tidak mendefinisikan metode eksklusifnya sendiri.

3. TreeSet

a. Deskripsi

Merupakan class yang sering digunakan untuk mengekstrak elemen dari collection dalam urutan tertentu. TreeSet juga salah satu implementasi terpenting dari interface SortedSet di Java yang menggunakan Tree untuk penyimpanan. TreeSet mengimplementasikan antarmuka SortedSet sehingga nilai duplikat tidak diperbolehkan. Objek dalam TreeSet disimpan dalam urutan yang diurutkan dan menaik.

Objek bertipe TreeSet memiliki sifat di mana elemen-elemennya terurut dalam urutan menaik. Iterator dari suatu TreeSet akan mengunjungi elemen dalam set tersebut dalam urutan menaik. TreeSet tidak bisa menyimpan semua objek, karena objek yang disimpan harus bisa diurutkan.

TreeSet menyediakan implementasi dari interface Set yang menggunakan pohon untuk penyimpanan. Objek disimpan dalam disortir, urutan menaik. Akses dan pengambilan waktu yang cukup cepat, yang membuat TreeSet pilihan yang sangat baik ketika menyimpan sejumlah besar informasi diurutkan yang harus ditemukan dengan cepat. TreeSet berfungsi untuk menampilkan data yang bersifat unik dengan urutan yang teratur dari kecil ke besar atau sebaliknya.

b. Metode TreeSet

TreeSet mendefinisikan metode berikut:

1) void add (Object o)

   Menambahkan elemen tertentu untuk set ini jika tidak sudah ada.

2) boolean addAll (Collection c)

   Menambahkan semua elemen dalam koleksi tertentu untuk set ini.

3) Void clear ()

   Menghapus semua elemen dari himpunan ini.

4) Object clone ()

   Pengembalian salinan dangkal contoh TreeSet ini.

5) Comparator comparator ()

   Mengembalikan komparator digunakan untuk memesan set ini disortir, atau null jika set pohon ini menggunakan unsur-unsurnya memesan alami.

6) boolean contains (Object o)

   Mengembalikan nilai true jika set ini berisi elemen tertentu.

7) Objectfirst()

   Mengembalikan pertama (terendah) elemen saat ini dalam set diurutkan ini.

8) SortedSet headset (Object toElement)

   Pengembalian pemandangan bagian dari himpunan ini yang unsur-unsurnya secara ketat kurang dari toElement.

9) boolean isEmpty ()

   Mengembalikan nilai true jika set ini tidak mengandung unsur-unsur.

10) Iterator iterator ()

    Mengembalikan sebuah iterator atas unsur-unsur dalam himpunan ini.

11) Objectlast ()

Mengembalikan terakhir elemen (tertinggi) saat ini dalam set diurutkan ini.

12) boolean remove (Object o)

Menghapus elemen tertentu dari set ini jika hadir.

13) size int ()

Mengembalikan jumlah elemen dalam himpunan ini (cardinality).

14) SortedSet Subset (Object dariElemen, Obyek toElement)

Pengembalian pemandangan bagian dari himpunan ini yang berkisar dari elemen dariElemen, inklusif, untuk toElement, eksklusif.

15) SortedSet tailSet (Object dariElemen)

Pengembalian pemandangan bagian dari himpunan ini yang elemen-elemennya lebih besar dari atau sama dengan dariElemen.

c. Konstruktor TreeSet

1) TreeSet()

Bentuk pertama membangun pohon himpunan kosong yang akan diurutkan dalam urutan sesuai dengan urutan alamiah unsur-unsurnya.

2) TreeSet(Collection c)

Bentuk kedua membangun satu set pohon yang mengandung unsur-unsur c.

3) TreeSet(Comparator comp)

Bentuk ketiga membangun pohon himpunan kosong yang akan diurutkan sesuai dengan komparator ditentukan oleh comp.

4) TreeSet(SortedSet ss)

Bentuk keempat membangun satu set pohon yang mengandung unsur-unsur ss.

D. Persamaan HashSet, LinkedHashSet, dan TreeSet

1. Duplikat

Ketiga alat Set interface berarti mereka tidak diizinkan untuk menyimpan duplikat.

2. Keamanan thread

HashSet, TreeSet dan LinkedHashSet tidak aman untuk thread, jika menggunakannya di lingkungan multi-threading di mana setidaknya satu Thread diubah. Set perlu menyinkronkannya secara eksternal.

3. Fail-Fast Iterator

Iterator dikembalikan oleh TreeSet, LinkedHashSet dan HashSet adalah Iterator yang gagal-cepat.

Contoh : Jika Iterator dimodifikasi setelah pembuatannya dengan cara apa pun selain metode Iterator hapus (), ia akan membuang ConcurrentModificationException dengan upaya terbaik. baca lebih lanjut tentang Iterator gagal-cepat vs gagal-aman di sini

E. Perbedaan HashSet, LinkedHashSet, dan TreeSet

TreeSet, LinkedHashSet, dan HashSet di Java adalah tiga set implementasi dalam kerangka koleksi dan seperti banyak lainnya mereka juga digunakan untuk menyimpan objek. Fitur utama TreeSet adalah pengurutan, LinkedHashSet adalah urutan penyisipan dan HashSet hanyalah koleksi tujuan umum untuk menyimpan objek.

HashSet diimplementasikan menggunakan HashMap di Java sementara TreeSet diimplementasikan menggunakan TreeMap. TreeSet adalah implementasi SortedSet yang memungkinkannya untuk menjaga elemen-elemen dalam urutan diurutkan yang didefinisikan oleh antarmuka Comparable atau Comparator. Sebanding digunakan untuk menyortir urutan alami dan Pembanding untuk menyortir pesanan kustom objek, yang dapat disediakan saat membuat instance TreeSet.

	HashSet	**LinkedHashSet**	**TreeSet**
Bagaimana mereka bekerja secara internal?	HashSet menggunakan HashMap secara internal untuk	LinkedHashSet menggunakan LinkedHashMap secara internal	TreeSet menggunakan TreeMap secara internal untuk menyimpan elemen-elemennya.

	menyimpan elemen-elemennya.	untuk menyimpan elemen-elemennya.	
Urutan elemen	HashSet tidak mempertahankan urutan elemen apa pun.	LinkedHashSet mempertahankan urutan penyisipan elemen. yaitu elemen ditempatkan saat dimasukkan.	TreeSet memesan elemen sesuai dengan Comparator yang disediakan. Jika tidak ada pembanding yang disediakan, elemen akan ditempatkan dalam urutan naik alami mereka.
Performa	HashSet memberikan kinerja yang lebih baik daripada LinkedHashSet dan TreeSet.	Kinerja LinkedHashSet adalah antara HashSet dan TreeSet. Performanya hampir mirip dengan HashSet. Tetapi sedikit di sisi yang lebih lambat karena juga mempertahankan LinkedList secara internal untuk mempertahankan urutan penyisipan elemen.	TreeSet memberikan kinerja yang lebih rendah daripada HashSet dan LinkedHashSet karena harus mengurutkan elemen setelah setiap operasi penyisipan dan penghapusan.
Operasi Penyisipan, Penghapusan, dan Pengambilan	HashSet memberikan kinerja urutan O(1) untuk operasi	LinkedHashSet juga memberikan kinerja urutan O(1) untuk operasi	TreeSet memberikan kinerja urutan O(log(n)) untuk operasi penyisipan,

	penyisipan, penghapusan, dan pengambilan.	penyisipan, penghapusan, dan pengambilan.	penghapusan, dan pengambilan.
Bagaimana mereka membandingkan elemen?	HashSet menggunakan metode equals() dan hashCode() untuk membandingkan elemen dan dengan demikian menghapus kemungkinan elemen duplikat.	LinkedHashSet juga menggunakan metode equals() dan hashCode() untuk membandingkan elemen.	TreeSet menggunakan metode compare() atau compareTo() untuk membandingkan elemen dan dengan demikian menghapus kemungkinan elemen duplikat. Itu tidak menggunakan metode equals() dan hashCode() untuk perbandingan elemen.
Null elements	HashSet memungkinkan maksimum satu elemen null.	LinkedHashSet juga memungkinkan maksimum satu elemen null.	TreeSet bahkan tidak mengizinkan satu elemen null. Jika Anda mencoba memasukkan elemen null ke TreeSet, itu melempar NullPointerException.
Pekerjaan Memori	HashSet membutuhkan lebih sedikit memori daripada LinkedHashSet dan TreeSet karena hanya menggunakan	LinkedHashSet membutuhkan lebih banyak memori daripada HashSet karena juga memelihara LinkedList bersama dengan	TreeSet juga membutuhkan lebih banyak memori daripada HashSet karena juga memelihara Comparator untuk mengurutkan elemen

	HashMap secara internal untuk menyimpan elemen-elemennya.	HashMap untuk menyimpan elemen-elemennya.	bersama dengan TreeMap.
Kapan Digunakan?	Gunakan HashSet jika Anda tidak ingin mempertahankan urutan elemen apa pun.	Gunakan LinkedHashSet jika Anda ingin mempertahankan urutan penyisipan elemen.	Gunakan TreeSet jika Anda ingin mengurutkan elemen menurut beberapa Comparator.

F. Pemograman

1. HashSet

HashSet merupakan suatu set yang dapat kita gunakan untuk menghindarkan kita dari duplikasi data. Dalam memasukkan data agar tidak terduplikasi maka kita dapat menggunakan metode add() dan nantinya data yang diinputkan lebih dari 1x hanya akan muncul sebanyak 1x di hasil running program. Apabila kita ingin menghapuskan element yang terdapat didalam HashSet kita dapat menggunakan metode remove(), nantinya yang kita remove tidak akan muncul lagi di hasil running program. Sedangkan, untuk mengosongkan semua element kita dapat menggunakan metode clear() , nantinya semua element yang kita inputkan akan kosong. Berikut contoh dari HashSet:

```
import java.util.ArrayList;
import java.util.HashSet;

public class hashset{
    public static void main(String[] args){
        ArrayList<Integer> data_array = new ArrayList<>();
        ArrayList<String> data_array2 = new ArrayList<>();
        HashSet<Integer> data_hashSet = new HashSet<>();
        HashSet<String> data_hashSet2 = new HashSet<>();
        //Memasukan Nilai Default
        //< ArrayList > dengan urutan  1 3 2 5 4
```

```
data_array.add(1);
data_array.add(3);
data_array.add(2);
data_array.add(5);
data_array.add(4);
data_array2.add("Budi");
data_array2.add("anton");

//< HashSet > dengan urutan  1 3 2 5 4
data_hashSet.add(1);
data_hashSet.add(3);
data_hashSet.add(2);
data_hashSet.add(5);
data_hashSet.add(4);
data_hashSet2.add("Budi");
data_hashSet2.add("Anton");

//Memasukan Nilai Duplukat/Yang Sama Dengan Nilai Sebelumnya
 //< ArrayList >
data_array.add(5);
data_array.add(4);
data_array.add(3);
data_array2.add("Budi");
data_array2.add("anton");

 //< HashSet >
data_hashSet.add(5);
data_hashSet.add(4);
data_hashSet.add(3);
data_hashSet2.add("Budi");
data_hashSet2.add("Anton");

//Menampilkan Daftar Nilai
```

```
        System.out.println("Array List :");
        System.out.println("Data Array data_array :" + data_array);
        System.out.println("Data Array data_array :" + data_array2);
        System.out.println("\nHashSet :");
        System.out.println("Data Array data_HashSet :" + data_hashSet);
        System.out.println("Data Array data_HashSet :" + data_hashSet2);
        System.out.println(" ");

        //method menghapus di HashSet menggunakan method Remove()
        data_hashSet.remove(4);
        data_hashSet.remove(3);

        System.out.println("data HashSet sesudah remove : "+ data_hashSet);
        System.out.println(" ");

        // Menghapus semua data pada Hash set
        data_hashSet.clear();
        data_hashSet2.clear();
        System.out.println("Menghapus semua data :");
        System.out.println("Data Array data_HashSet :" + data_hashSet);
        System.out.println("Data Array data_HashSet :" + data_hashSet2);
    }
}
```

Hasil dari running program diatas adalah sebagai berikut :

```
Array List :
Data Array data_array :[1, 3, 2, 5, 4, 5, 4, 3]
Data Array data_array :[Budi, anton, Budi, anton]

HashSet :
Data Array data_HashSet :[1, 2, 3, 4, 5]
Data Array data_HashSet :[Anton, Budi]
```

```
data HashSet sesudah remove : [1, 2, 5]


Menghapus semua data :
Data Array data_HashSet :[]
Data Array data_HashSet :[]
```

2. LinkedHashSet

LinkedHashSet berbeda dengan HashSet ketika kita peduli terhadap urutan iterasi. Bila kita melakukan iterasi melalui HashSet, urutan elemen tidak dapat diprediksi, sedangkan dengan LinkedHashSet memungkinkan kita untuk melakukan iterasi melalui unsur-unsur dalam urutan dimana mereka dimasukkan (*inserted*). Dan juga pada LinkedHashSet ini akan mengabaikan inputan yang memiliki unsur duplikasi. Berikut contoh dari LinkedHashSet:

```
import java.util.LinkedHashSet;


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


      LinkedHashSet<String> menu = new LinkedHashSet<String>();
      menu.add("Mie Pedas");
      menu.add("Ayam Geprek");
      menu.add("Pempek");
      menu.add("Kebab");
      menu.add("Jasuke");
      menu.add("Lemon Tea");


      System.out.println("Menu Awal Expo\t\t\t:" + menu);
      System.out.println("Banyak menu\t\t\t:" + menu.size());
      System.out.println("Menghapus Lemon Tea dari menu\t:" +
menu.remove("Lemon Tea"));
      System.out.println("Menghapus Green Tea dari menu\t:" +
menu.remove("Green Tea"));
```

```
        System.out.println("Menu tanpa minuman\t\t:" + menu);
        System.out.println("Menambah menu Kebab kembali\t:" +
menu.add("Kebab")); // kelebihan linkedlist tidak bisa
                                                        // duplikat data
        System.out.println("Melihat tersedianya Kebab di menu:" +
menu.contains("Kebab"));
        System.out.println("Banyak menu akhir\t\t:" + menu.size());
        System.out.println("Menampilkan menu secara vertikal:");
        for (String strLHS : menu) {
            System.out.println(strLHS);
        }

    }
}
```

Hasil dari running program diatas adalah sebagai berikut :

```
Menu Awal Expo                  :[Mie Pedas, Ayam Geprek, Pempek, Kebab,
Jasuke, Lemon Tea]
Banyak menu                     :6
Menghapus Lemon Tea dari menu   :true
Menghapus Green Tea dari menu   :false
Menu tanpa minuman              :[Mie Pedas, Ayam Geprek, Pempek, Kebab,
Jasuke]
Menambah menu Kebab kembali     :false
Melihat tersedianya Kebab di menu:true
Banyak menu akhir               :5
Menampilkan menu secara vertical :
Mie Pedas
Ayam Geprek
Pempek
Kebab
Jasuke
```

3. TreeSet

TreeSet adalah salah satu implementasi terpenting dari antarmuka SortedSet di Java yang menggunakan Pohon untuk penyimpanan. Untuk menambahkan elemen ke TreeSet, kita dapat menggunakan metode add(). Namun, urutan penyisipan tidak dipertahankan di TreeSet. Secara internal, untuk setiap elemen, nilai dibandingkan dan diurutkan dalam urutan naik. Kita perlu mencatat bahwa elemen duplikat tidak diperbolehkan dan semua elemen duplikat diabaikan. Dan juga, nilai Null tidak diterima oleh TreeSet. Setelah menambahkan elemen, jika kita ingin mengakses elemen, kita dapat menggunakan metode bawaan seperti contains(), first(), last(), dll. Berikut contoh dari TreeSet:

```
import java.util.*;
/**
 * treeSet
 */
public class treeSet {
    public static void main(String[] args) {
        /* Creating a Set interface with
        reference to TreeSet class
        Deklarasi objek bertipe String */
        TreeSet<String> ts = new TreeSet<>();

        // Element ditambahkan menggunakan method add()
        ts.add("SI");
        ts.add("Adalah");
        ts.add("Sistem Informasi");

        // Print semua element pada object
        System.out.println(ts);
        String check = "Sistem Informasi";

        // Memperiksa apakah String check diatas ada pada TreeSet atau tidak
        System.out.println("Contains " + check + " " + ts.contains(check));
```

```
        // Print element pertama pada TreeSet
        System.out.println("First Value " + ts.first());

        // Print element terakhir pada TreeSet
        System.out.println("Last Value " + ts.last());

        String val = "SI";

        /* Mencari nilai apakah nilai lebih besar
        atau lebih kecil dari String diatas*/
        System.out.println("Higher " + ts.higher(val));
        System.out.println("Lower " + ts.lower(val));
    }
}
```

Hasil dari running program diatas adalah sebagai berikut :

```
[Adalah, SI, Sistem Informasi]
Contains Sistem Informasi true
First Value Adalah
Last Value Sistem Informasi
Higher Sistem Informasi
Lower Adalah
```

Link Github Program :

https://github.com/annisacan/Tugas-Kel-2-PBO-B.git

DAFTAR PUSTAKA

Budisma. (2021, 06 3). Apa itu treeset dalam koleksi java? Retrieved 12 1, 2022, from Budisma.net: https://budisma.net/tag/apa-itu-treeset-dalam-koleksi-java.html

Computer Science. 2017. *Hash Tables and Hash Functions*. URL: https://www.youtube.com/watch?v=KyUTuwz_b7Q. Diakses tanggal 1 Desember 2022.

Earth Computer. 2019. *What is a HashSet.* https://www.youtube.com/watch?v=y5Cx07OHaOI. Diakses tanggal 1 Desember 2022.

Edureka. 2019. What is LinkedHashSet in Java? Understand with examples. URL: https://www.edureka.co/blog/linkedhashset-in-java/. Diakses tanggal 4 Desember 2022.

Geeksforgeeks. 2022. *HashSet in Java*. URL: https://www.geeksforgeeks.org/hashset-in-java/. Diakses tanggal 1 Desember 2022.

Kannan, B. B. 2019. UNIT I Set & Map of JCF.

Oracle. (1993,2020). Class Treeset. Retrieved 12 1, 2022, from Oracle: https://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html

Pramodblablad. 2015. HashSet Vs LinkedHashSet Vs TreeSet In Java. URL: https://javaconceptoftheday.com/hashset-vs-linkedhashset-vs-treeset-in-java/. Diakses tanggal 4 Desember 2022.

Pratiwi, N. (2020, 05 20). Apaitu TreeSet? Retrieved 12 1, 2022, from Apa yang Dimaksud: https://apayangdimaksud.com/tag/apa-itu-treeset/index.html

Viska Mutiawani, K. S. (2014, 03 11). Praktikum. Retrieved 12 1, 2022, from Unsyiah: https://informatika.unsyiah.ac.id/~viska/pjl/prak/T4-Collections-Indo/#:~:text=TreeSet%20merupakan%20class%20yang%20sering,dari%20collection%20dalam%20urutan%20tertentu

W3schools. 2022. *Java HashSet*. URL: https://www.w3schools.com/java/java_hashset.asp. Diakses tanggal 1 Desember 2022.