Java Collections Framework

Prepared by Viska Mutiawani

viska@unsyiah.ac.id

Subtopik Java Collections Framework

- What is and why collections?
- Interfaces & Implementations in Collection Framework
- Core Collection interfaces
  - Set Interface & implementations
  - List Interface & implementations
  - Map Interface & implementations
  - Queue Interface & implementations
- Algorithms
- Custom implementations
- Interoperability

viska@unsyiah.ac.id

What is and why collections?

viska@unsyiah.ac.id

What is a Collection?

- Nama lainnya container
- Objek yang mengandung objek-objek lain.
  - Bayangkan seperti tas yang berisi berbagai macam barang
- Collection digunakan untuk menyimpan, mendapatkan, memanipulasi dan mengkomunikasikan data kumpulan.
- Collection merepresentasikan item-item data yang membentuk grup, contoh:
  - Koleksi kartu
  - Mail folder
  - Telephone directory

viska@unsyiah.ac.id

What is Collection Framework?

- Arsitektur yang yang merepresentasikan dan memanipulasi collection.
- Collection framework terdiri dari:
  - Collection interfaces
  - Collection implementations
    - Implementasi dari collection interfaces
    - JDK menyediakan implementasi tapi kita dapat membuat versi sendiri

viska@unsyiah.ac.id

Kegunaan Collection Framework

- Mempermudah membuat program
  - Implementasi telah ada dalam JDK
- Mempercepat kecepatan program dan kualitasnya
  - Implementasi yang ada dalam JDK sudah teroptimisasi
- Membolehkan interoperability antara API yang tak berkait
  - Collection interface adalah bahasa umum yang digunakan API untuk saling berkirim collection
- Membantu perkembangan *software reuse*
  - Setiap struktur data yang menggunakan standar collection interface secara alami pasti reusable

viska@unsyiah.ac.id

Interfaces & Implementations in Collection Framework

viska@unsyiah.ac.id

Collection Interfaces

- Merupakan tipe data abstrak yang mewakili collection
- Interface membolehkan collection untuk dimanipulasi secara mandiri berdasarkan detil implementasinya
  - Karena ada perilaku polymorphic
- Interface membentuk hirarki
  - Pilih sesuai yang diperlukan

viska@unsyiah.ac.id

Collection Interfaces & Implementations

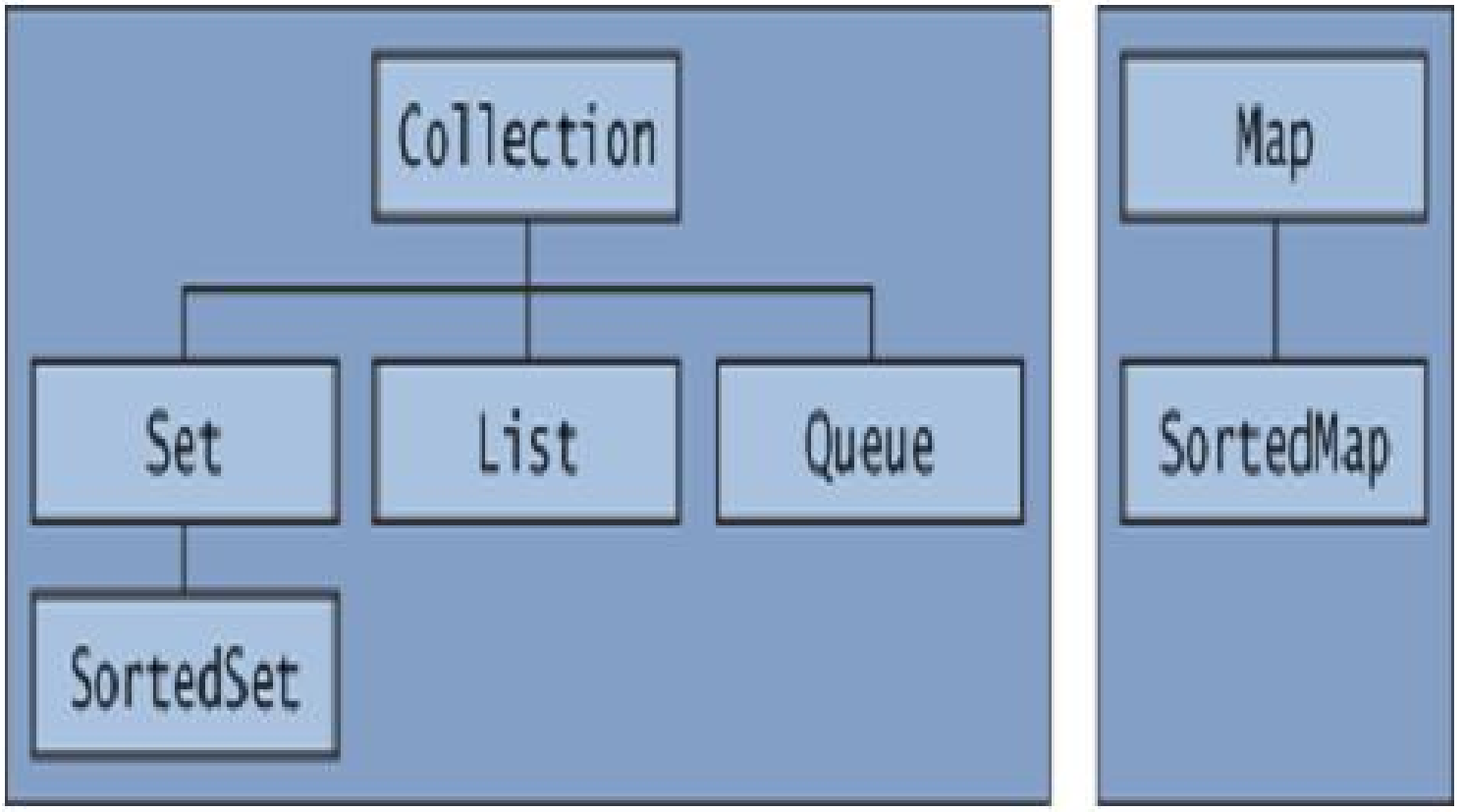
Interface	Hash Table	Resizable Array	Balanced Tree	Linked List	Hash Table + Linked List
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Deque		ArrayDeque		LinkedList	
Map	HashMap		TreeMap		LinkedHashMap

viska@unsyiah.ac.id

Core Collection Interfaces

viska@unsyiah.ac.id

Hirarki Core Collection Interfaces



viska@unsyiah.ac.id

Core Collection Interfaces

- Core collection interfaces adalah fondasi dari Java Collections Framework
- Core collection interface membentuk hirarki inheritance (pewarisan)
  - Anda juga dapat membuat collection interface yang baru dari yang telah ada

viska@unsyiah.ac.id

Java Interface “Collection”

viska@unsyiah.ac.id

Java Interface “Collection”

- Merupakan induk pada hirarki collection
  - Setiap objek collection pastilah bertipe interface “Collection”
- Digunakan untuk saling berkirim objek collection dan memanipulasinya ketika generalitas diperlukan
  - Gunakan interface “Colletion” sebagai tipe data
- JDK tidak menyediakan implementasi pada interface “Collection”
  - Namun implementasi ada pada subinterface seperti Set dan List

Struktur Interface "Collection"

```
public interface Collection<E> extends Iterable<E> {
  // Basic operations
  int size();
  boolean isEmpty();
  boolean contains(Object element);
  boolean add(E element);
  boolean remove(Object element);
  Iterator<E> iterator();

// Bulk operations
  boolean containsAll(Collection<?> c);
  boolean addAll(Collection<? extends E> c);
  boolean removeAll(Collection<?> c);
  boolean retainAll(Collection<?> c);
  void clear();

// Array operations
  Object[] toArray();
  <T> T[] toArray(T[] a);

}
```


viska@unsyiah.ac.id

Contoh Penggunaan “Collection”

```
// Create a ArrayList collection object instance and
// assign it to Collection type.
Collection c1 = new ArrayList();

// Use methods of Collection interface.
//
// Polymorphic behavior is expected. For example,
// the add() implementation of ArrayList class will
// be invoked. And depending on the implementation,
// duplication could be allowed or not allowed.

boolean b1 = c1.isEmpty();
boolean b2 = c1.add(new Integer(1));
```

Method add() dan remove() pada interface “Collection”

- Method add()
  - Menambah unsur ke dalam collection
  - Method add() pada interface Set memiliki aturan “no duplicate”
  - Mengembalikan nilai true jika unsur berhasil ditambah
- Method remove()
  - Menghapus satu unsur yang dispesifikkan dari collection, jika wujud
  - Mengembalikan nilai true jika unsur berhasil dihapus

viska@unsyiah.ac.id

Dua Cara Traversing/Menjejaki Collection

- Looping for
  - Contoh:

```
for (Object o: collectionObject){
     // Do whatever you need to do with a member object.
}
```

- Iterator
  - Iterator merupakan objek yang memungkinkan untuk menjejaki setiap unsur pada collection satu per satu dan menghapus unsur bila perlu

Interface Iterator dan penggunaannya

```
public interface Iterator {
      boolean hasNext();
      Object next();
      void remove();
}
```

- Method hasNext()
  - Mengembalikan nilai true jika masih ada unsur dalam iteration
- Method next()
  - Mengembalikan unsur berikutnya pada iteration
- Method remove()
  - Menghapus unsur terakhir yang dikembalikan oleh iterator
  - Hanya dapat sekali dipanggil sesudah next()

viska@unsyiah.ac.id

Bulk Operations

- containsAll() – kembalikan true jika target collection mengandung collection yang dicari
- addAll() – tambah semua unsur pada suatu collection ke dalam target collection
- removeAll() – menghapus semua unsur dalam target collection yang juga terkandung dalam collection
- retainAll() – menghapus semua unsur dalam target collection yang tidak terkandung dalam collection
- clear() – menghapus semua unsur dalam collection

Operasi Array

- Method toArray() disediakan sebagai jembatan antara collection dan API versi lama yang memerlukan array sebagai input
- Operasi array inilah yang memungkinkan isi collection dirubah menjadi array

Contoh Operasi Array

- Disimpan dalam Object:
  - `Object[] a = myCollection.toArray();`
- Disimpan dalam tipe data yang sudah diketahui sebelumnya:
  - `String[] a = myCollection.toArray();`

viska@unsyiah.ac.id

Interface “Set”

- Mewakili collection yang tidak boleh ada unsur duplikasi
- Contoh penggunaan:
  - Permainan kartu
  - Kursus pada jadwal
  - Proses pada mesin
- Jadi implementasi pada interface Set ada batasan bahwa unsur-unsur tidak boleh ada yang duplikasi

viska@unsyiah.ac.id

Interface “Set”

```
public interface Set<E> extends Collection<E> {
// Basic operations
  int size();
  boolean isEmpty();
  boolean contains(Object element);
  boolean add(E element); //optional
  boolean remove(Object element); //optional
  Iterator<E> iterator();
// Bulk operations
  boolean containsAll(Collection<?> c);
  boolean addAll(Collection<? extends E> c); //optional
  boolean removeAll(Collection<?> c); //optional
  boolean retainAll(Collection<?> c); //optional
  void clear(); //optional
// Array Operations
  Object[] toArray();
  <T> T[] toArray(T[] a);
}
```

Operasi “equals”

- Untuk memastikan tidak ada duplikasi, implementasi Set memanfaatkan equals dan hashCode.
- Dua unsur Set dikatakan sama/equal jika keduanya mengandung unsur yang sama

viska@unsyiah.ac.id

Interface “SortedSet”

- Merupakan Set yang menjaga unsur-unsurnya dalam urutan menaik.
- Penggunaannya seperti pada rangkaian kata, daftar anggota

Implementasi Interface “Set”

- Class HashSet
- Class TreeSet
- Class LinkedHashSet

General-purpose Implementations

Interfaces	Implementations				
	Hash table	Resizable array	Tree	Linked list	Hash table + Linked list
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Queue					
Map	HashMap		TreeMap		LinkedHashMap

viska@unsyiah.ac.id

Class HashSet

- HashSet lebih cepat dari TreeSet tapi HashSet tidak terurut, sedangkan TreeSet terurut
  - HashSet: O(1)
  - TreeSet: O(log n)
- Tips menggunakan HashSet:
  - Jika memilih initial capacity terlalu besar bisa mubazir ruang penyimpanan
  - Jika memilih initial capacity terlalu kecil bisa menghabiskan waktu dalam proses copy data jika ukurannya perlu diperbesar

viska@unsyiah.ac.id

Class TreeSet

- TreeSet menggunakan interface SortedSet
- Sehingga unsur yang ditambah akan disusun menjadi terurut
- Digunakan jika anda perlu membuat set yang nilainya harus terurut

viska@unsyiah.ac.id

LinkedHashSet

- Merupakan hash table yang memanfaatkan konsep linked list
- Unsur yang masuk sesuai dengan *insertion-ordered*
- Hampir secepat HashSet namun urutannya tidak aneh seperti HashSet, dan tidak memerlukan proses pengurutan seperti pada TreeSet

viska@unsyiah.ac.id

Contoh HashSet, TreeSet, LinkedHashSet

```
public class MyOwnUtilityClass {

    // Note that the first parameter type is set to
    // Set interface not a particular implementation
    // class such as HashSet.  This makes the caller of
    // this method to pass instances of different
    // implementations of Set interface while
    // this function picks up polymorphic behavior
    // depending on the actual implementation type
    // of the object instance passed.

    public static void checkDuplicate(Set s, String[] args){
        for (int i=0; i<args.length; i++)
            if (!s.add(args[i]))
                System.out.println("Duplicate detected: "+args[i]);

        System.out.println(s.size()+" distinct words detected: "+s);
    }
}
```


viska@unsyiah.ac.id

Contoh HashSet, TreeSet, LinkedHashSet

```
public class Main {

    public static void main(String[] args) {

        Set s = new HashSet();      // Order is not guaranteed
        MyOwnUtilityClass.checkDuplicate(s, args);

        s = new TreeSet();          // Order according to values
        MyOwnUtilityClass.checkDuplicate(s, args);

        s = new LinkedHashSet(); // Order according to insertion
        MyOwnUtilityClass.checkDuplicate(s, args);
    }
}
```

Contoh HashSet, TreeSet, LinkedHashSet

- Andaikan kita memberikan input dengan urutan seperti berikut:

  2
  3
  4
  1
  2

- Apa yang akan tercetak?

viska@unsyiah.ac.id

List Interface & implementations

viska@unsyiah.ac.id

Interface “List”

- Merupakan interface berupa rentetan/rangkaian
- Unsur pertama masuk akan tetap pada urutan pertama, urutan kedua masuk akan tetap pada urutan kedua, dst
- List membolehkan unsur yang sama/duplikat

Beberapa tambahan fungsi pada List

- Akses sesuai dengan indeks
- Pencarian suatu unsur dalam list dan mengembalikan posisi indeksnya
- Iteration – memanfaatkan iterator untuk traversing
- Operasi dapat dibatasi dalam suatu range/jarak tertentu

Interface “List”

```
public interface List<E> extends Collection<E> {
   // Positional access
   E get(int index);
   E set(int index, E element);
   boolean add(E element);
   void add(int index, E element);
   E remove(int index);
   boolean addAll(int index,
      Collection<? extends E> c);

   // Search
   int indexOf(Object o);
   int lastIndexOf(Object o);

   // Iteration
   ListIterator<E> listIterator();
   ListIterator<E> listIterator(int index);

   // Range-view
   List<E> subList(int from, int to);
}
```


viska@unsyiah.ac.id

Implementasi Interface “List”

General-purpose Implementations

Interfaces	Implementations				
	Hash table	Resizable array	Tree	Linked list	Hash table + Linked list
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Queue					
Map	HashMap		TreeMap		LinkedHashMap

viska@unsyiah.ac.id

Implementasi Interface “List”

- Class ArrayList
  - Akses dengan kecepatan konstan O(1)
  - Cepat
  - Mirip seperti class Vector, bedanya ArrayList unsynchronized
- Class LinkedList
  - Menggunakan konsep Linked list
- Perbedaan keduanya dapat dilihat pada situs ini

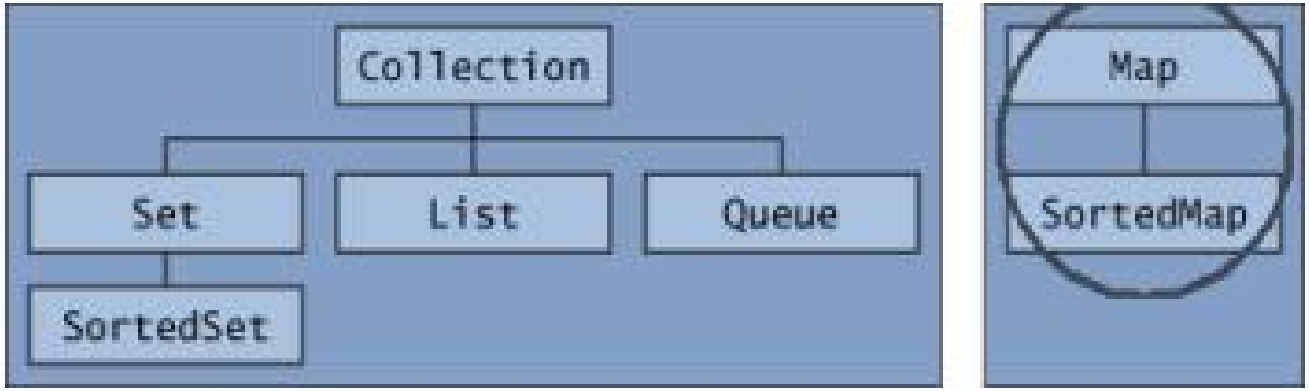
viska@unsyiah.ac.id

Map Interface & Implementations

viska@unsyiah.ac.id

Interface “Map”

- Menangani pasangan kunci (key) dan nilai (value)
- Map tidak boleh mengandung kunci yang duplikat



viska@unsyiah.ac.id

Interface “Map”

```
public interface Map<K,V> {
    // Basic operations
    V put(K key, V value);
    V get(Object key);
    V remove(Object key);
    boolean containsKey(Object key);
    boolean containsValue(Object value);
    int size();
    boolean isEmpty();

    // Bulk operations
    void putAll(Map<? extends K, ? extends V> m);
    void clear();

    // Collection Views
    public Set<K> keySet();
    public Collection<V> values();
    public Set<Map.Entry<K,V>> entrySet();

    // Interface for entrySet elements
    public interface Entry {
        K getKey();
        V getValue();
        V setValue(V value);
```


viska@unsyiah.ac.id

Interface “SortedMap”

- Map yang menjaga kunci agar tetap terurut menaik
  - Mirip seperti SortedSet
- Dapat digunakan untuk collection berupa pasangan yang harus terurut sepert:
  - Kamus
  - Buku telpon

viska@unsyiah.ac.id

Implementation Interface “Map”

- Class HashMap
  - Pakai ini jika anda mementingkan kecepatan tanpa mempedulikan urutan iterasi
- Class TreeMap
  - Pakai ini jika perlu implementasi dari SortedMap atau iteration yang terurut berdasarkan kunci
- Class LinkedHashMap
  - Pakai ini jika ingin kecepatan yang hampir sama dengan HashMap namun iteration berdasarkan insertion-order

Queue Interface & Implementations

viska@unsyiah.ac.id

Interface “Queue”

- Collection yang digunakan untuk menyimpan banyak unsur terlebih dahulu sebelum diproses
- Queue memiliki tambahan operasi insertion, extraction dan inspection
- Mengikuti konsep FIFO (first in, first out)

Implimentasi Interface “Queue”

- LinkedList mengimplement interface Queue agar menjadi FIFO
- Class PriorityQueue adalah queue yang memberi prioritas dan berbasis struktur data heap

viska@unsyiah.ac.id

Method lain

- Untuk mengosongkan collection
  - emptySet, emptyList dan emptyMap
- Method untuk sorting dan shuffling
  - Collections.sort(l);
  - Collections.shuffle(l);

viska@unsyiah.ac.id

Abstract Class

- Implementasi abstract
  - AbstractCollection
  - AbstractSet
  - AbstractList
  - AbstractSequentialList
  - AbstractMap
- Mempermudah implementasi kustomisasi
  - Mengurangi coding yang harus dibuat

viska@unsyiah.ac.id

Algorithms

viska@unsyiah.ac.id

Algorithms

- Sorting
- Shuffling
- Manipulasi data
- Searching
- Composition

viska@unsyiah.ac.id

Sorting

- Merupakan algoritma untuk mengurutkan unsur secara menaik.
- Dapat digunakan pada List
- Ada 2 bentuk:
  - Dengan menggunakan List, urut dengan method sort()
  - Dengan menggunakan List, manfaatkan Comparator, dan urut dengan method sort()

Urutan yang lazim

- Unsur dalam List telah mengimplement interface Comparable
- Contoh:
  - Jika List mengandung unsur String, akan diurut secara alfabet
  - Jika List mengandung unsur Date, akan diurut secara kronologi
  - Karena String dan Date implement Comparable sehingga memungkinkan unsurnya diurut otomatis dengan urutan lazim
- Jika sort dilakukan pada unsur yang tidak implement Comparable, Collections.sort(list) akan melempar ClassCastException

viska@unsyiah.ac.id

Contoh bentuk urutan yang pertama

```
// Set up test data
String n[] = {
    new String("John"),
    new String("Karl"),
    new String("Groucho"),
    new String("Oscar")
};

// Create a List from an array
List l = Arrays.asList(n);

// Perform the sorting operation
Collections.sort(l);
```


viska@unsyiah.ac.id

Contoh bentuk urutan yang kedua

- Fungsi perbandingan pada comparator memaksa pengurutan pada koleksi objek.
- Comparator dapat menjadi parameter pada Collections.sort atau Arrays.sort.
- Comparator dapat digunakan pada koleksi objek yang tidak memiliki urutan lazim

```
// Set up test data
ArrayList u2 = new ArrayList();
u2.add("Beautiful Day");
u2.add("Stuck In A Moment You Can't Get Out Of");
u2.add("Elevation");
u2.add("Walk On");
u2.add("Kite");
u2.add("In A Little While");
u2.add("Wild Honey");
u2.add("Peace On Earth");
u2.add("When I Look At The World");
u2.add("New York");
u2.add("Grace");

Comparator comp = Comparators.stringComparator();
Collections.sort(u2, comp);
System.out.println(u2);
```


viska@unsyiah.ac.id

Shuffling

- Algoritma shuffle merupakan kebalikan dari sort
- Bekerja dengan menghilangkan semua jejak data terurut yang wujud pada List
  - Algoritma akan menyusun List dengan urutan acak
- Dapat digunakan pada permainan keberuntungan
  - Dapat digunakan untuk men-shuffle kartu
  - Menghasilkan test case

viska@unsyiah.ac.id

Manipulasi Data Rutin

- Collections menyediakan 5 algoritma untuk melakukan manipulasi data rutin pada objek List:
  - reverse – membalik urutan unsur pada List
  - fill – menimpa setiap unsur pada List dengan nilai tertentu. Berguna untuk re-initializing.
  - copy – ada dua parameter, destinasi dan sumber, copy semua unsur dari sumber ke List destinasi. Destinasi list haruslah sepanjang List sumber, kalaupun lebih panjang maka data sisa pada List destinasi tidak akan berubah
  - swap – tukar unsur pada posisi tertentu dalam List
  - addAll – tambah sejumlah unsur tertentu dalam Collection.

viska@unsyiah.ac.id

Searching

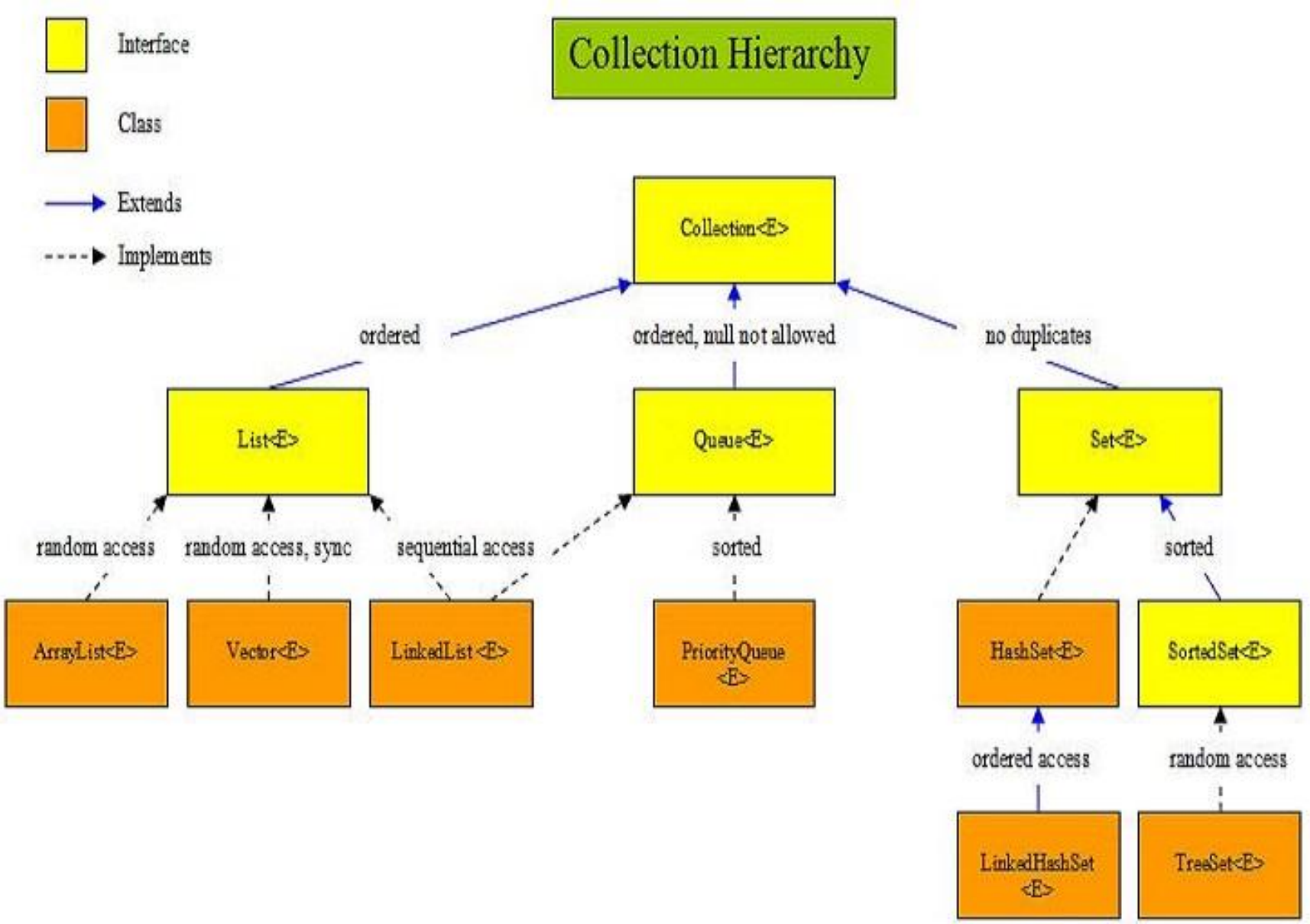
- Collections memiliki method binarySearch() untuk mencari unsur tertentu pada List yang terurut.

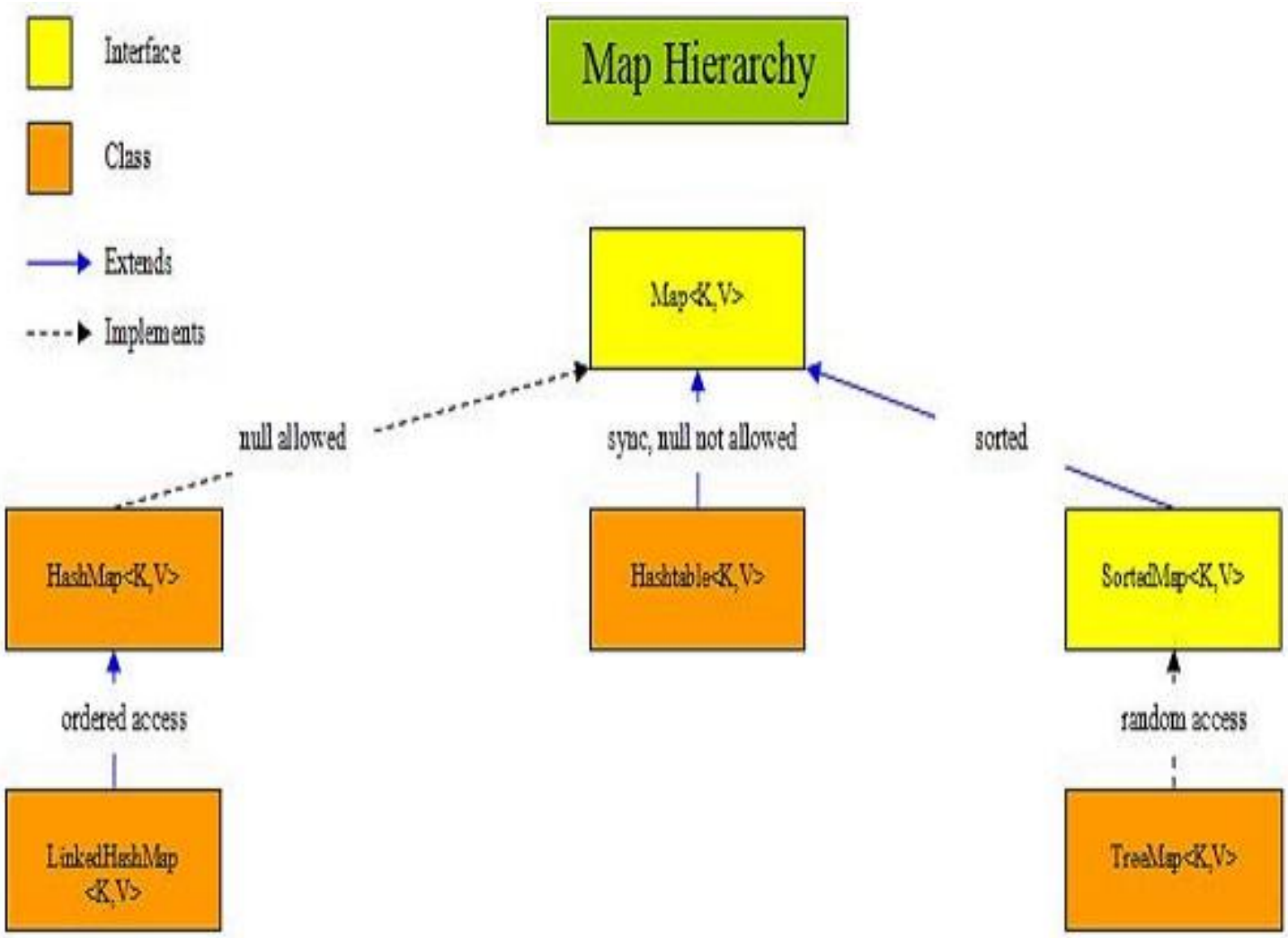
```
// Set up testing data
String name[] = {
        new String("Sang"),
        new String("Shin"),
        new String("Boston"),
        new String("Passion"),
        new String("Shin"),
};

List l = Arrays.asList(name);

int position = Collections.binarySearch(l, "Boston");
System.out.println("Position of the searched item = " + position);
```

Composition

- Collections.frequency(l) – menghitung berapa kali suatu unsur muncul di dalam collection
- Collections.disjoint(l1, l2) – menentukan apakah kedua collection disjoint, maksudnya data di dalamnya tidak ada yang sama

viska@unsyiah.ac.id


Interface
Class
Extends
Implements
Collection Hierarchy
Collection<E>
ordered
ordered, null not allowed
no duplicates
List<E>
Queue<E>
Set<E>
random access
random access, sync
sequential access
sorted
sorted
ArrayList<E>
Vector<E>
LinkedList<E>
PriorityQueue<E>
HashSet<E>
SortedSet<E>
ordered access
random access
LinkedHashSet<E>
TreeSet<E>



viska@unsyiah.ac.id

viska@unsyiah.ac.id