

Java Collections

Silver - Chapter 2 - Topic 5

**Selamat datang di Chapter 2 Topic 5
online course Back End Java dari
Binar Academy!**



Hello, everyone! 🎉

Gimana nih konsep Java pada dua topik sebelumnya? udah makin kebayang?

Pada topik baru ini kita bakal move on ke materi yang bakal menjelajahi tentang **Java Collections**.

Apakah makna collections seperti kata koleksi yang kita kira? Yuk, langsung aja kita kepoin~



Dari sesi ini, harapannya kamu bisa mendapatkan beberapa hal berikut!

- Konsep Java Collection
- Jenis Java Collection - List, Set dan Map



Spoiler dikit, Java Collections ada kaitannya nih sama object.

Untuk memahami materi as a whole kita harus tahu dulu konsep Collection pada Java.

Yuk, belajar **Collections Concepts**.



Apa sih Java Collections itu sendiri?

Java collections merupakan **framework** yang **digunakan untuk mengatur kumpulan object secara terstruktur.**

Atau secara sederhana, collection ini dipakai untuk menyimpan object dengan berbagai tipe data object.

Object-object yang disimpan di dalam suatu collection ini disebut dengan **element**.



“Emang Java Collections itu penting?”

Penting, dong! Dalam membuat sebuah aplikasi yang cukup kompleks, Java Collections membantu kita menghindari kerepotan.

“Lho, kenapa?”

Karena banyak hal yang harus dilakukan untuk membuat collection yang memenuhi kebutuhan kita. **Dengan adanya collections ini bakal bikin proses pembuatan aplikasi jadi lebih efektif dan efisien.**



“Terus, apa sih keuntungannya kalau kita pakai Java Collections?”

Keuntungannya tuh kayak gini:

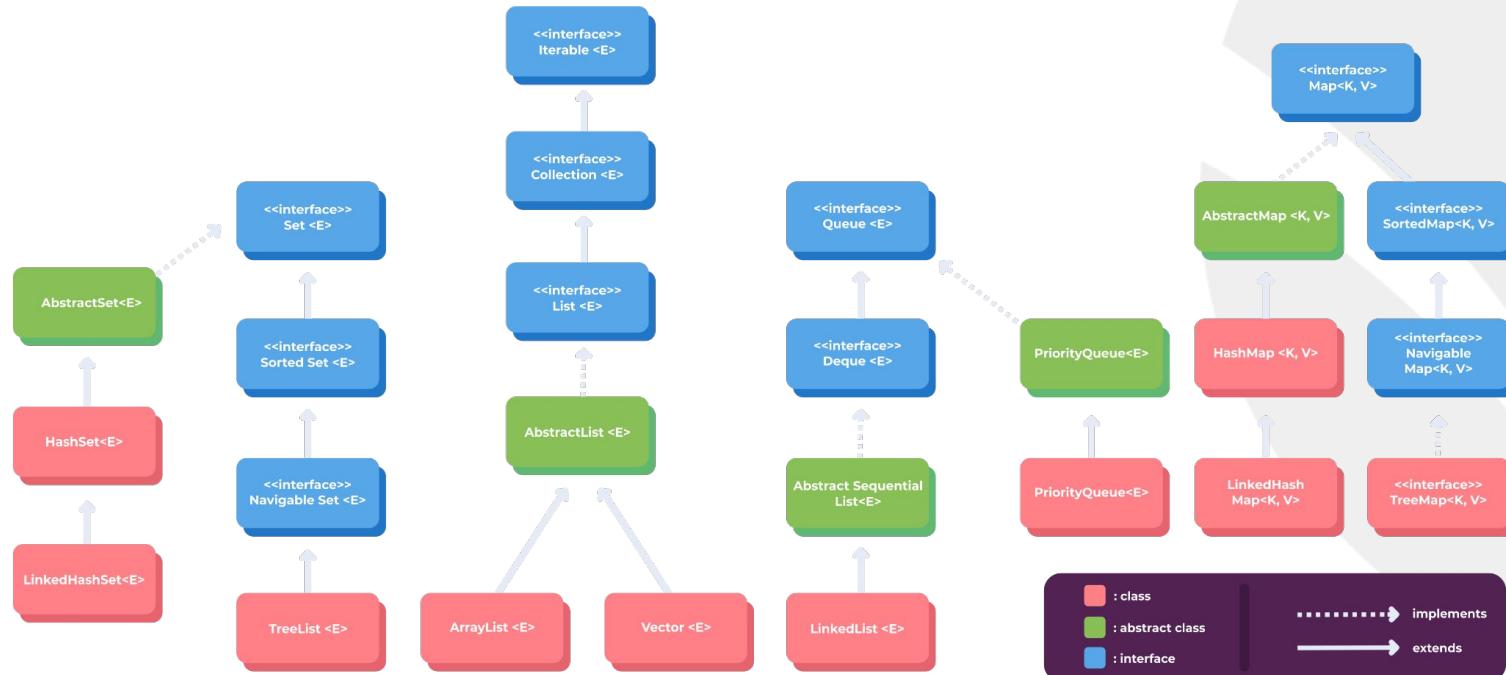
1. **Mengurangi effort** atau usaha pemrograman karena nggak perlu men-develop sesuatu yang baru untuk melakukan collection.
2. **Meningkatkan performance program** karena Java collections udah dibuat dengan kualitas yang baik.

Mantap betul~



Ini adalah gambar dari tampilan interface yang ada di Java collections.

Ada banyak banget ya~



Kalau kita lihat di gambar sebelumnya, ada dua interface paling atas berwarna biru yang terdiri dari Iterable dan Map.

Penjelasannya kayak gini, nih:

- **Iterable** adalah kumpulan suatu elemen yang punya sequence.
- **Map** adalah kumpulan suatu elemen yang dipetakan pakai key dan value.



Oh iya, pada topic ini, kita nggak bahas semua jenis collection yang ada di Java. Melainkan belajar 3 jenis collection yang paling sering dipakai aja, yaitu **List**, **Map**, dan **Set**.

Penjelasannya gimana, ya?



Ada 3 Java Collections yang sering dipakai.

Pertama dulu, nih. Kita bakal jelasin tentang **List**.



Java Collections yang pertama adalah List~

List adalah jenis collection yang mewariskan sifat dari interface iterable, yaitu punya **urutan**.

Yang dimaksud urutan disini tuh mirip kaya sistem waiting list pas kita makan di restoran.



Lebih detailnya gini, pas kamu mau makan di restoran,
eh kebetulan kursinya lagi penuh.

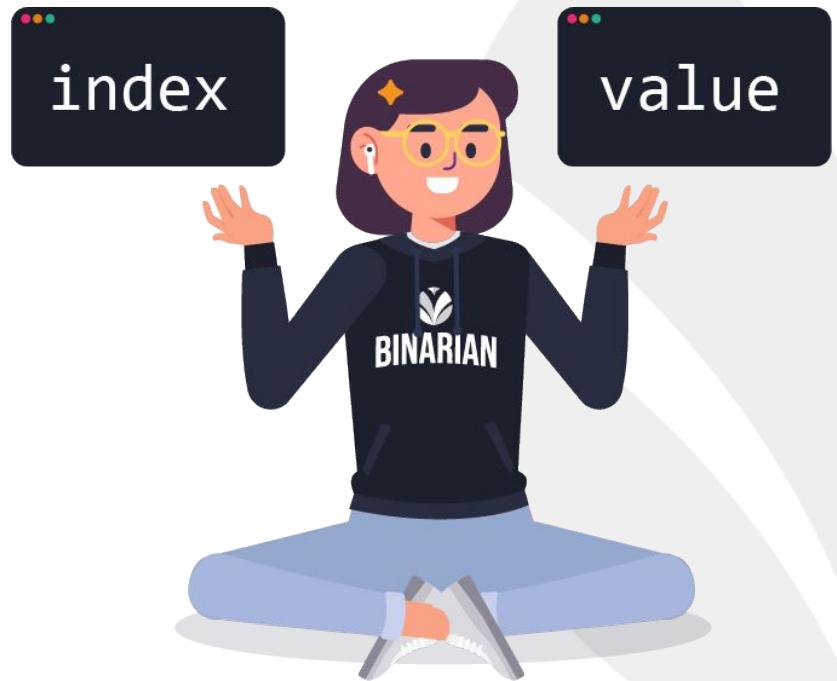
Dari situ mba mas pelayannya langsung menawarkan
kamu buat jadi waiting list restoran mereka.

Iya, mirip kayak gitu!



List punya dua elemen yang terdiri dari **index** dan **value**:

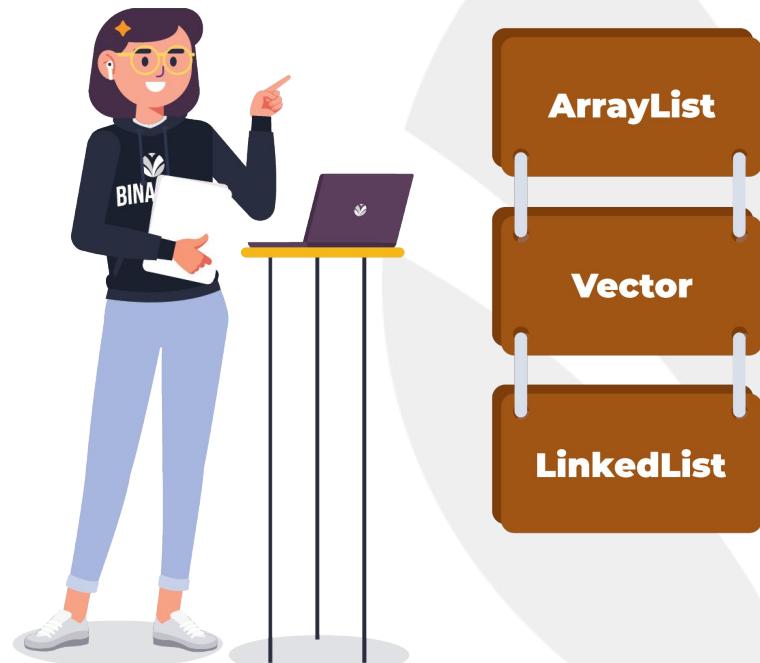
- Index dalam List disusun berdasarkan urutan kapan item dimasukkan ke dalam List.
- Value dari List bersifat nggak unik, artinya value dari sebuah index bisa di-duplicated dengan value di index lain.



Dalam Java Collections, khususnya List, ada 3 turunan yang perlu kita ketahui nih. Turunan yang berasal dari List, yaitu:

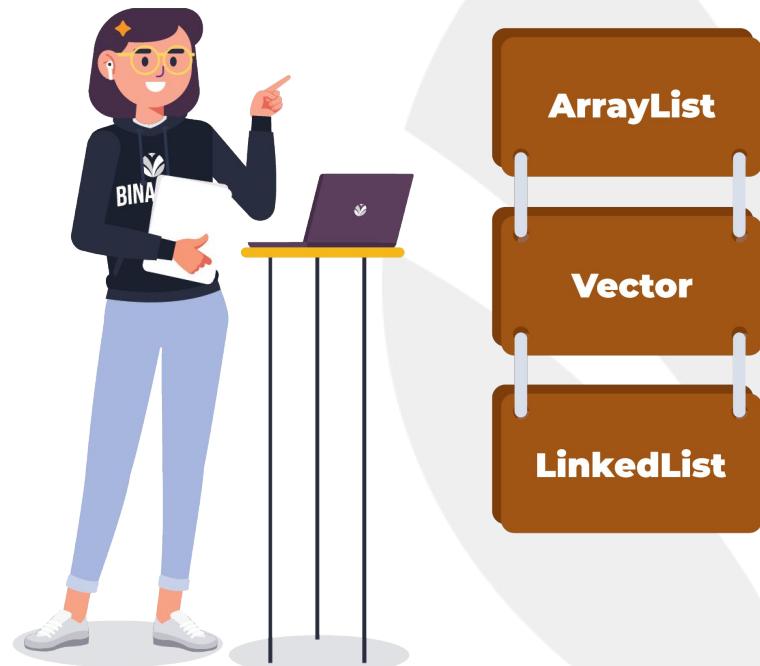
- **ArrayList**,
- **Vector**, dan
- **LinkedList**.

Walaupun ada tiga, tapi pada topik ini kita cuma pakai salah satu turunan yang sering dipakai aja, yaitu **ArrayList**.



“Apa itu ArrayList?”

ArrayList merupakan list yang mirip kayak array, tapi sizenya bisa bertambah menyesuaikan element-nya.



Ngomongin turunan udah, selanjutnya ada beberapa method yang sering dipakai pada List!

- **get(int index)**

Dipakai untuk mengambil isi dari list berdasarkan index (urutan item dimasukkan ke dalam List).

- **indexOf(Object o)**

Dipakai untuk tahu nomor index dari object yang ada dalam List.



- **add(Object o)**

Dipakai untuk menambahkan object ke dalam List.

- **add(int index, Object o);**

Dipakai untuk menambahkan object ke dalam List di index tertentu.

- **addAll(Collection<? extends E> c)**

Dipakai untuk menambahkan object dengan parameter dari collection lain. Collection bisa berupa list ataupun set. Penambahan collection ini diletakkan mulai dari index yang terakhir, ya.



- **set(int index, Object o)**

Dipakai untuk mengganti element array yang udah ada.

- **remove(int index)**

Dipakai untuk menghapus element di suatu index. Sehingga element berikutnya bakal bergeser ke index yang sebelumnya.

- **RemoveAll(Collection<? extends E> c)**

Dipakai untuk menghapus object yang ada dengan parameter dari collection lain.



Nah, kalau contoh penggunaannya bisa dilihat pada gambar di bawah ini ya~

```
● ● ●

public class ListBarang {
    public static void main(String[] args) {
        List<String> list = new ArrayList<String>();
        list.add("kursi");
        list.add("meja");
        list.add("lemari");
        list.add("ember");
        list.add("gayung");
        System.out.println("isi dari list : ");
        for(String s : list){
            System.out.println(s);
        }
    }
}
```



```
for(String s : list){  
    System.out.println(s);  
}  
}
```

Ada lagi nih contoh yang lain.

Pada gambar disamping ada dua contoh, dimana keduanya akan menghasilkan output yang sama.

Walaupun output sama, contoh yang atas lebih umum dipakai pada Collection.

Ekivalen atau sepadan dengan



```
for(int i=0; i< list.size(); i++) {  
    System.out.println(list.get(i));  
}
```

Setelah List, selanjutnya kita bakal bahas Java Collections yang kedua, yaitu Set.

Biar makin satset juga pemahamannya, yuk kita langsung bahas!



Set itu apa sih?

Set adalah salah satu turunan dari Collections yang **nggak kenal sama yang namanya index.**

Dikarenakan nggak kenal, setiap element dari sebuah Set harus unik atau nggak boleh duplicated.



Element yang ada di Set itu **nggak punya urutan** alias bisa ngacak kayak kita pas lagi isi teka teki silang. Iya, kayak gitu.

Class turunan yang mengimplementasi Set ada tiga, yaitu HashSet, LinkedHashSet, dan TreeSet.



Lalu apa bedanya HashSet, LinkedHashSet, dan TreeSet?

Penasaran, kan? ini dia perbedaannya!

- **HashSet** merupakan Set yang nggak boleh ada duplikasi dan nggak punya urutan.
- **LinkedHashSet** merupakan Set dengan urutan berdasarkan urutan penambahan element.
- **TreeSet** merupakan Set yang bisa melakukan sorting berdasarkan valuenya.



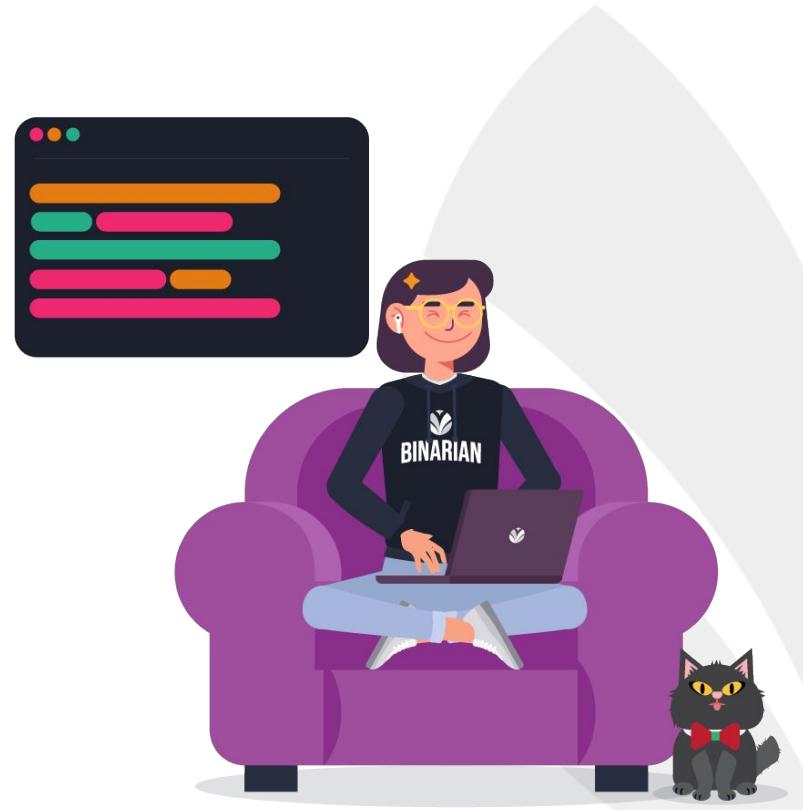
Kita move on ke method. Berikut adalah beberapa method yang sering dipakai pada Set. Simak baik-baik, ya~

- **add(Object o)**

Dipakai buat menambahkan object ke dalam Set.

- **addAll(Collection<? extends E> c)**

Dipakai buat menambahkan object dengan parameter dari collection lain. Collection bisa berupa list ataupun set. Penambahan ini diletakkan mulai dari index yang terakhir, ya.



- **remove(Object o)**

Dipakai untuk menghapus satu element.

- **RemoveAll(Collection<? extends E> c)**

Dipakai untuk menghapus object yang ada dengan parameter dari collection lain.



Biar lebih paham, ini ya gengs contoh dari Set~

```
Set<String> names = new HashSet<>();
names.add("Tom");
names.add("Mary");
names.add("Peter");
names.add("Alice");

Iterator<String> iterator = names.iterator();

while (iterator.hasNext()) {
    String name = iterator.next();
    System.out.println(name);
}
```

Konsep Java Collections ✓

Java Collections - List ✓

Java Collections - Set ✓

Biar materinya jadi lengkap, sekarang kita
bakal masuk ke materi terakhir, yaitu Map.
Let's gow~



Apa itu Map?

Map disini bukan kayak lagunya Maroon 5 ya~

Map adalah **bentuk struktur data yang berpasangan antara key dan value**. Key dan value yang dimaksud merupakan sebuah object.

Key ini harus bersifat unik karena implementasinya pakai Set.



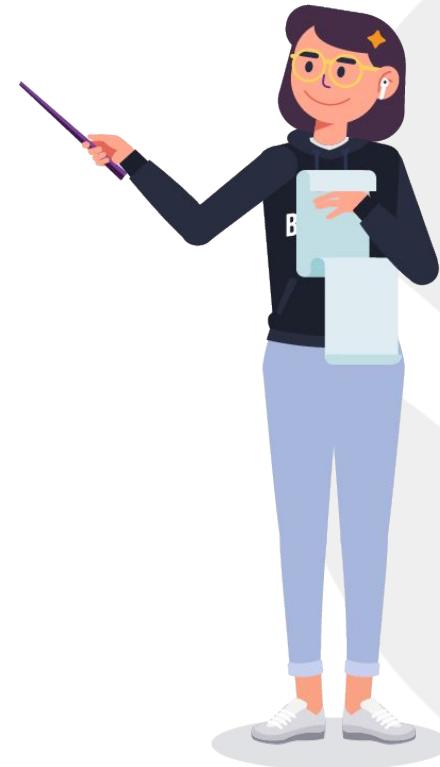
Selain itu, **map nggak punya index** sehingga fungsi dari key adalah sebagai pengganti index. Sehingga untuk melakukan pengambilan value dari suatu element, kita butuh sebuah key.

Oh iya, walaupun key-nya nggak boleh duplicated, value boleh ter-duplicated.



Mirip dengan Set, Map juga punya beberapa turunan, yaitu HashMap, LinkedHashMap, dan TreeMap!

- **HashMap** adalah map turunan yang nggak punya urutan dan memperbolehkan null.
- **LinkedHashMap** adalah turunan dari sebuah HashMap yang punya urutan berdasarkan kapan sebuah element ditambahkan.
- **TreeMap** adalah Map yang nggak memperbolehkan adanya null karena di dalam TreeMap, Key atau Value bisa dilakukan sorting.



Berikut Method yang umum untuk digunakan pada suatu Map! Lumayan banyak nih gengs, jadi simak baik-baik, ya~

- **put(Object key, Object value)**

Dipakai untuk menambahkan element.

- **remove (Object key)**

Dipakai untuk menghapus element.

- **get (Object key)**

Dipakai untuk mendapatkan value dari suatu element.



- **keySet()**

Dipakai untuk mendapatkan seluruh key dari semua element di suatu map dalam bentuk sebuah set.

- **values()**

Dipakai untuk mendapatkan semua value dari semua element dalam bentuk sebuah collection.

- **entrySet()**

Dipakai untuk mendapatkan pasangan key dan value dalam bentuk sebuah set.



- **replace(K key, V value)**

Dipakai buat mengganti value dari sebuah element.

- **size()**

Dipakai buat mendapatkan size() dari sebuah map.



- **addAll(Map<K, V> m)**

Dipakai untuk menambahkan element berdasarkan element-element yang punya map lain.

- **clear()**

Dipakai untuk mengosongkan suatu map.



Berikut adalah contoh perulangan dengan iterator!

```
● ● ●  
Map<String, String> mapCountryCodes = new HashMap<>();  
  
mapCountryCodes.put("1", "USA");  
mapCountryCodes.put("44", "United Kingdom");  
mapCountryCodes.put("33", "France");  
mapCountryCodes.put("81", "Japan");  
  
Set<String> setCodes = mapCountryCodes.keySet();  
Iterator<String> iterator = setCodes.iterator();  
  
while (iterator.hasNext()) {  
    String code = iterator.next();  
    String country = mapCountryCodes.get(code);  
  
    System.out.println(code + " => " + country);  
}
```

Kalau yang ini contoh perulangan **tanpa** iterator~

```
● ● ●

Map<Integer, String> mapHttpErrors = new
HashMap<>();

mapHttpErrors.put(200, "OK");
mapHttpErrors.put(303, "See Other");
mapHttpErrors.put(404, "Not Found");
mapHttpErrors.put(500, "Internal Server
Error");

for(Integer i : map mapHttpErrors.keySet()
{
    System.out.println(mapHttpErrors.get(i));
}
```

Ada lagi nih contoh perulangan tanpa iterator~

```
● ● ●

Map<String, String> mapCountryCodes = new HashMap<>();

mapCountryCodes.put("1", "USA");
mapCountryCodes.put("44", "United Kingdom");
mapCountryCodes.put("33", "France");
mapCountryCodes.put("81", "Japan");

Set<String> setCodes = mapCountryCodes.keySet();
Iterator<String> iterator = setCodes.iterator();

while (iterator.hasNext()) {
    String code = iterator.next();
    String country = mapCountryCodes.get(code);

    System.out.println(code + " => " + country);
}
```

Yang terakhir nih!

Contoh perulangan untuk mendapatkan key dan value dari sebuah method.

```
Set<Map.Entry<String, String>> entries
= mapCountryCodes.entrySet();

for (Map.Entry<String, String> entry :
entries) {
    String code = entry.getKey();
    String country = entry.getValue();

    System.out.println(code + " => " +
country);
}
```



Coba sini Sabrina mau denger suaranya yang udah belajar Java Collections 😊

Nah Sabrina mau tanya nih, kira-kira kenapa ya Java itu perlu menyimpan object dengan berbagai tipe data object? Apakah bisa jika disimpan dengan tidak berdasarkan tipe data objectnya?

Saatnya
Quiz

1

Berikut pernyataan yang salah mengenai list, yaitu....

- A. List menyimpan suatu element dalam bentuk key and value
- B. List memiliki index
- C. List menyimpan elemen dengan teratur

1

- A.** List menyimpan suatu element dalam bentuk key and value

Key dan value digunakan pada map

2

Berikut pernyataan yang salah mengenai set, yaitu....

- A. Set memperbolehkan elementnya bernilai null
- B. Set memperbolehkan elementnya untuk terduplicasi
- C. Set merupakan turunan dari collections

2

**B. Set memperbolehkan elementnya
untuk terduplicasi**

Pada Set, element-elementnya
harus bersifat unik

3

Berikut method yang digunakan suatu map untuk menambahkan satu object adalah....

- A. clear()
- B. get(Object object)
- C. put(K key, V Value)

3

C. **put(K key, V Value)**

Method put akan menambahkan elemen dengan parameter key dan value

4

Key dari sebuah map bisa dijadikan sebagai collection yang berbentuk....

- A. ArrayList
- B. TreeMap
- C. KeySet

4

C. KeySet

Karena key bersifat unik, key pada map bisa dihimpun dalam bentuk set

5

Method yang digunakan untuk menampilkan sebuah set dari pasangan key dan value adalah....

- A. clear()
- B. entrySet()
- C. keyset()

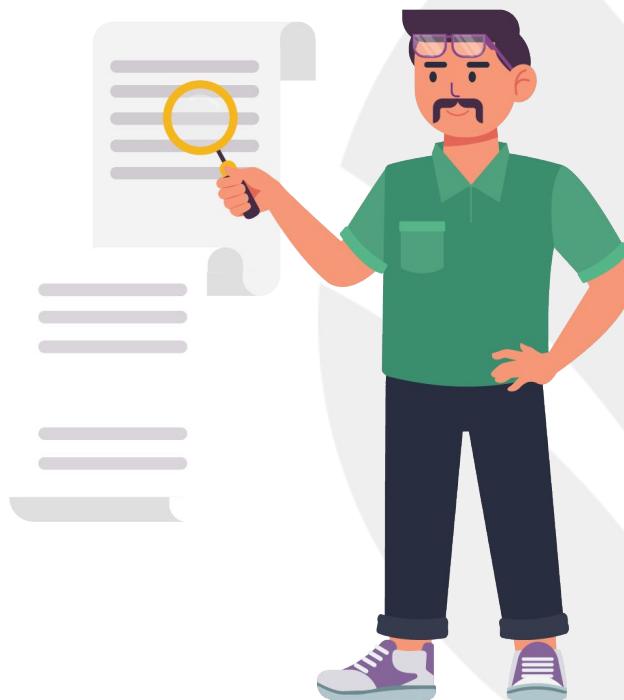
5

B. entrySet()

entrySet merupakan sebuah set yang mengandung object untuk menyimpan key dan value dari satu element pada map

Referensi

1. [Java Collections tutorial and example](#)



Nah, selesai sudah pembahasan kita di Chapter 2 Topic 5 ini.

Selanjutnya, kita bakal bahas tentang Java Standard Class.

Penasaran kayak gimana? Cus langsung ke topik selanjutnya~

