Java Collection Framework

جامعة طرابلس - كلية تقنية المعلومات

د. عبدالحميد الواعر

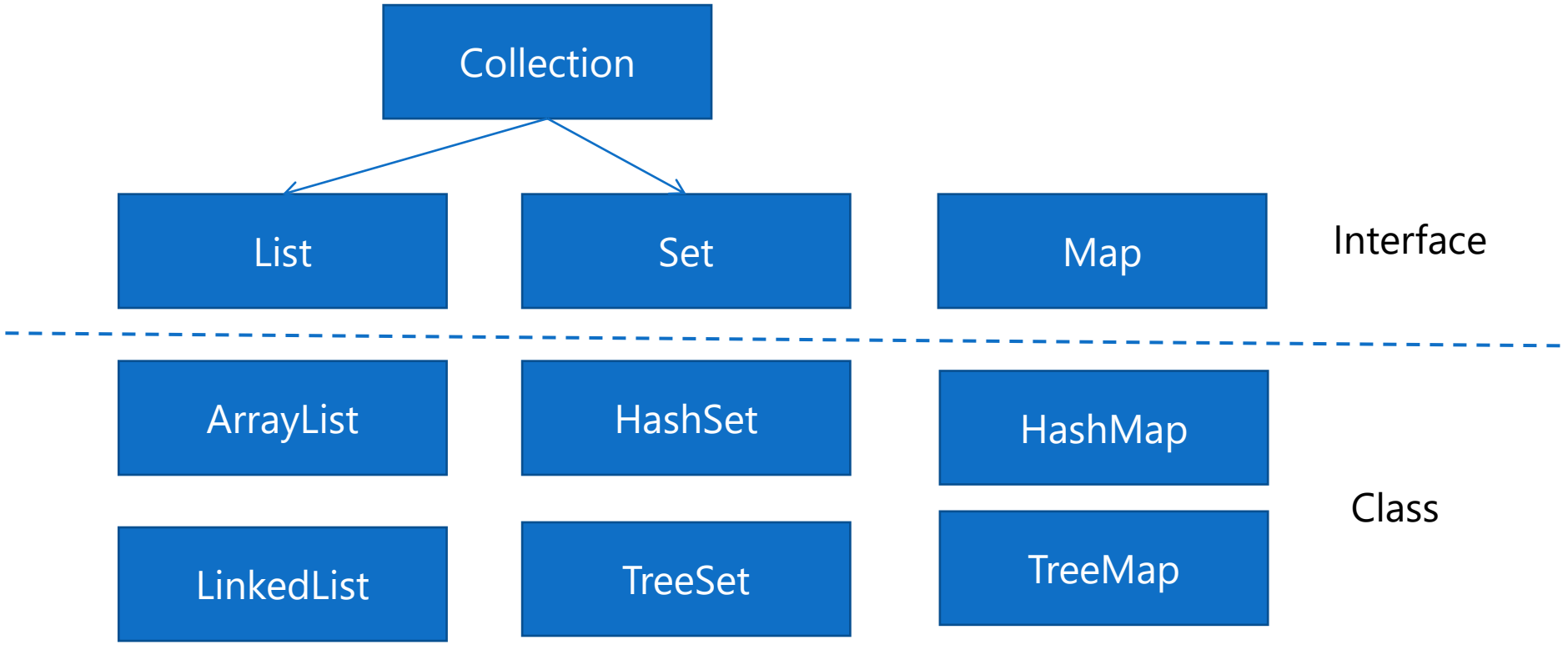
Java Collection Framework

Java Collection Framework

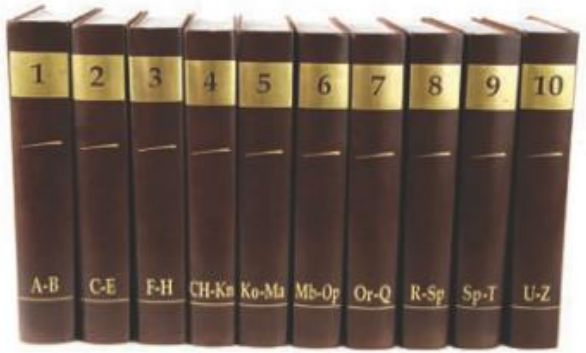
- تستخدم في تجميع مجموعة من objects في entity واحدة .
- توفر جافا collection & Map interfaces للقيام بذلك.

Java Collection Framework

- الشكل التالي يبين Java Collections Framework

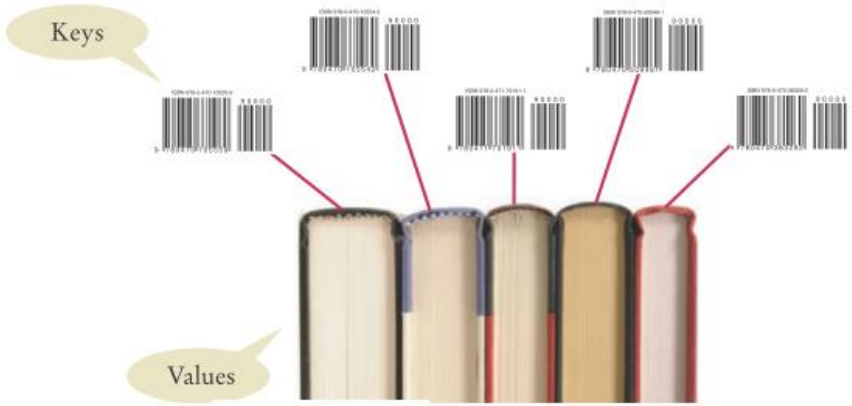


مثال:



A List of Books

A Set of Books



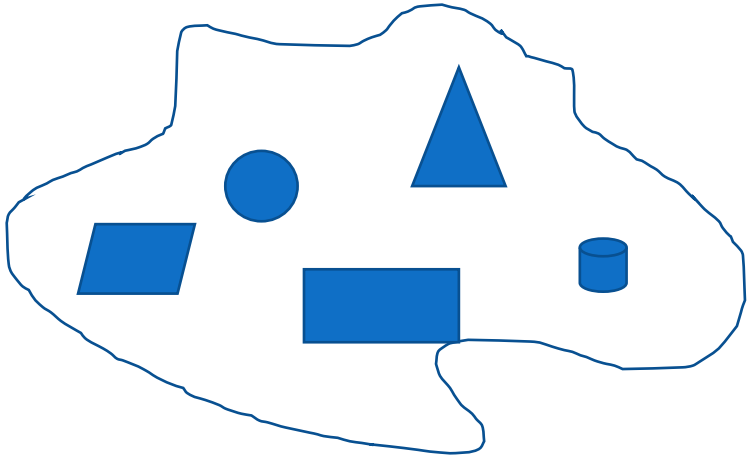
A Map from Bar Codes to Books

Collection Interface

- Collection interface يوفر مجموعة من العمليات المتنوعة حسب التالي:

1. **Informative methods:**
   - Iterator iterator()
   - Boolean isEmpty()
   - int Size()

2. **Object based methods:**
   - Boolean equals(Object o)
   - int hashCode()



Collection A

Collection Interface

3. Element based methods

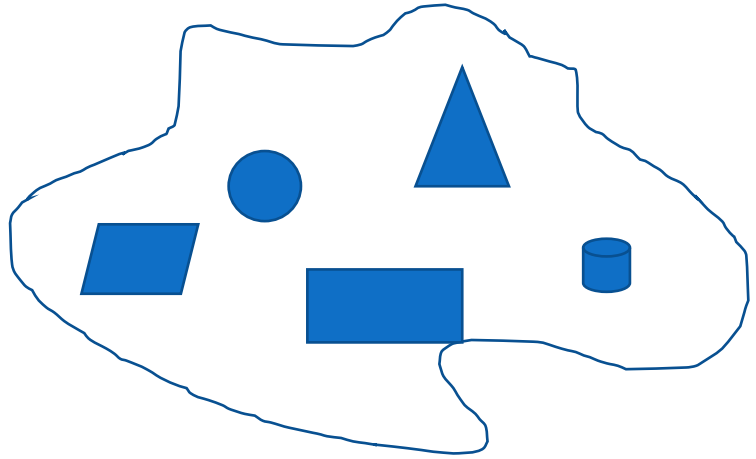
- Boolean add(Object o)
- Boolean remove(Object o)
- contains (Object o)

4. Output to Arrays methods:

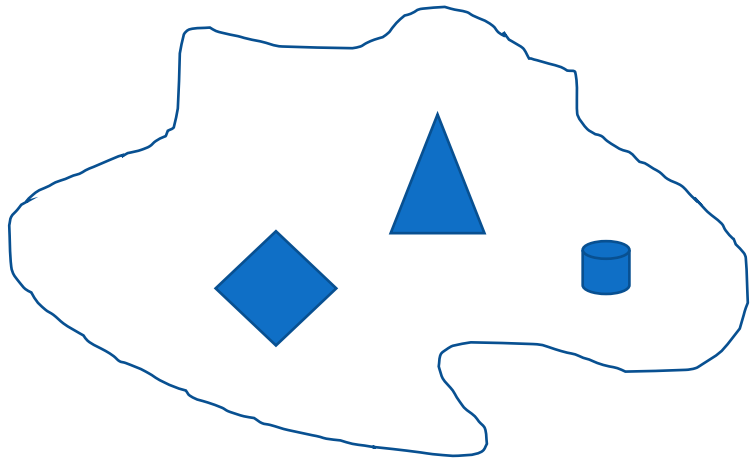
- Object[] toArray()
- Object toArray(Object[] a)

5. Collection based methods:

- Boolean addAll(Collection c)
- Boolean containsAll(Collection c)
- Void clear()
- Boolean retainAll(Collection c)



Collection A



Iterator Object

- يستخدم في الانتقال عبر عناصر collectionللقيام ببعض العمليات عليها.
- يتم الحصول على iterator objectعند أستدعاء الدالة iterator()
- يوفر iterator objectبعض methodsالمهمة للتعامل معه ومنها:

- Boolan hasNext()
- Object next()
- Void remove()

مثال:

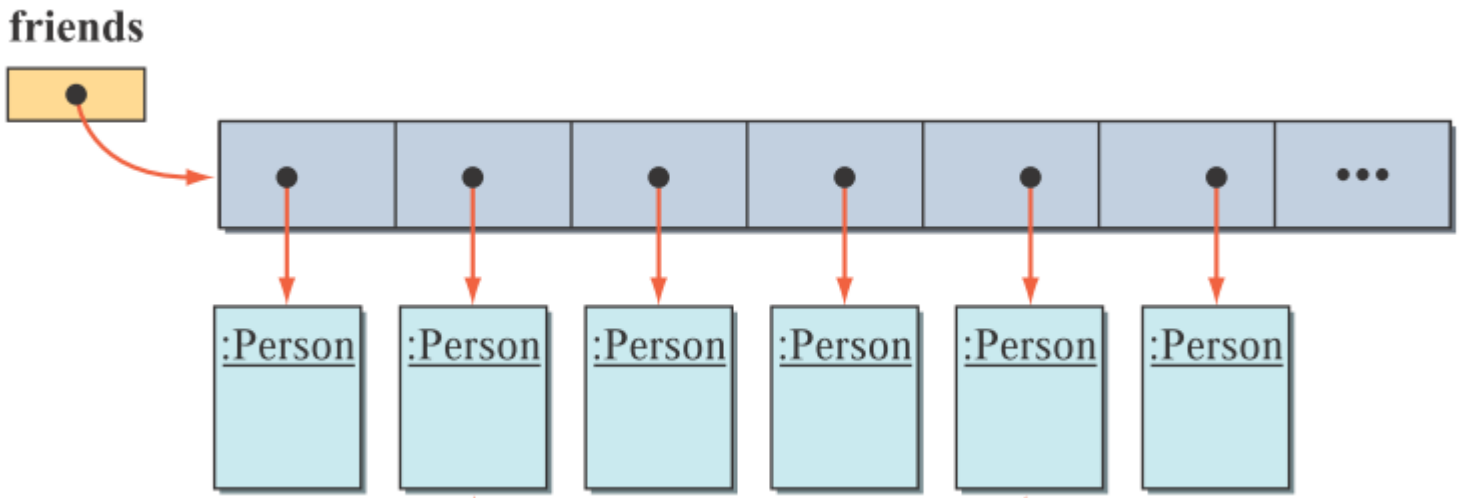
```
Collection col = ...
Iterator it = col.iterator();
while(it.hasNext()){
     Student s = (Student) it.next();
}
```

ArrayList

- ArrayListهي أحد classesالناتجة من List interfaceوهي لها الصفات التالي:
- تسمح بتكرار العناصر المخزنة بها
- كل عنصر له indexخاص به
- يتم أسترجاع العنصر بأستخدام indexالخاص به
- يتم ترتيب العناصر حسب أولوية أدخالها

مثال:

```
List<Person> friends;
...
friends = new ArrayList<Person>( );
```



ArrayList constructors& methods

`ArrayList()`
Constructs an empty list with an initial capacity of ten.

`ArrayList(Collection<? extends E> c)`
Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

`ArrayList(int initialCapacity)`
Constructs an empty list with the specified initial capacity.

methods

E	**get**(int index) Returns the element at the specified position in this list.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
int	**lastIndexOf**(Object o) Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
ListIterator<E>	**listIterator**() Returns a list iterator over the elements in this list (in proper sequence).
ListIterator<E>	**listIterator**(int index) Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list.
E	**set**(int index, E element) Replaces the element at the specified position in this list with the specified element.
List<E>	**subList**(int fromIndex, int toIndex) Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.

مثال:

```
public class ArrayListExample {
    public static void main(String[] args) {
        ArrayList<String> al = new ArrayList<String>();
        al.add("Ahmad");
        al.add("Khalid");
        al.add("Salma");
        al.add("Ahmad");
        al.add("Mona");
        //Print list
        System.out.println(al);
        //get the element with indwx 1
        System.out.println(al.get(1));
        //get the index of element "Salma"
        System.out.println(al.indexOf("Salma"));
        //Replace the element in postion 3 with value "Ali"
        al.set(3, "Ali");
        System.out.println(al);
        //Get Iterator
        ListIterator it = al.listIterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }
        System.out.println("*********List in reverse order**********");
        while (it.hasPrevious()) {
            System.out.println(it.previous());
        }
    }
}
```

LinkedList

- LinkedListهي أحد classesالناتجة من List interface وهي لها الصفات التالي:
- تسمح بتكرار العناصر المخزنة بها
- تستخدم doubly linked list
- يتم ترتيب العناصر حسب أولوية أدخالها

LinkedList constructors& methods

Constructor

`LinkedList()`

Constructs an empty list.

`LinkedList(Collection<? extends E> c)`

Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

methods

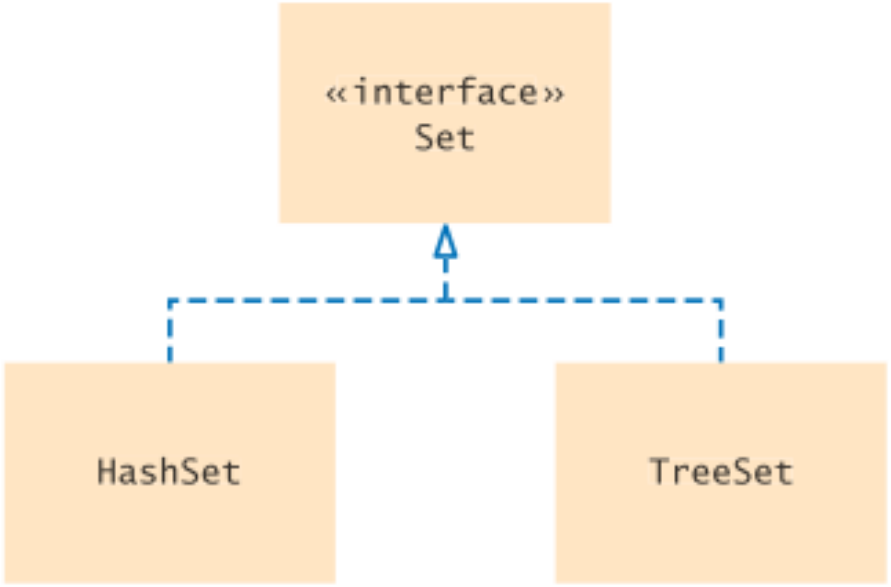
void	**addFirst(E e)** Inserts the specified element at the beginning of this list.
void	**addLast(E e)** Appends the specified element to the end of this list.
E	**element()** Retrieves, but does not remove, the head (first element) of this list.
E	**get(int index)** Returns the element at the specified position in this list.
E	**getFirst()** Returns the first element in this list.
E	**getLast()** Returns the last element in this list.
E	**removeFirst()** Removes and returns the first element from this list.
boolean	**removeFirstOccurrence(Object o)** Removes the first occurrence of the specified element in this list (when traversing the list from head to tail).
E	**removeLast()** Removes and returns the last element from this list.

مثال:

```
public class LinkedListExample {

    public static void main(String[] args) {
        LinkedList<String> staff = new LinkedList<String>();
        staff.addLast("Diana");
        staff.addLast("Harry");
        staff.addLast("Romeo");
        staff.addLast("Tom");

        ListIterator<String> iterator = staff.listIterator();
        iterator.next();
        iterator.next();

        iterator.add("Juliet");
        iterator.add("Nina");

        iterator.next();

        iterator.remove();

        System.out.println(staff);
        System.out.println("Expected: [Diana, Harry, Juliet, Nina, Tom]");
    }
}
```

Set Interface

- Set interface هي أحد interfaces الناتجة من Collection interface وهي لها الصفات التالي:
- لاتسمح بعملية التكرار في العناصر
- لايتم حفظ العناصر حسب الترتيب الذي ادخلت به.
- جد في لغة جافا classes مبنية على Set Interface وهي:
  - HashSet
  - TreeSet



HashSet

- هي classمبنية على Set Interfaceولكي يتم أستخدامها يجب أن يتوفر في العناصر hashCode method.

Constructor

HashSet() Constructs a new, empty set; the backing HashMap instance has default initial capacity (16) and load factor (0.75).
HashSet(Collection<? extends E> c) Constructs a new set containing the elements in the specified collection.
HashSet(int initialCapacity) Constructs a new, empty set; the backing HashMap instance has the specified initial capacity and default load factor (0.75).
HashSet(int initialCapacity, float loadFactor) Constructs a new, empty set; the backing HashMap instance has the specified initial capacity and the specified load factor.

مثال:

```java
public class HashSetExample {

    public static void main(String[] args) {
        HashSet <String> s = new HashSet <String>();
        s.add("Ali");
        s.add("Khalid");
        s.add("Zinab");
        s.add("Ali");
        s.add("Samira");
        System.out.println(s);
    }

}
```

مثال:

```
public class HashSetMethodsExample {
    public static void main(String[] args) {
        HashSet <String> s = new HashSet<String>();
        s.add("Ali");
        s.add("Khalid");
        s.add("Zinab");
        s.add("Ali");
        s.add("Samira");
        System.out.println("s= "+s);
        s.remove("Zinab");
        System.out.println("s= "+s);
        HashSet <String> s1 = new HashSet<String>();
        s1.add("Ahmad");
        s1.add( "Zinab");
        s1.add("Sami");
        s1.add("Khalid");
        System.out.println("s1= "+s1);
        System.out.println("is Ali in s1?  "+s1.contains("Ali"));
        System.out.println("After adding s1 to s  ");
        s.addAll(s1);
        System.out.println("s1= "+s);
        System.out.println("After removing s1 to s  ");
        s.removeAll(s1);
        System.out.println("s= "+s);
        s.retainAll(s1);
        System.out.println("s= "+s);
    }
}
```

TreeSet

- هي classمبنية على Set Interfaceو هي تكون مرتبة بترتيب معين (binary search tree) ولكي يتم أستخدامها يجب على عناصرها أن تقوم بعمل implementation لcomparable interfaceأو comparator interface.

Constructor

`TreeSet()`
Constructs a new, empty tree set, sorted according to the natural ordering of its elements.

`TreeSet(Collection<? extends E> c)`
Constructs a new tree set containing the elements in the specified collection, sorted according to the *natural ordering* of its elements.

`TreeSet(Comparator<? super E> comparator)`
Constructs a new, empty tree set, sorted according to the specified comparator.

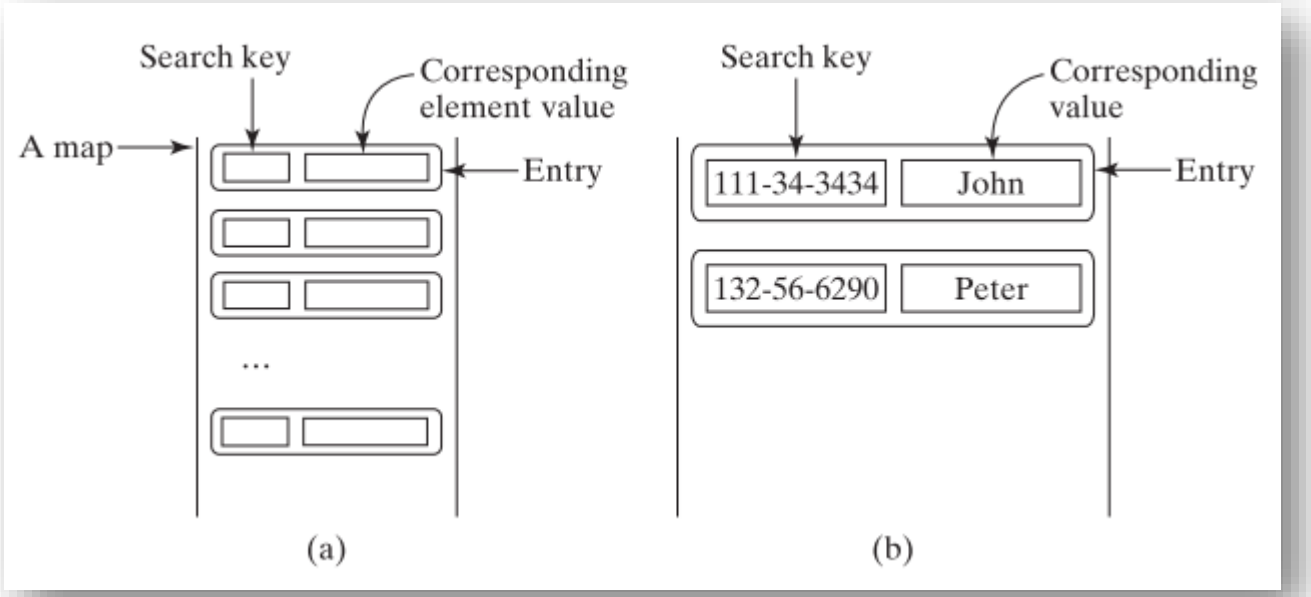
`TreeSet(SortedSet<E> s)`
Constructs a new tree set containing the same elements and using the same ordering as the specified sorted set.

مثال:

```
public class TreeSetExample {
    public static void main(String[] args) {
    Set<String> set = new HashSet<String>();
    set.add("London");
    set.add("Paris");
    set.add("New York");
    set.add("San Francisco");
    set.add("Beijing");
    set.add("New York");
    System.out.println("set: "+set);
    TreeSet<String> treeSet = new TreeSet<String>(set);
    System.out.println("Sorted tree set: " + treeSet);

    System.out.println("first(): " + treeSet.first());
    System.out.println("last(): " + treeSet.last());
    System.out.println("headSet(\"New York\"): " +
    treeSet.headSet("New York"));
    System.out.println("tailSet(\"New York\"): " +
    treeSet.tailSet("New York"));
  }
}
```

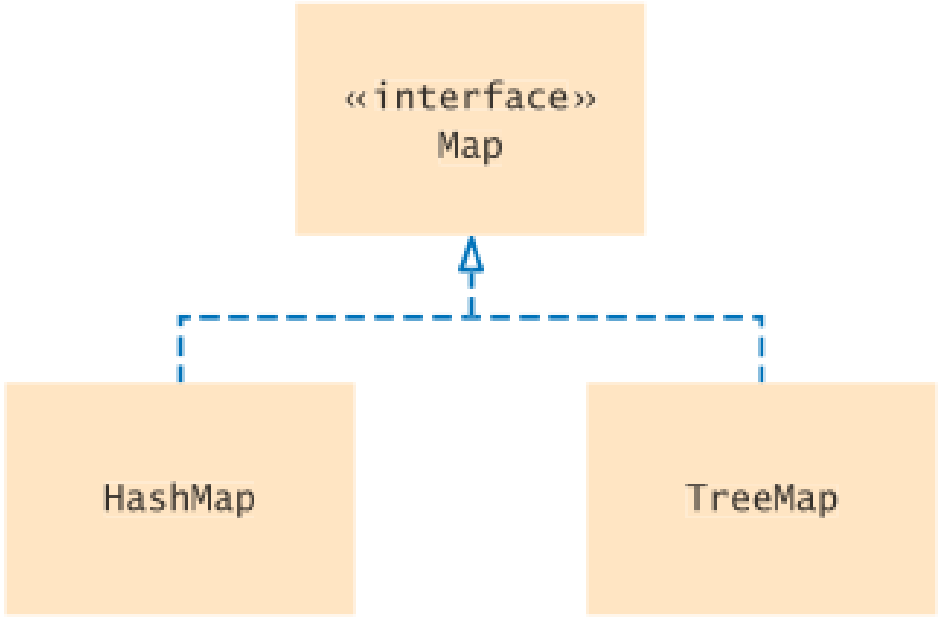
Map Interface

- هو نوع من تراكيب البيانات يستخدم في حفظ بيانات من نوع key/value.



Map Interface

- يوجد في لغة جافا Map Interface وهي: مبنية على classes
- HashMap
- TreeMap



HashMap

Constructor

`HashMap()`
Constructs an empty `HashMap` with the default initial capacity (16) and the default load factor (0.75).

`HashMap(int initialCapacity)`
Constructs an empty `HashMap` with the specified initial capacity and the default load factor (0.75).

`HashMap(int initialCapacity, float loadFactor)`
Constructs an empty `HashMap` with the specified initial capacity and load factor.

`HashMap(Map<? extends K,? extends V> m)`
Constructs a new `HashMap` with the same mappings as the specified `Map`.

TreeMap

Constructor

`TreeMap()`

Constructs a new, empty tree map, using the natural ordering of its keys.

`TreeMap(Comparator<? super K> comparator)`

Constructs a new, empty tree map, ordered according to the given comparator.

`TreeMap(Map<? extends K,? extends V> m)`

Constructs a new tree map containing the same mappings as the given map, ordered according to the *natural ordering* of its keys.

`TreeMap(SortedMap<K,? extends V> m)`

Constructs a new tree map containing the same mappings and using the same ordering as the specified sorted map.

مثال:

```
public class MapExample {
    public static void main(String[] args) {
    // Create a HashMap
    HashMap<String, Integer> hashMap = new HashMap<String, Integer>();
    hashMap.put("Smith", 30);
    hashMap.put("Anderson", 31);
    hashMap.put("Lewis", 29);
    hashMap.put("Cook", 29);
    System.out.println("Display entries in HashMap");
    System.out.println(hashMap + "\n");

    // Create a TreeMap from the preceding HashMap
    TreeMap<String, Integer> treeMap =
      new TreeMap<String, Integer>(hashMap);
    System.out.println("Display entries in ascending order of key");
    System.out.println(treeMap);

    // Display the age for Lewis
    System.out.println("\nThe age for " + "Lewis is " +
      treeMap.get("Lewis"));
  }
}
```

Comparable Example

```
public class Item1 implements Comparable<Item1> {

    String name;
    double price;

    public Item1(String name, double price) {
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString() {
        return "Item{" + "name=" + name + ", price=" + price

    }

    @Override
    public int compareTo(Item1 it) {
        return (this.price > it.price) ? 1 : -1;
    }

}
```

Comparable Example

```
public class TreeSetComparableExample1 {
    public static void main(String[] args) {
        TreeSet<Item1> treeSet = new TreeSet<Item1>();
        treeSet.add(new Item1("Sugar",1.50));
        treeSet.add(new Item1("Salt",1.00));
        treeSet.add(new Item1("Water",0.75));
        treeSet.add(new Item1("Juice",2.5));
        System.out.println(treeSet);

    }
}
```

شكراً