

collection

Nama : Muhamad Wahyu Saputra

Kelas : XII RPL B

Absen : 34

1.Latihan SET

1.HashSet

```
2 package collection:
3
4 import java.util.*;
5
6 public class Tester {
7
8     //set => struktur data yg menyimpan elemen unik(tidak sama)
9     //berguna menyimpan data yg tidak mungkin sama misal => ktp,nomor telepon,email
10    //implementasi set menggunakan Hashset -> tidak memperhatikan posisi data
11
12    public static void main(String[] args) {
13
14        Set<String> set = new HashSet<String>();
15        set.add("08111");
16        set.add("08222");
17        set.add("08333");
18        set.add("08666");
19        set.add("08444");
20        set.add("08555");
21
22        System.out.println(set);
23    }
24 }
25
26
```

Output - STRING (run) X

```
run:
[08666, 08444, 08555, 08222, 08333, 08111]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Berdasarkan program&output diatas menggunakan HashSet hasil outputnya adalah acak

2.LinkedHashSet

```
25
26
27 System.out.println("\n=====LinkedHashSet=====\\n");
28
29 Set<String> set2 = new LinkedHashSet<String>();
30 set2.add("08111");
31 set2.add("08222");
32 set2.add("08333");
33 set2.add("08444");
34 set2.add("08555");
35 set2.add("08666");
36
37 System.out.println(set2);
38
39 }
40
41
```

Output - STRING (run) X

```
run:
=====HashSet=====
[08666, 08444, 08555, 08222, 08333, 08111]
=====LinkedHashSet=====
[08111, 08222, 08333, 08444, 08555, 08666]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jika menggunakan LinkedHashSet Akan urut sesuai inputan

3.TreeSet

```
37 System.out.println("\n=====TreeSet=====\\n");
38
39 Set<String> set3 = new TreeSet<String>();
40
41 set3.add("08111");
42 set3.add("08222");
43 set3.add("08333");
44 set3.add("08666");
45 set3.add("08444");
46 set3.add("08555");
47
48 System.out.println(set3);
49
50 }
51
52 }
```

Output - STRING (run)

```
run:
=====HashSet=====
[08666, 08444, 08555, 08222, 08333, 08111]
=====LinkedHashSet=====
[08111, 08222, 08333, 08444, 08555, 08666]
=====TreeSet=====
[08111, 08222, 08333, 08444, 08555, 08666]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ascending yaitu nilai diurutkan dari yg terkecil sampai yg terbesar jika huruf A -Z

2.Latihan LIST

1.ArrayList

```
1 package collection;
2 import java.util.*;
3
4 public class TesterArrayList {
5
6
7
8     public static void main(String[] args) {
9         List<String> list = new ArrayList<String>();
10        list.add("PEMROGRAMAN BERORIENTASI OBJEK");
11        list.add("PEMROGRAMAN DASAR");
12        list.add("SISTEM KOMPUTER");
13        list.add("JARINGAN DASAR");
14        list.add("ANDROID STUDIO");
15        list.add(1, "PEMROGRAMAN WEB");
16
17        System.out.println(list);
18    }
19 }
20
21 }
22 }
```

Output - STRING (run)

```
run:
[PEMROGRAMAN BERORIENTASI OBJEK, PEMROGRAMAN WEB, PEMROGRAMAN DASAR, SISTEM KOMPUTER, JARINGAN DASAR, ANDROID STUDIO]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Pada output diatas setelah PBO yaitu pemrograman web, karena output dieksekusi berdasarkan indeks ,dimanan pemrograman web berada pada indeks 1

```
7
8
9     public static void main(String[] args) {
10        List<String> list = new ArrayList<String>();
11        list.add("PEMROGRAMAN BERORIENTASI OBJEK");
12        list.add("PEMROGRAMAN DASAR");
13        list.add("SISTEM KOMPUTER");
14        list.add("JARINGAN DASAR");
15        list.add("ANDROID STUDIO");
16        list.add(1, "PEMROGRAMAN WEB");
17
18        //menghapus data berdasarkan index
19        list.remove(0);
20
21        System.out.println(list);
22    }
23 }
24 }
```

Output - STRING (run)

```
run:
[PEMROGRAMAN WEB, PEMROGRAMAN DASAR, SISTEM KOMPUTER, JARINGAN DASAR, ANDROID STUDIO]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Jika ingin menghapus data tertentu bisa menggunakan perintah remove disertai dengan nomor indeks yg akan dihapus, pada array list juga dapat menambah data duplikat.

2. LinkedList Menggunakan Queue

```
1 package collection;
2 import java.util.*;
3 public class LinkedListQueue {
4
5     public static void main(String[] args) {
6         Queue<String> queue = new LinkedList<String>();
7         queue.add("PEMROGRAMAN BERORIENTASI OBJEK");
8         queue.add("PEMROGRAMAN DASAR");
9         queue.add("SISTEM KOMPUTER");
10        queue.add("JARINGAN DASAR");
11        queue.add("ANDROID STUDIO");
12        queue.add("PEMROGRAMAN WEB");
13
14        System.out.println(queue);
15    }
16 }
17
18
19
20
```

Output-STRING (run) x

```
run:
[PEMROGRAMAN BERORIENTASI OBJEK, PEMROGRAMAN DASAR, SISTEM KOMPUTER, JARINGAN DASAR, ANDROID STUDIO, PEMROGRAMAN WEB]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output berurutan sesuai inputan

```
1 package collection;
2 import java.util.*;
3 public class LinkedListQueue {
4
5     public static void main(String[] args) {
6         Queue<String> queue = new LinkedList<String>();
7         queue.add("PEMROGRAMAN BERORIENTASI OBJEK");
8         queue.add("PEMROGRAMAN DASAR");
9         queue.add("SISTEM KOMPUTER");
10        queue.add("JARINGAN DASAR");
11        queue.add("ANDROID STUDIO");
12        queue.add("PEMROGRAMAN WEB");
13
14        queue.remove("ANDROID STUDIO");
15
16        System.out.println(queue);
17    }
18 }
19
20
21
22
```

Output-STRING (run) x

```
run:
[PEMROGRAMAN BERORIENTASI OBJEK, PEMROGRAMAN DASAR, SISTEM KOMPUTER, JARINGAN DASAR, PEMROGRAMAN WEB]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Menghapus data menggunakan perintah remove(index)

```
6 public static void main(String[] args) {
7     Queue<String> queue = new LinkedList<String>();
8     queue.add("PEMROGRAMAN BERORIENTASI OBJEK");
9     queue.add("PEMROGRAMAN DASAR");
10    queue.add("SISTEM KOMPUTER");
11    queue.add("JARINGAN DASAR");
12    queue.add("ANDROID STUDIO");
13    queue.add("PEMROGRAMAN WEB");
14
15    System.out.println(queue + "\n");
16
17    queue.remove("ANDROID STUDIO");
18
19    System.out.println(queue + "\n");
20
21    System.out.println(queue.element());
22 }
23
24
25
26
```

Output-STRING (run) x

```
run:
[PEMROGRAMAN BERORIENTASI OBJEK, PEMROGRAMAN DASAR, SISTEM KOMPUTER, JARINGAN DASAR, ANDROID STUDIO, PEMROGRAMAN WEB]
[PEMROGRAMAN BERORIENTASI OBJEK, PEMROGRAMAN DASAR, SISTEM KOMPUTER, JARINGAN DASAR, PEMROGRAMAN WEB]
PEMROGRAMAN BERORIENTASI OBJEK
BUILD SUCCESSFUL (total time: 0 seconds)
```

Mengambil data di posisi pertama menggunakan perintah .element()

3.PriorityQueue

```
1 package collection;
2 import java.util.*;
3 public class LinkedListPriorityQueue {
4
5
6     public static void main(String[] args) {
7         Queue<String> queue = new PriorityQueue<String>();
8         queue.add("PEMROGRAMAN BERORIENTASI OBJEK");
9         queue.add("PEMROGRAMAN DASAR");
10        queue.add("SISTEM KOMPUTER");
11        queue.add("JARINGAN DASAR");
12        queue.add("ANDROID STUDIO");
13        queue.add("PEMROGRAMAN WEB");
14
15        System.out.println(queue);
16    }
17 }
18
19
```

Output - STRING (run) X

run:
[ANDROID STUDIO, JARINGAN DASAR, PEMROGRAMAN WEB, PEMROGRAMAN DASAR, PEMROGRAMAN BERORIENTASI OBJEK, SISTEM KOMPUTER]
BUILD SUCCESSFUL (total time: 0 seconds)

Tergantung kebutuhannya dan bisa diatur, dimana terdapat kriteria yg dapat diatur di interface comparable

2.Latihan MAP

1.HashMap

```
1 package collection;
2
3 import java.util.*;
4 public class TestHashMap {
5
6
7     public static void main(String[] args) {
8         Map<String, String> map = new HashMap<String, String>();
9         map.put("0011", "Wahyu");
10        map.put("1222", "Laba laba");
11        map.put("0132", "Hakim");
12        map.put("0124", "Dhani");
13        map.put("0022", "paijen");
14
15        System.out.println(map);
16    }
17 }
18
19
20
```

Output - Editor

Debugger Console X STRING (run) X

run:
{0134=Dhani, 0011=Wahyu, 1222=Laba laba, 0132=Hakim, 0022=paijen}
BUILD SUCCESSFUL (total time: 0 seconds)

Output acak

2. Linked Hash Map

```
1 package collection;
2
3
4 import java.util.*;
5 public class TestLinkedHashMap {
6
7
8     public static void main(String[] args) {
9         Map<String, String> map = new LinkedHashMap<String, String>();
10         map.put("0011", "Wahyu");
11         map.put("1222", "Laba laba");
12         map.put("0132", "Hakim");
13         map.put("0124", "Dhani");
14         map.put("0022", "paijan");
15
16
17         System.out.println(map);
18     }
19 }
20
21
```

Output - Editor

Output x

Debugger Console x STRING (run) x

run:

```
{0011=Wahyu, 1222=Laba laba, 0132=Hakim, 0124=Dhani, 0022=paijan}
```

BUILD SUCCESSFUL (total time: 0 seconds)

Data berurutan sesuai inputan

3. TreeMap

```
1 package collection;
2
3
4 import java.util.*;
5 public class TestTreeMap {
6
7
8     public static void main(String[] args) {
9         Map<String, String> map = new TreeMap<String, String>();
10         map.put("0011", "Wahyu");
11         map.put("1222", "Laba laba");
12         map.put("0132", "Hakim");
13         map.put("0124", "Dhani");
14         map.put("0022", "paijan");
15
16         //menghapus data dengan key nya
17         map.remove("0011");
18
19         System.out.println(map);
20     }
21 }
22
23
```

Ascending nilai dari kecil ke besar dan menghapus data dengan key nya dan

```
3
4 import java.util.*;
5 public class TestTreeMap {
6
7
8     public static void main(String[] args) {
9         Map<String, String> map = new TreeMap<String, String>();
10         map.put("0011", "Wahyu");
11         map.put("1222", "Laba laba");
12         map.put("0132", "Hakim");
13         map.put("0124", "Dhani");
14         map.put("0022", "paijan");
15
16         //menghapus data dengan key nya
17         map.remove("0011");
18
19         System.out.println(map);
20
21         System.out.println(map.get("1222"));
22     }
23 }
24
```

Output - Editor

Output x

Debugger Console x STRING (run) x

run:

```
{0022=paijan, 0124=Dhani, 0132=Hakim, 1222=Laba laba}
```

Laba laba

BUILD SUCCESSFUL (total time: 0 seconds)

mengambil data dengan method get

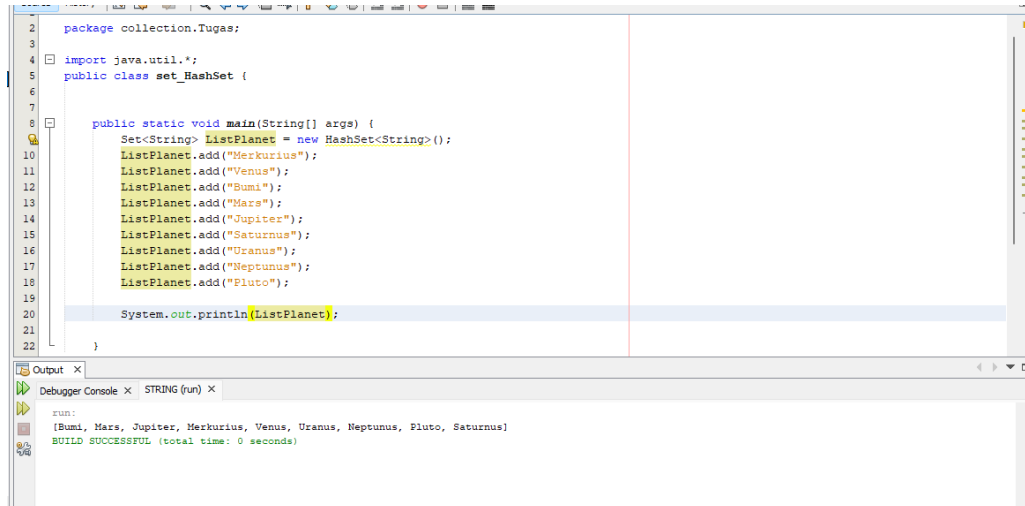
TUGAS 1

Collection Set

Yaitu struktur data yg menampung elemen2 unik(tidak boleh sama),
Berguna untuk mengelola data yg tidak mungkin sama misal => (KTP,Email,no hp)

IMPLEMENTASI SET:

1.HashSet



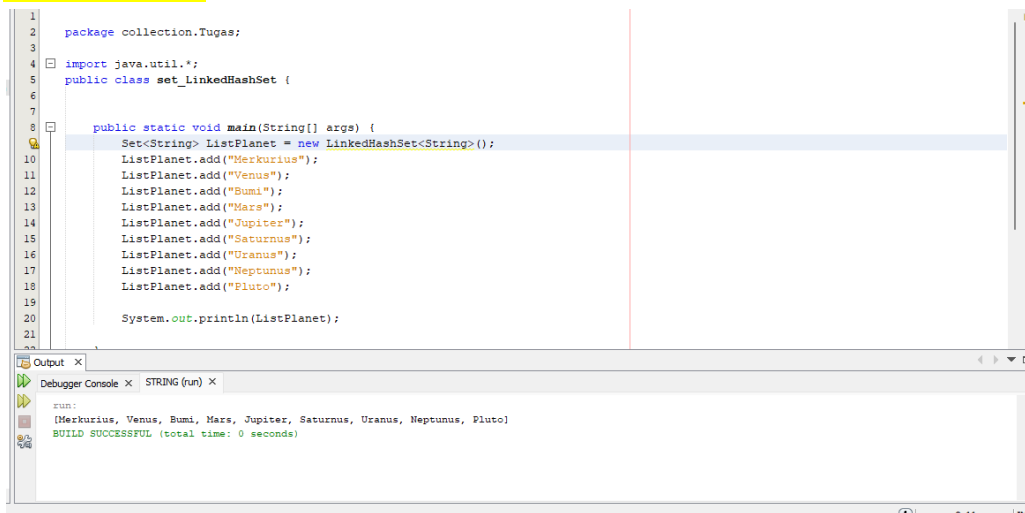
```
1 package collection.Tugas;
2
3
4 import java.util.*;
5 public class set_HashSet {
6
7
8     public static void main(String[] args) {
9         Set<String> ListPlanet = new HashSet<String>();
10        ListPlanet.add("Merkurius");
11        ListPlanet.add("Venus");
12        ListPlanet.add("Bumi");
13        ListPlanet.add("Mars");
14        ListPlanet.add("Jupiter");
15        ListPlanet.add("Saturnus");
16        ListPlanet.add("Uranus");
17        ListPlanet.add("Neptunus");
18        ListPlanet.add("Pluto");
19
20        System.out.println(ListPlanet);
21    }
22 }
```

Output:

```
run:
[Bumi, Mars, Jupiter, Merkurius, Venus, Uranus, Neptunus, Pluto, Saturnus]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Implementasi HashSet menghasilkan output yg acak(tidak urut)sesuai inputan

2.LinkedHashSet



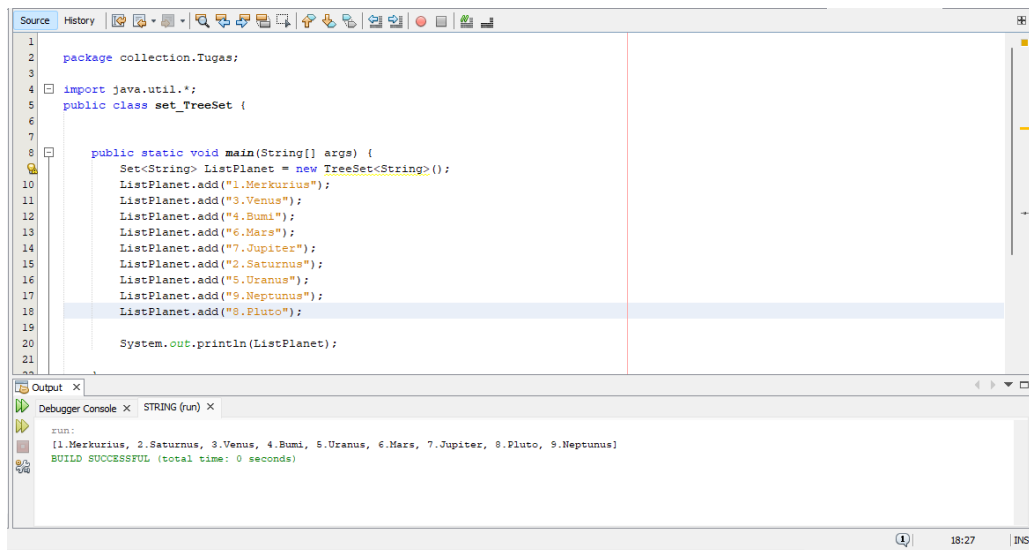
```
1 package collection.Tugas;
2
3
4 import java.util.*;
5 public class set_LinkedHashSet {
6
7
8     public static void main(String[] args) {
9         Set<String> ListPlanet = new LinkedHashSet<String>();
10        ListPlanet.add("Merkurius");
11        ListPlanet.add("Venus");
12        ListPlanet.add("Bumi");
13        ListPlanet.add("Mars");
14        ListPlanet.add("Jupiter");
15        ListPlanet.add("Saturnus");
16        ListPlanet.add("Uranus");
17        ListPlanet.add("Neptunus");
18        ListPlanet.add("Pluto");
19
20        System.out.println(ListPlanet);
21    }
22 }
```

Output:

```
run:
[Merkurius, Venus, Bumi, Mars, Jupiter, Saturnus, Uranus, Neptunus, Pluto]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Implementasi LinkedHashSet menghasilkan output urut sesuai inputan

3.TreeSet



```
1 package collection.Tugas;
2
3 import java.util.*;
4 public class set_TreeSet {
5
6     public static void main(String[] args) {
7         Set<String> ListPlanet = new TreeSet<String>();
8         ListPlanet.add("1.Merkurius");
9         ListPlanet.add("3.Venus");
10        ListPlanet.add("4.Bumi");
11        ListPlanet.add("6.Mars");
12        ListPlanet.add("7.Jupiter");
13        ListPlanet.add("2.Saturnus");
14        ListPlanet.add("5.Uranus");
15        ListPlanet.add("9.Neptunus");
16        ListPlanet.add("8.Pluto");
17
18        System.out.println(ListPlanet);
19    }
20 }
21
```

Output:

```
run:
[1.Merkurius, 2.Saturnus, 3.Venus, 4.Bumi, 5.Uranus, 6.Mars, 7.Jupiter, 8.Pluto, 9.Neptunus]
BUILD SUCCESSFUL (total time: 0 seconds)
```

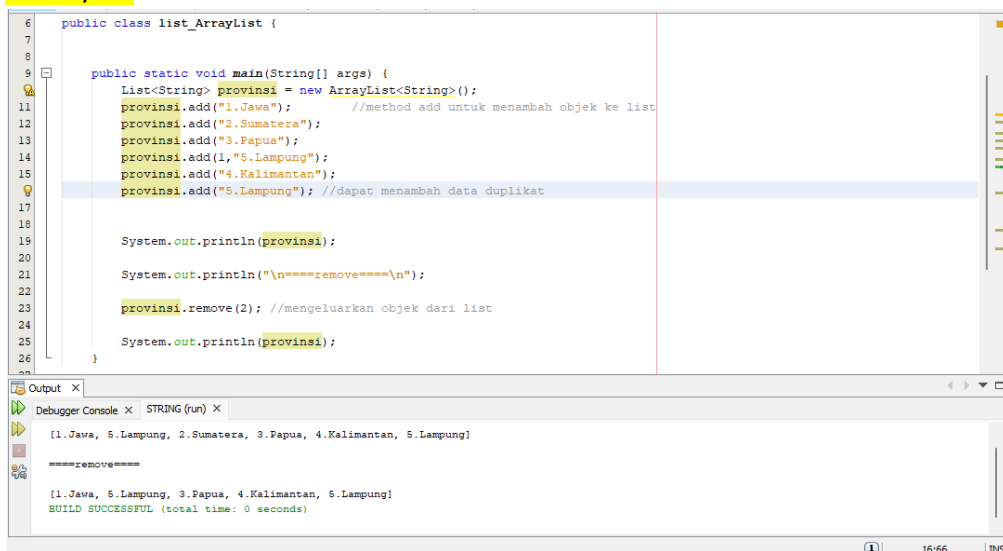
Implementasi TreeSet menghasilkan nilai ascending/ nilai dari kecil ke besar

Collection List

- * List Menyimpan data secara sekuensial seperti array sehingga dapat diakses menggunakan sistem indexing.
- * dapat menyimpan elemen duplikat, user dapat menentukan penyimpanan elemen
- * berguna untuk mengelola data yg perlu memperhatikan posisi data
contoh => (Presensi siswa, kursi bioskop)

IMPLEMENTASI List:

1.ArrayList



```
6 public class list_ArrayList {
7
8     public static void main(String[] args) {
9         List<String> provinsi = new ArrayList<String>();
10        provinsi.add("1.Jawa"); //method add untuk menambah objek ke list
11        provinsi.add("2.Sumatera");
12        provinsi.add("3.Papua");
13        provinsi.add("5.Lampung");
14        provinsi.add("4.Kalimantan");
15        provinsi.add("5.Lampung"); //dapat menambah data duplikat
16
17        System.out.println(provinsi);
18
19        System.out.println("\n====remove====\n");
20
21        provinsi.remove(2); //mengeluarkan objek dari list
22
23        System.out.println(provinsi);
24    }
25 }
26
```

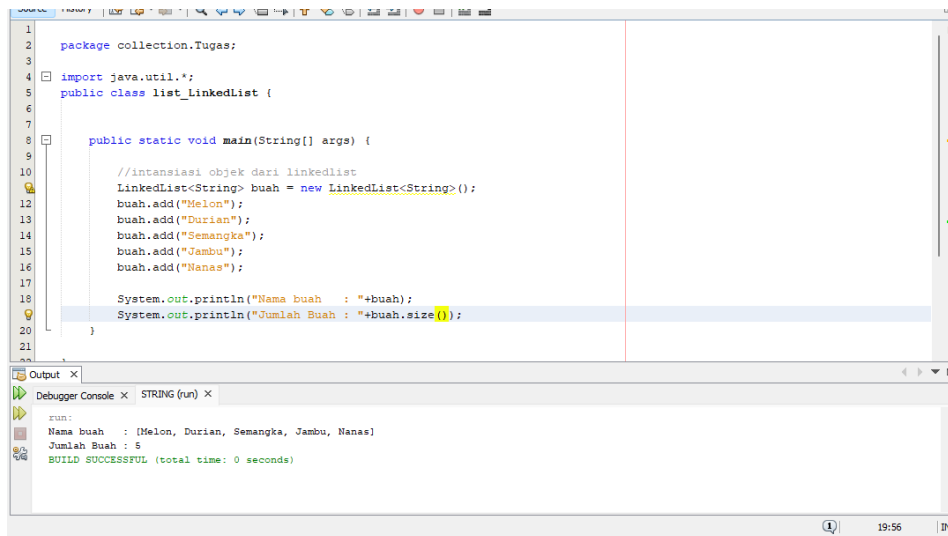
Output:

```
run:
[1.Jawa, 5.Lampung, 2.Sumatera, 3.Papua, 4.Kalimantan, 5.Lampung]
====remove====
[1.Jawa, 5.Lampung, 3.Papua, 4.Kalimantan, 5.Lampung]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output tersebut dipanggil berdasarkan index ,itulah sebabnya data LAMPUNG ditampilkan setelah jawa,karena data lampung berada pada index 1.

Array list juga dapat menambah data duplikat dan jika ingin menghapus gunakan perintah => remove(index data)

2. LinkedList



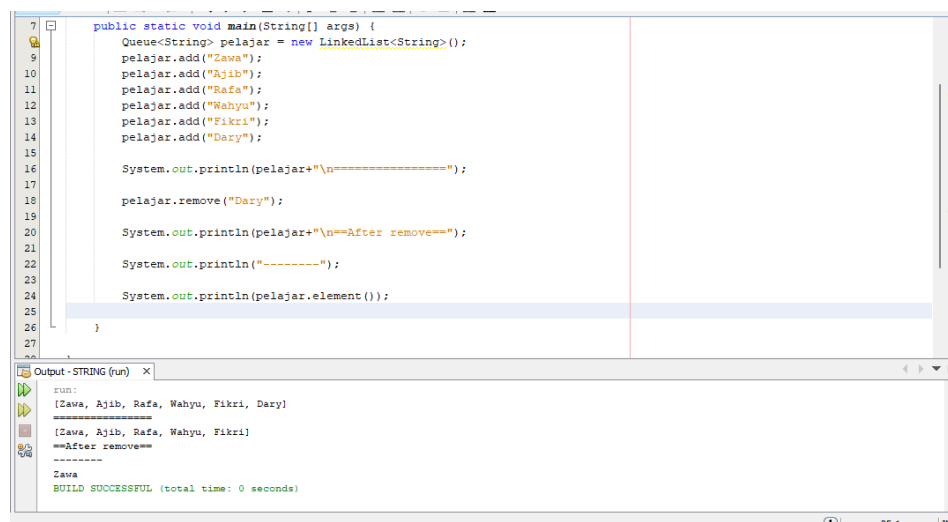
```
1 package collection.Tugas;
2
3 import java.util.*;
4 public class list_LinkedList {
5
6
7     public static void main(String[] args) {
8
9         //intansiasi objek dari linkedlist
10        LinkedList<String> buah = new LinkedList<String>();
11        buah.add("Melon");
12        buah.add("Durian");
13        buah.add("Semangka");
14        buah.add("Jambu");
15        buah.add("Nanas");
16
17        System.out.println("Nama buah : "+buah);
18        System.out.println("Jumlah Buah : "+buah.size());
19    }
20
21 }
```

Output:

```
run:
Nama buah : [Melon, Durian, Semangka, Jambu, Nanas]
Jumlah Buah : 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Untuk menambahkan data pada Objek buah, kita menggunakan fungsi add(), lalu menampilkan nilainya pada statement System.out.println(), dan untuk melihat ukuran atau jumlah data yang terdapat pada Objek buah, kita bisa menggunakan fungsi size().

3. Queue dengan linked list



```
7     public static void main(String[] args) {
8         Queue<String> pelajar = new LinkedList<String>();
9         pelajar.add("Zawa");
10        pelajar.add("Ajib");
11        pelajar.add("Rafa");
12        pelajar.add("Wahyu");
13        pelajar.add("Fikri");
14        pelajar.add("Dary");
15
16        System.out.println(pelajar+"\n=====");
17
18        pelajar.remove("Dary");
19
20        System.out.println(pelajar+"\n==After remove==");
21
22        System.out.println("-----");
23
24        System.out.println(pelajar.element());
25
26    }
27 }
```

Output - STRING (run):

```
run:
[Zawa, Ajib, Rafa, Wahyu, Fikri, Dary]
=====
[Zawa, Ajib, Rafa, Wahyu, Fikri]
==After remove==
-----
Zawa
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output sesuai inputan, menghapus data perintah remove(), mengambil data awal menggunakan perintah element()

4.PriorityQueue

```
3
4 import java.util.*;
5 public class List_PriorityQueue {
6
7
8     public static void main(String[] args) {
9
10         Queue<String> buah = new PriorityQueue<String>();
11         buah.add("Melon");
12         buah.add("Durian");
13         buah.add("Semangka");
14         buah.add("Jambu");
15         buah.add("Nanas");
16
17         System.out.println("Nama buah : "+buah);
18         System.out.println("Jumlah Buah : "+buah.size());
19     }
20 }
21
22
```

Output - STRING (run) X

```
run:
Nama buah : [Durian, Jambu, Semangka, Melon, Nanas]
Jumlah Buah : 5
BUILD SUCCESSFUL (total time: 0 seconds)
```

Tergantung kebutuhan + bisa diatur ,ada kriteria yg dapat diatur di interface comparable

Collection Map

- * Merupakan container yg menyimpan elemen bersama dengan kuncinya(index), map menggunakan sistem indexing yg disebut dengan Key
- * Kunci Harus Unik/tidak boleh sama misal => penduduk(Key : nomor KTP)

1.HashMap

```
1
2 package collection.Tugas;
3
4 import java.util.*;
5 public class map_HashMap {
6
7
8     public static void main(String[] args) {
9
10         Map<String, String> PendudukMars = new HashMap<String, String>();
11         PendudukMars.put("0001", "Raffa");
12         PendudukMars.put("0321", "Manshur");
13         PendudukMars.put("0456", "Petrurk");
14         PendudukMars.put("00131", "Ajib");
15         PendudukMars.put("00211", "Wahyu");
16
17         System.out.println(PendudukMars);
18     }
19 }
20
```

Output - STRING (run) X

```
run:
{0456=Petrurk, 0321=Manshur, 00131=Ajib, 0001=Raffa, 00211=Wahyu}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output Menggunakan HashMap tidak memperhatikan posisi data(acak)

2. LinkedHashMap

```
1 package collection.Tugas;
2
3 import java.util.*;
4 public class map_LinkedHashMap {
5
6
7
8
9
10 public static void main(String[] args) {
11     Map<String, String> PendudukMars = new LinkedHashMap<String, String>();
12     PendudukMars.put("Raf", "Raffa");
13     PendudukMars.put("Man", "Manshur");
14     PendudukMars.put("Pet", "Petruk");
15     PendudukMars.put("Aj", "Ajib");
16     PendudukMars.put("Wah", "Wahyu");
17
18     System.out.println("===LinkedHashMap===");
19     System.out.println(PendudukMars);
20 }
21 }
```

Output - STRING (run) x

```
run:
===LinkedHashMap===
{Raf=Raffa, Man=Manshur, Pet=Petruk, Aj=Ajib, Wah=Wahyu}
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output Urut Sesuai Inputan

3. TreeMap

```
4 import java.util.*;
5 public class map_TreeMap {
6
7
8
9
10 public static void main(String[] args) {
11     Map<String, String> PendudukMars = new TreeMap<String, String>();
12     PendudukMars.put("0001", "Raffa");
13     PendudukMars.put("0321", "Manshur");
14     PendudukMars.put("0456", "Petruk");
15     PendudukMars.put("00131", "Ajib");
16     PendudukMars.put("00211", "Wahyu");
17
18     System.out.println(PendudukMars);
19
20     System.out.println("====sesudah di remove====");
21
22     PendudukMars.remove("0001");
23     System.out.println(PendudukMars);
24     System.out.println("====Ambil Data====");
25     System.out.println(PendudukMars.get("00211"));
26 }
```

Output - STRING (run) x

```
run:
{0001=Raffa, 00131=Ajib, 00211=Wahyu, 0321=Manshur, 0456=Petruk}
====sesudah di remove====
{00131=Ajib, 00211=Wahyu, 0321=Manshur, 0456=Petruk}
====Ambil Data====
Wahyu
BUILD SUCCESSFUL (total time: 0 seconds)
```

Output Ascending nilai dari kecil ke besar,remove kemudian masukkan key untuk menghapus data tertentu,method get untuk mengambil data dari key.

```
=====
=====()=====()=====
=====
-----()-----
-----
```