

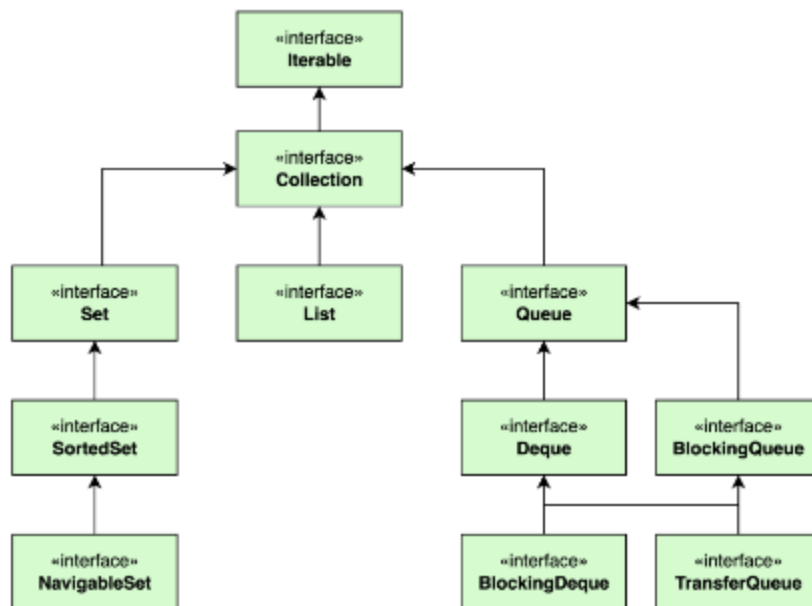
## Summary Iterable Data Structure

Nama : Theresia Rumapea

Kelas : QE-A

### ▼ PENGERTIAN ITERABLE

Sekumpulan data atau struktur data yang dapat dilakukan iterasi/perulangan kepadanya. Berikut Hirarki Interface pada Java Iterable. Iterable adalah struktur tertinggi dalam hirarki dalam bahasa pemrograman java seperti collection dan list.



- <interface> Iterable
  - <interface> Collection
    - <interface> Set
      - <interface> SortedSet
      - <interface> Navigable
    - <interface> List

- <interface> Queue
  - <interface> Deque
  - <interface> BlockingDeque
  - <interface> BlockingDeque
  - <interface>

## ▼ PENGERTIAN INTERFACE

Seluruh daftar interface dalam hierarchy merupakan struktur data yang dapat dilakukan perulangan terhadap masing-masing data yang disimpan.

General-Purpose Implementations adalah implementasi yang paling sering digunakan yang didesain untuk penggunaan setiap hari. Mereka dirangkum dalam tabel berjudul General-purpose-implementations.

Interfaces	Hash table Implementations	Resizable array Implementations	Tree Implementations	Linked List Implementations	Hash table + Linked list Implementations
Set	HashSet		Treeset		LinkedHashSet
List		ArrayList		LinkedList	
Deque		ArrayDeque		LinkedList	

Artinya jika kita ingin menggunakan struktur data dari masing-masing implementasi tersebut, kita juga bisa menggunakan interface (set, list, dan deque) sebagai tipe data dari variabel tempat kita menyimpan kumpulan data tersebut.

Berikut cara menggunakan interface iterable :

```

Method dalam kelas Iterable:
forEach(Consumer<? super T>): void
iterator(): Iterator<T>
spliterator: Spliterator<T>

//=====Contoh:
public static void main(String[] args) {
    Iterable<string> names = List.of("alterra", "academy");
    for (var name: name) {
        System.out.println(name) ;
    }
}

```

```

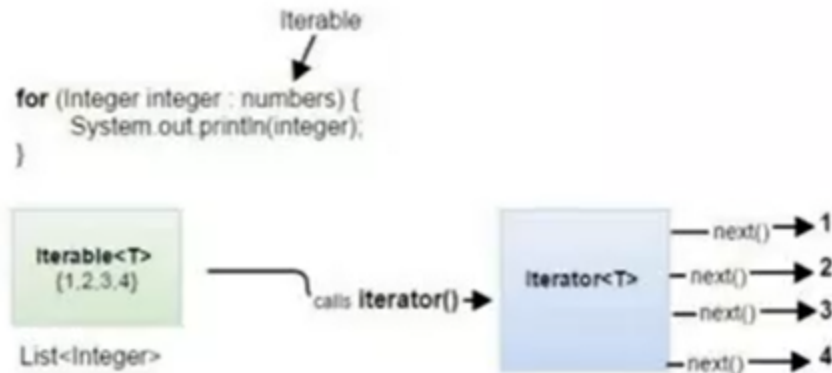
    }
}
//=====Output:
alterra
academy

```

## ▼ PENGERTIAN ITERATOR

Interface iterable memiliki satu method iterator()

Iterator adalah kelas yang mengelola iterasi di atas iterable. itu mempertahankan keadaan di mana kita berada dalam iterasi saat ini, dan tahu apa elemen atau data selanjutnya yang akan diambil dan bagaimana mendapatkannya



Contoh iterable :

```

import java.util.Iterator;
import java.util.List;
Public class Main {
    public static void main(String[] args) {
        Iterable<string> names = Li  st.of("alterra", "academy");
        Iterator<String> it = names.iterator();
        while(it.hasNext()) {
            String.i = it.next();
            System.out.println(i) ;
        }
    }
}
//=====Output :
alterra
academy

```

## ▼ PENGERTIAN COLLECTION

Collection merupakan kumpulan dari objek atau data yang diletakkan dalam satu tempat yang sama (Collection of Objects).

Collection adalah setiap kelompok objek individu yang direpresentasikan sebagai satu unit adalah untuk mengetahui sebagai kumpulan objek

Method Collection

- add()
- addAll()
- clear()
- contains()
- containsAll()
- equals()
- hashCode()
- isEmpty()
- Iterator()
- remove()
- removeAll()
- stream()
- toArray()

=====

Contoh:

```
public static void main(String[] args) {  
    Collection<String> names = new ArrayList<>();  
    names.add("Alterra");  
    names.add("Academy");  
    names.add("Coba");  
    names.remove("Coba");  
    System.out.println(names.contains("Aterra"));  
}
```

```
//=====Output :  
true
```

## ▼ PENGERTIAN LIST

1. Merupakan sebuah interface yang menyediakan cara untuk menyimpan data secara linear
2. List merupakan turunan dari interface collection
3. List juga menerima nilai yang sama , sehingga apabila meng-input 2kali data yang sama , tetap dapat dilakukan
4. Urutan data pada list tergantung kapan data tersebut dimasukkan kedalam list, maka data yang dimasukkan pertama , akan menjadi urutan pertama , maka urutan selanjutnya akan menyusul ke urutan berikutnya.

Method pada List :

- Menambah
- Menghapus
- Mengecek

Apakah data tersedia dalam list , dan mengecek apakah list yang satu sama dengan list yang lainnya.

## ▼ IMPLEMENTASI DARI INTERFACE LIST

### ▼ Abstract List

### ▼ Array List

- Menyediakan penyimpanan data yang dinamis artinya tidak terikat dengan kapasitas yang dapat disimpan oleh ArrayList tersebut
- Data yang disimpan oleh ArrayList bisa sebanyak apapun , selama memori data dari komputer masih sanggup untuk menyimpan data tersebut

### ▼ Array List Work

- Step 1 : Pembuatan Array List (Array Initialization) yang data nya masih kosong

- Step 2 : Menambah elemen pada Array List , maka masing2 posisi akan terisi
- Step 3 : Ketika data sudah penuh , lalu kita tetap memaksa memasukkan data kedalam Array List maka,
- Step 4 : Array List akan membuat array baru dan memindahkan seluruh data pada array tersebut
- Step 5 : Akan dilakukan hal yang sama , jika kapasitas penyimpanan sudah penuh

Maka sebetulnya panjang dari Array List terbatas , tetapi Array List dapat memanjangkan dirinya sehingga seolah-olah data yang dapat disimpan itu tidak terbatas.

#### ▼ Contoh Cara Kerja Array List

## Array List

```
public static void main(String[] args) {
    List<Integer> umur = new ArrayList<>();
    umur.add(17);
    umur.add(60);
    umur.add(30);
    System.out.println(umur.get(2));
}
```

Line 2 : terdapat variabel yang isi array list nya kosong

Line 3-5 : pada variabel tersebut , akan kita tambahkan 3 buah data

Line 6 : akan kita tampilkan pada layar data yang terletak pada urutan ke 2

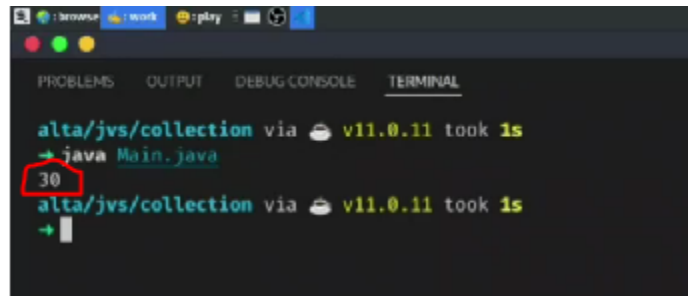
Array List memberi urutan data mulai dari 0 , sehingga :

Urutan 0 = 17

Urutan 1 = 60

Urutan 2 = 30

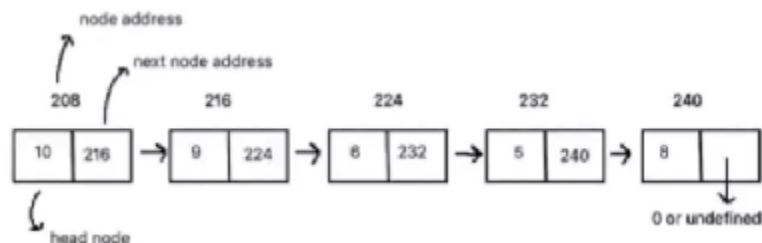
Maka yang akan tampil ke layar adalah 30



```
alta/jvs/collection via v11.0.11 took 1s
→ java Main.java
30
alta/jvs/collection via v11.0.11 took 1s
→
```

### ▼ LinkedList

- Merupakan struktur data yang menyimpan setiap elemen nya saling terhubung satu sama lain , cara Linked List menghubungkan satu elemen dengan elemen yang lainnya adalah dengan menyimpan alamat memori dari masing-masing elemen , baik elemen setelah atau sebelumnya .
- Jadi apabila ada sebuah elemen terdapat dalam Linked List , pada elemen tersebut akan menyimpan alamat memori dari elemen yang sebelumnya
- Lalu elemen sebelumnya juga akan menyimpan alamat memori dari elemen yang sebelumnya lagi



- Contoh linkedlist:

```

public static void main(String[] args) {
    List<Integer> umur = new LinkedList<>();
    umur.add(17);
    umur.add(60);
    umur.add(30);
    System.out.println(umur.get(2));
}
//=====Output:
30

```

## ▼ Perbedaan Array List dan Linked List

### ▼ 1. Secara Structure /Alur :

Pada Array List → Terbentuk ruang kosong , lalu akan diisi data  
pada Linked List :

- Berbentuk seperti gerbong kereta api , elemen pertama akan menyimpan data yang akan disimpan (tidak kosong)
- menyimpan alamat memori data yang berikutnya, maka akan terhubung elemen satu dengan elemen yang lainnya

### ▼ 2. Secara Time Complexity :

## Time Complexity

### ArrayList vs LinkedList

	ArrayList	LinkedList
Accessing element	$O(1)$	$O(n)$
Insert / remove from begining	$O(n)$	$O(1)$
Insert / remove from end	$O(n)$	$O(n)$
Insert / remove from middle	$O(n)$	$O(n)$



a. Accessing element :

- Array List  $O(1)$  artinya konstan
- Linked List  $O(n)$  artinya mengakses elemen yang memiliki kemungkinan untuk memerlukan waktu yang lebih lama

b. Insert/remove from beginning

- Array List  $O(n)$  artinya bisa lebih lama dari Linked List
- Linked List memiliki waktu yang lebih cepat yaitu  $O(1)$

c. Insert / remove from end

- Array List dan Linked List memiliki waktu yang sama

d. Insert /remove from middle

- Array List dan Linked List memiliki waktu yang sama

### ▼ Immutable List

Immutable list merupakan bagian dari list tapi tidak dapat diubah datanya.

```
public static void main(String[] args) {
    List<Integer> umur = new LinkedList<>();
    var umurImmutable = Collections.unmodifiableList(umur);
    umurImmutable.add(2); // tidak diizinkan
    umur.add(17);
    umur.add(60);
    umur.add(30);
    System.out.println(umur.get(2));
}

//=====Output
error
```

### ▼ Stack

Stack menyimpan data secara linear tetapi dalam proses penambahan dan pengambilan datanya, menggunakan last-in-first-out (LIFO), artinya data yang baru dimasukkan akan dikeluarkan terlebih dahulu Contohnya tumpukan buku.

▼ Method:

- Stack();

- `empty()`: boolean
- `peek()`: E, method untuk mengintip data
- `pop()`: E, method untuk mengambil data
- `push(E)` : E, method untuk menambah data
- `search[Object]`: Int
- `serialVersion`

▼ Contoh :

```
public static void main(String[] args) {
    Stack<Integer> umur = new Stack<>();
    umur.push(7);
    umur.push(10);
    umur.push(5);
    System.out.println(umur.pop());
}

//=====Output:
5
```

▼ **Set**

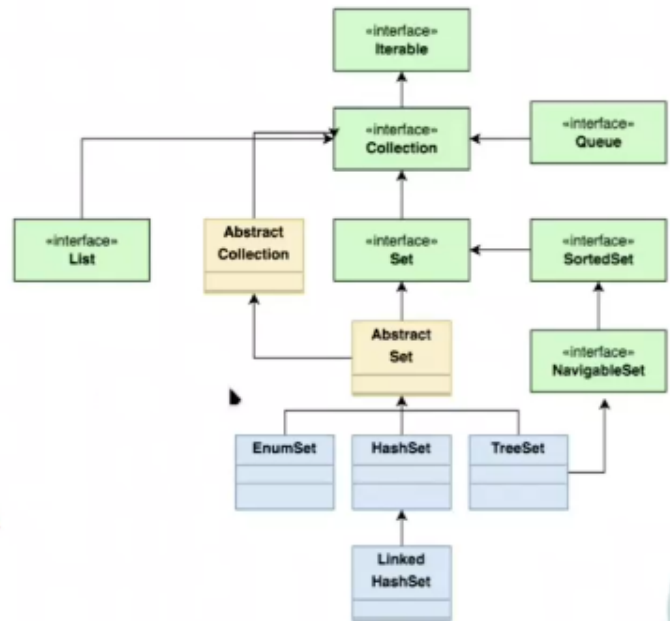
Set merupakan sebuah struktur data yang dapat menyimpan sekumpulan data secara linear seperti list, tetapi set tidak dapat menerima data yang duplikat.

Hierarchy Interface & Implementation



## Set

### Hierarchy Interface & Implementation



### ▼ COMPARING SET (HashSet VS LinkedHashSet VS EnumSet VS TreeSet)

	HashSet	LinkedHashSet	TreeSet	EnumSet
Data Structure	Hash Table	Hash Table + Linked List	Red-Black-Tree	Bit Vector
Sorting	No	Insertion Order	Sorted	Natural Order
Iterator	Fail-Fast	Fail-Fast	fail-Fast	Weakly Consistent
Nulls	Yes	Yes	Depends	No

### ▼ HashSet

Method-method tersebut dapat digunakan untuk mengelola seluruh data atau elemn yang terdapat dalam set. Hash set disebut sebagai struktur data yang menyediakan cara tercepat untuk proses pencarian data.

- HashSet();
- add();
- clear();

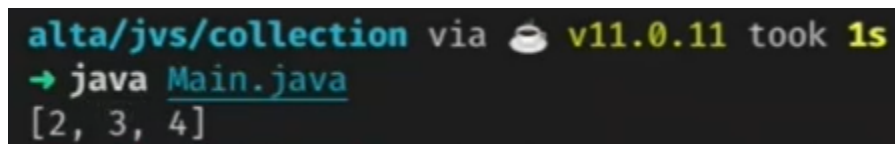
- close();
- isEmpty();
- iterator();
- remove()

Contoh:

```
public static void main(String[] args) {
    Set<Integer> umur = new HashSet<>();
    umur.add(3);
    umur.add(3);
    umur.add(4);
    umur.add(2);
    System.out.println(umur);
}
```

Output  
[2, 3, 4]

jika dari gambar diatas menjelaskan bahwa, tidak menyediakan insectionorder dimana urutan data tidak sesuai dengan apa yang dimasukkan tetapi jdiurutkan berdasarkan ninail dari data nya dari 2, 3 dan 4. Menurut kode yang dibawah ini seharusnya hasil nya mengikuti insectionorder yaitu : 3, 4 dan 2. tetapi hashSet tidak menggunakan mekanisme insection order.

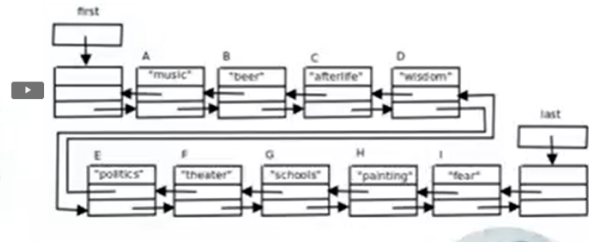


```
alta/jvs/collection via v11.0.11 took 1s
→ java Main.java
[2, 3, 4]
```

## ▼ LindkedHashSet

# LinkedHashSet

- Java LinkedHashSet class contains **unique elements** only like HashSet.
- Java LinkedHashSet class provides all optional set operation and permits **null elements**.
- Java LinkedHashSet class is non **synchronized**.
- Java LinkedHashSet class maintains **insertion order**.



Dari gambar diatas menjelaskan bahwa LinkedHashSet menyimpan insection order dimana, data yang di input lebihdahulu akan terdapat pada urutan yang pertama.

Contoh :

```
public static void main(String[] args) {  
    Set<Integer> umur = new LinkedHashSet<>();  
    umur.add(3);  
    umur.add(3);  
    umur.add(4);  
    umur.add(2);  
    System.out.println(umur);  
}  
//=====Output:  
[3, 4, 2]
```

Pada code diatas di jelakan bahwa yang awal nya variabel “umur berisikan HashSet di ubah menjadi LinkedHashSet”

## ▼ EnumSet

# EnumSet

- It can contain only *enum* values and all the values have to belong to the same *enum*
- It doesn't allow to add null values, throwing a *NullPointerException* in an attempt to do so
- It's not thread-safe, so we need to synchronize it externally if required
- The elements are stored following the order in which they are declared in the *enum*
- It uses a fail-safe iterator that works on a copy, so it won't throw a *ConcurrentModificationException* if the collection is modified when iterating over it

Enum Set digunakan untuk menyimpan nilai “enum” . untuk urutan nya mengikuti penulisan enum tersebut.

Contoh :

```
public class Main {  
  
    enum months{  
        JANUARY, FEBRUARY, MARCH, MAY, JUNE, JULY, AUGUST, SEPTEMBER,  
        OCTOBER, NAVEMBER, DECEMBER  
    }  
  
    public static void main(String[] args) {  
        Set<months> set = EnumSet.allof(months.class);  
        Iterator<months> iter = set.iterator();  
        while (iter.hasNext())  
            System.out.println(iter.next());  
    }  
}
```

HasilRun

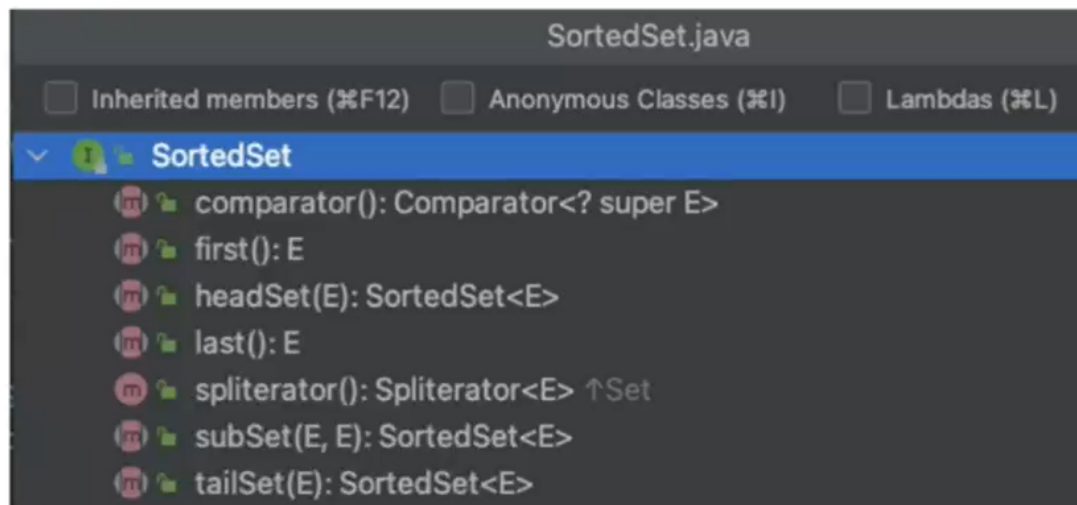
```

alta/jvs/collection via ☕ v11.0.11 took 1s
→ java Main.java
JANUARY
FEBRUARY
MARCH
MAY
JUNE
JULY
AUGUST
SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER
alta/jvs/collection via ☕ v11.0.11 took 1s

```

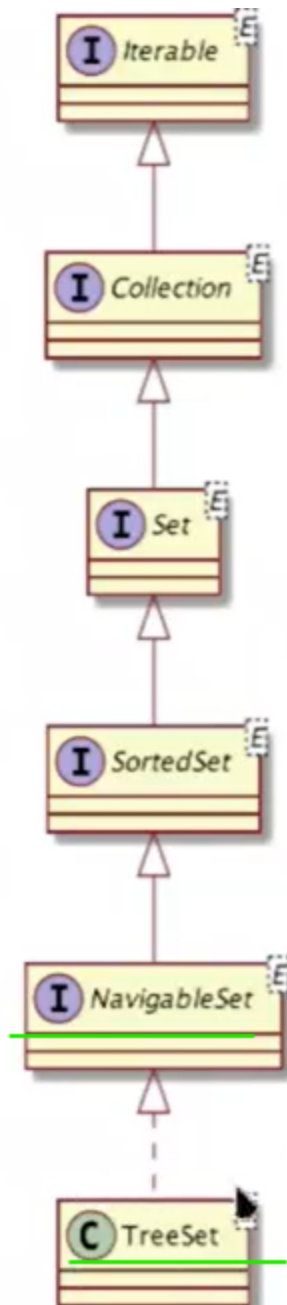
#### ▼ SortedSet

Berikut method-method yang digunakan :



- `comparator()`
- `first()`
- `headSet()`
- `last()`

- splititerator()
  - subSet()
  - tailSet()
- ▼ Berikut Implementasi Sorted Set





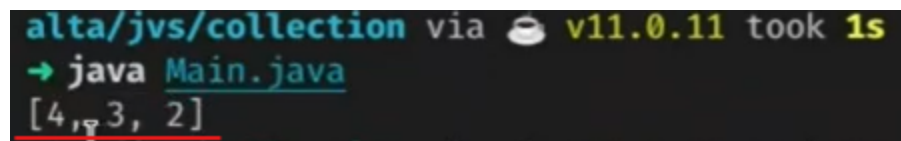
## ▼ NavigableSet

```
public static void main(String[] args) {  
    NavigableSet<Integer> umur = new TreeSet<>();  
    NavigableSet<Integer> umurDesc = umur.descendingSet<>();  
    umur.add(3);  
    umur.add(3);  
    umur.add(4);  
    umur.add(2);  
    System.out.println(umurDesc);  
}
```

Penjelasan :

umur.descendingSet<>(); berfungsi dalam proses perubahan urutan pada variable “umur”

**Hasil Run:**



```
alta/jvs/collection via v11.0.11 took 1s  
→ java Main.java  
[4, 3, 2]
```

Pada gambar diatas menunjukan bahwa akan menampilkan semua yang ada di data set namun urutan nya terbalik.

## ▼ Queue

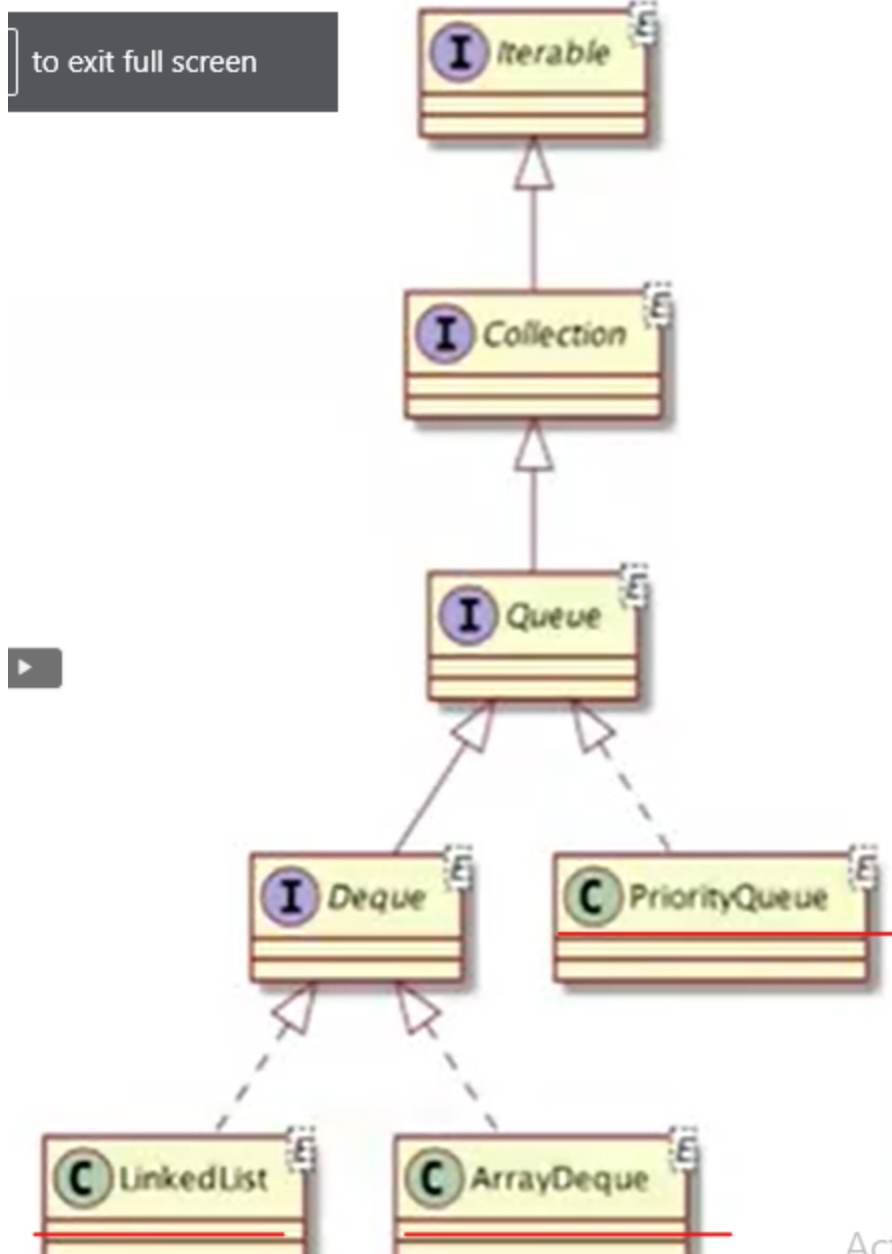
# Queue

1. A queue can be defined as an ordered list which enables insert operations to be performed at one end called REAR and delete operations to be performed at another end called FRONT.
2. Queue is referred to be as First In First Out list.
3. For example, people waiting in line for a rail ticket form a queue.



Mengimplementasi mekanisme first out artinya data yang terlebih dahulu di masukkan akan lebih dahulu di ambil sehingga Queue dapat di ibaratkan “antrian” . untuk mengambil data pada queue akan di gunakan dequeue berfungsi untuk mengambil data yang paling depan dari sebuah queue. kemudian untuk menambahkan data kita bisa menggunakan enqueue berfungsi untuk menambahkan data pada bagian belakang karena pada konsep antrian jika mengantri itu berbaris pada belakang.

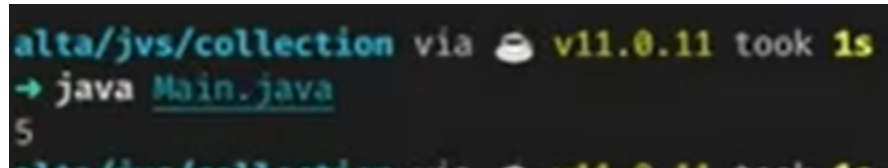
## ▼ Implementasi Queue




- Berikut ini code pada Queue :

```
public static void main(String[] args) {  
    Queue<Integer> umur = new PriorityQueue<>();  
    umur.add(7);  
    umur.add(10);  
    umur.add(5);  
    System.out.println(umur.poll());  
}
```

Hasil Run :

A terminal window with a dark background. The first line shows a command prompt 'alta/jvs/collection' followed by 'via' and a Docker icon, then 'v11.0.11 took 1s'. The second line shows a command '→ java Main.java'. The third line shows the output '5'.

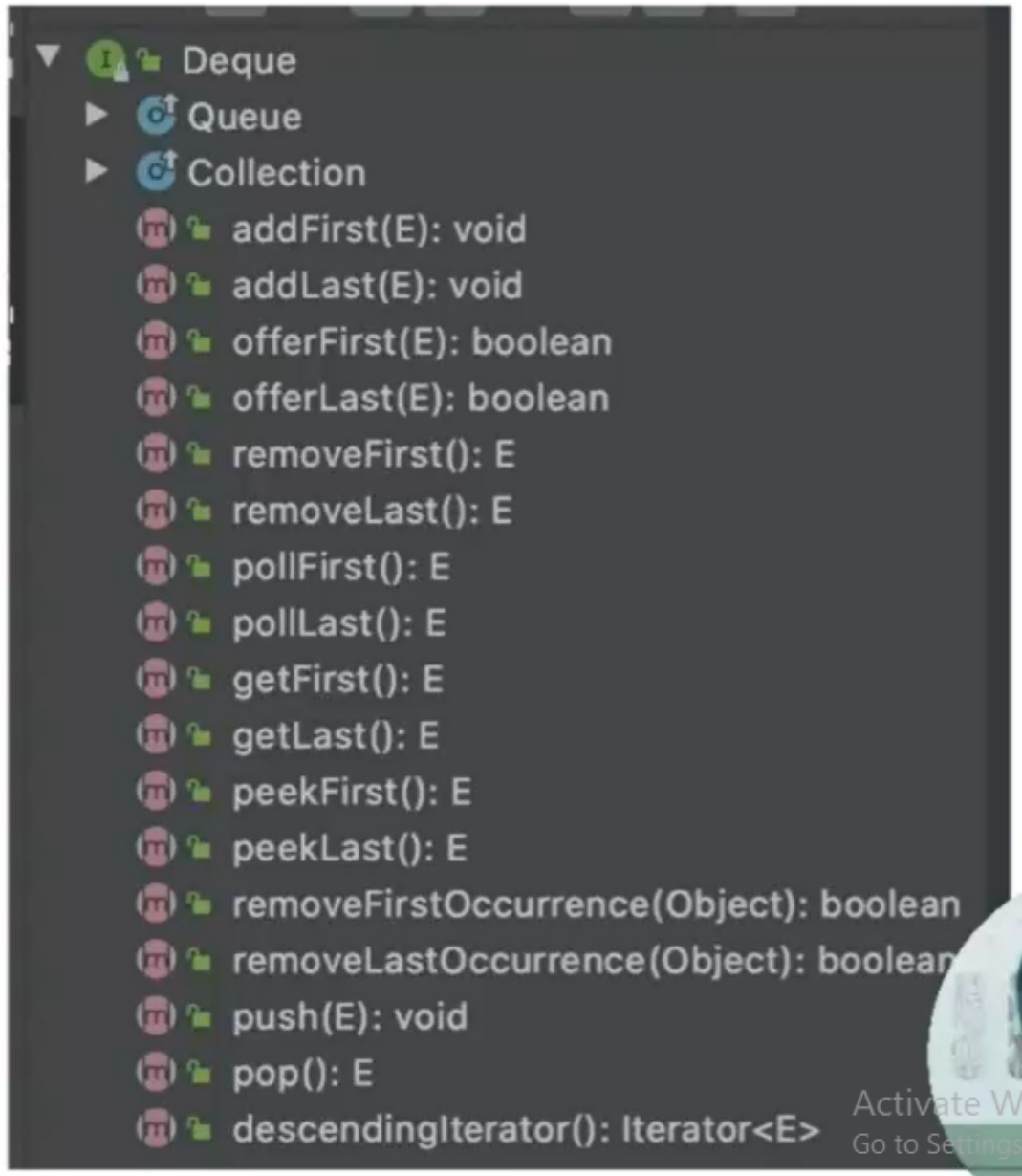
```
alta/jvs/collection via  v11.0.11 took 1s
→ java Main.java
5
```

Kenapa menghasilkan nilai “5” karena pada method .poll() akan mengambil data berdasarkan urutan prioritasnya

### ▼ Dequeue

Merupakan sebuah queue tetapi memiliki dua ujung. Method yang digunakan pada Dequeue.


#### ▼ Method pada Dequeue



#### ▼ Contoh Dequeue:

```
public static void main(String[] args) {  
    Deque<String> bulan = new LinkedList<>();  
    bulan.offerLast("1");  
    bulan.offerLast("2");  
    bulan.offerLast("3");  
    System.out.println(bulan.pollLast());  
}
```

Hasil Run :

```
alta/jvs/collection via  v11.0.11 took 1s  
→ java Main.java  
3
```

Kenapa menampilkan nilai “3” karena dari kode diatas data yang terakhir adalah “3”