Fundamentos de programación

Grado de Ingeniería del Software

Jorge García Gutiérrez
jorgarcia@us.es
F1.56

Lo último que vimos fue...

- Metodología para definir tipos
- Inmutabilidad: Clases y Records
- Herencia e interfaces
- Tipo Object
- Tipo Comparable
- Restricciones y excepciones
- Constructor a partir de String

Illustrations by Pixeltrue on icons8

Índice

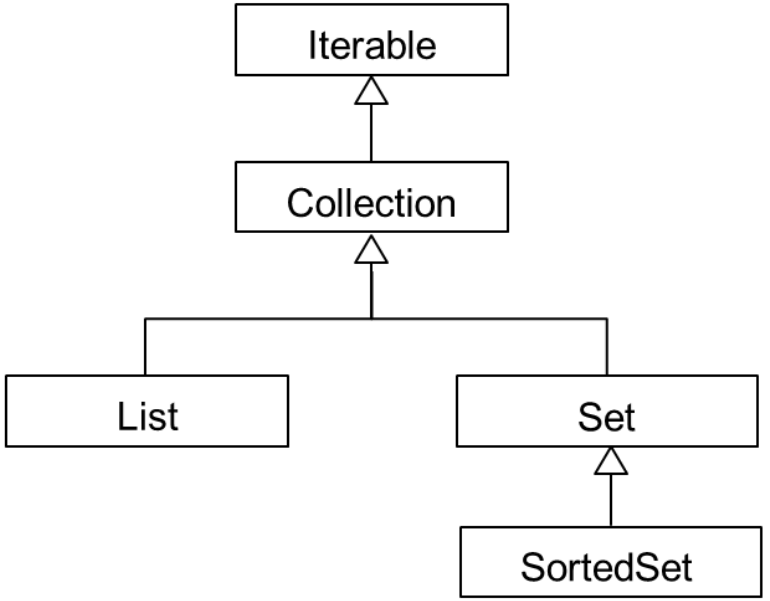
- Collections
- List
- Set
- SortedSet
- Map
- SortedMap

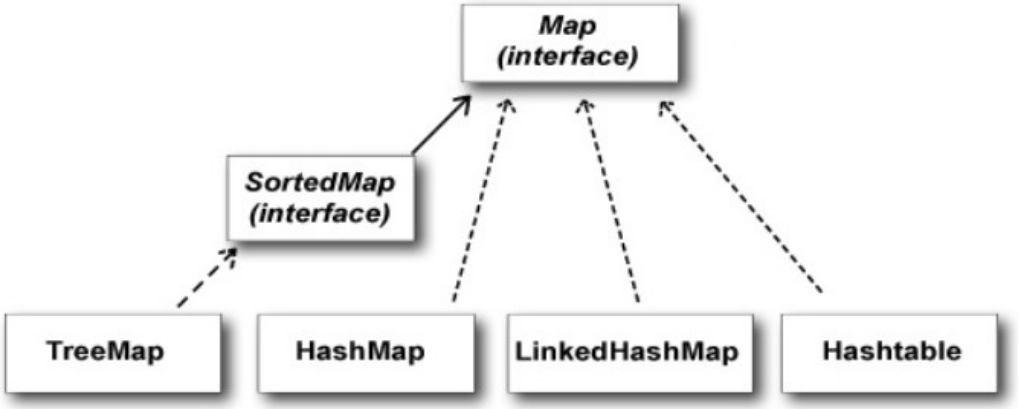
Illustrations by Pixeltrue on icons8

Índice

- **Collections**
- List
- Set
- SortedSet
- Map
- SortedMap

Illustrations by Pixeltrue on icons8





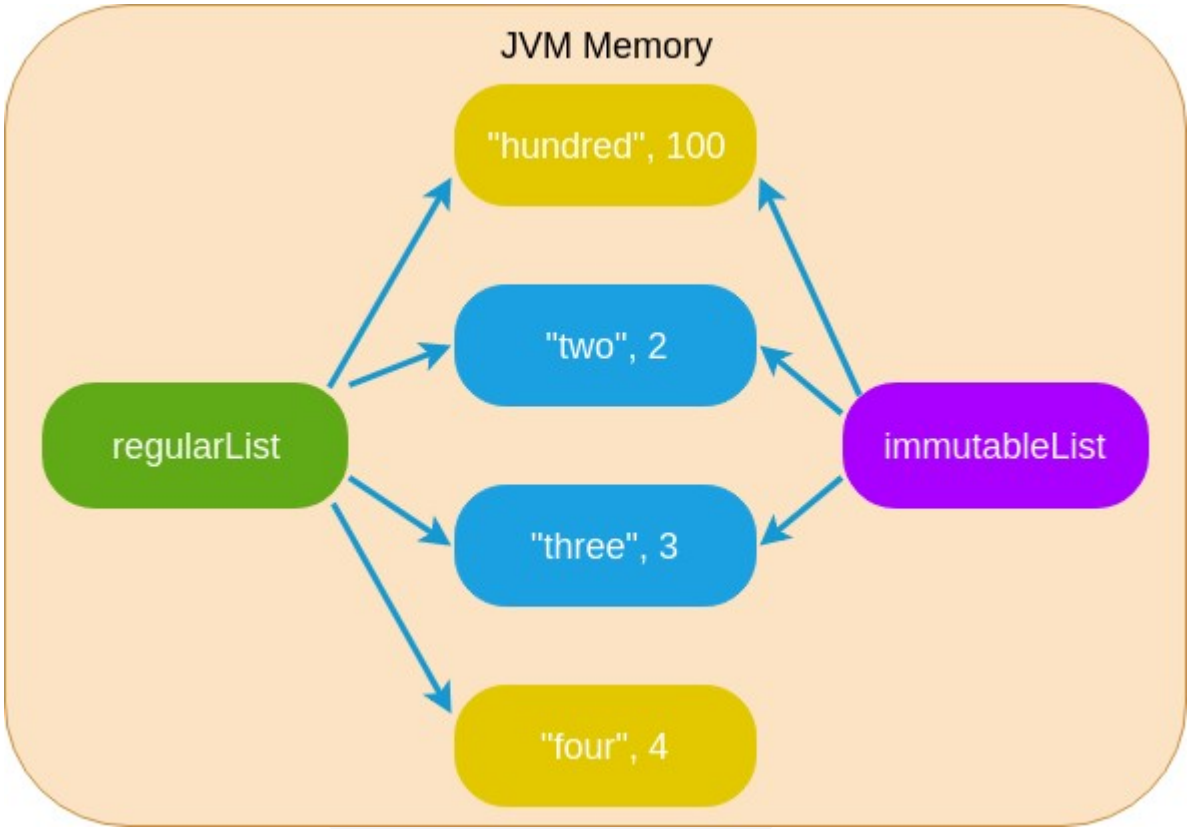
_____ : extends

----------: implements

https://www.w3schools.blog/hashmap-linkedhashmap-treemap-hashtable-java

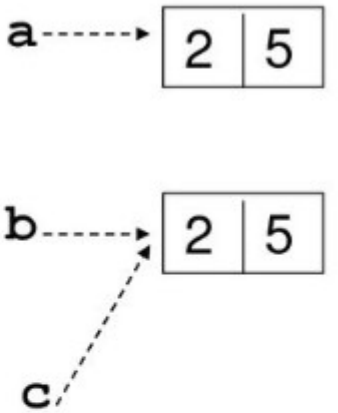
Collection	Collections
Collection is an interface.	Collections is a utility class.
Belongs **java.util** package.	Belongs **java.util** package.
Collection can be used to represent a group of individual objects as a single entity.	Collections define several utility methods (like sorting , searching --) for collection objects.
Collection is the root interface from which almost all Data-structures are derived, commonly known as collection framework.	Collections class contains many static methods with which Data-structures manipulation becomes easier.

https://www.youtube.com/watch?v=7x8RTD_-dHc



https://blog.wacekdziewulski.ovh/2019/10/28/java-unmodifiable-immutable-copyof/

```
class Point {
    int x; // x-coordinate
    int y; // y-coordinate
    Point(int x, int y) {
      this.x = x;
      this.y = y;
    }
}
a = new Point(2,5);
b = new Point(2,5);
c = b;
```



true or false? `a == b` ?
true or false? `b == c` ?
true or false? `a.equals(b)` ?
true or false? `b.equals(c)` ?

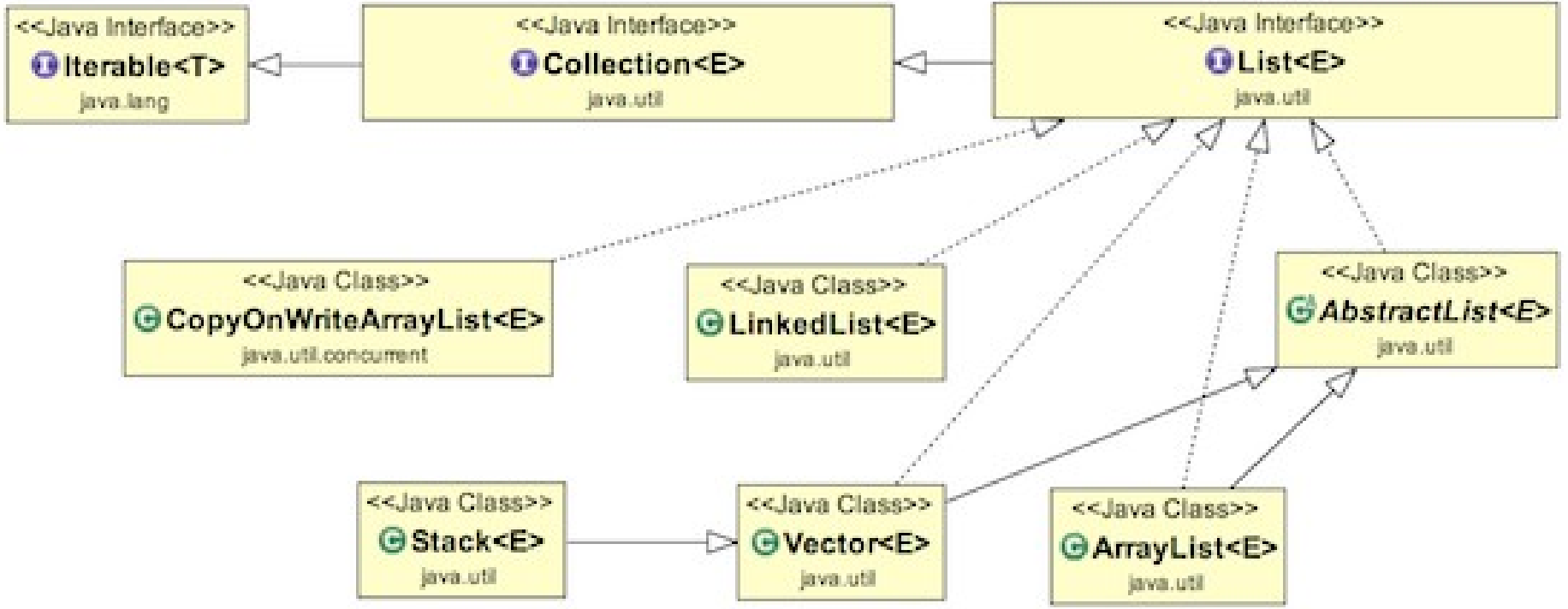
https://slideplayer.com/slide/14666106/

```java
public class TestLists {
    public static void main(String... args) {
        Persona p = new Persona("11111111X ,Jorge ,Garcia Gutierrez ,03/05/1981");
        List<Persona> lista = new ArrayList<>();
        lista.add(p);
        //lista = Collections.unmodifiableList(lista);
        p.setApellidos("");
        System.out.println(lista.get(0).getApellidos());
    }
}
```

Índice

- Collections
- **List**
- Set
- SortedSet
- Map
- SortedMap

Illustrations by Pixeltrue on icons8

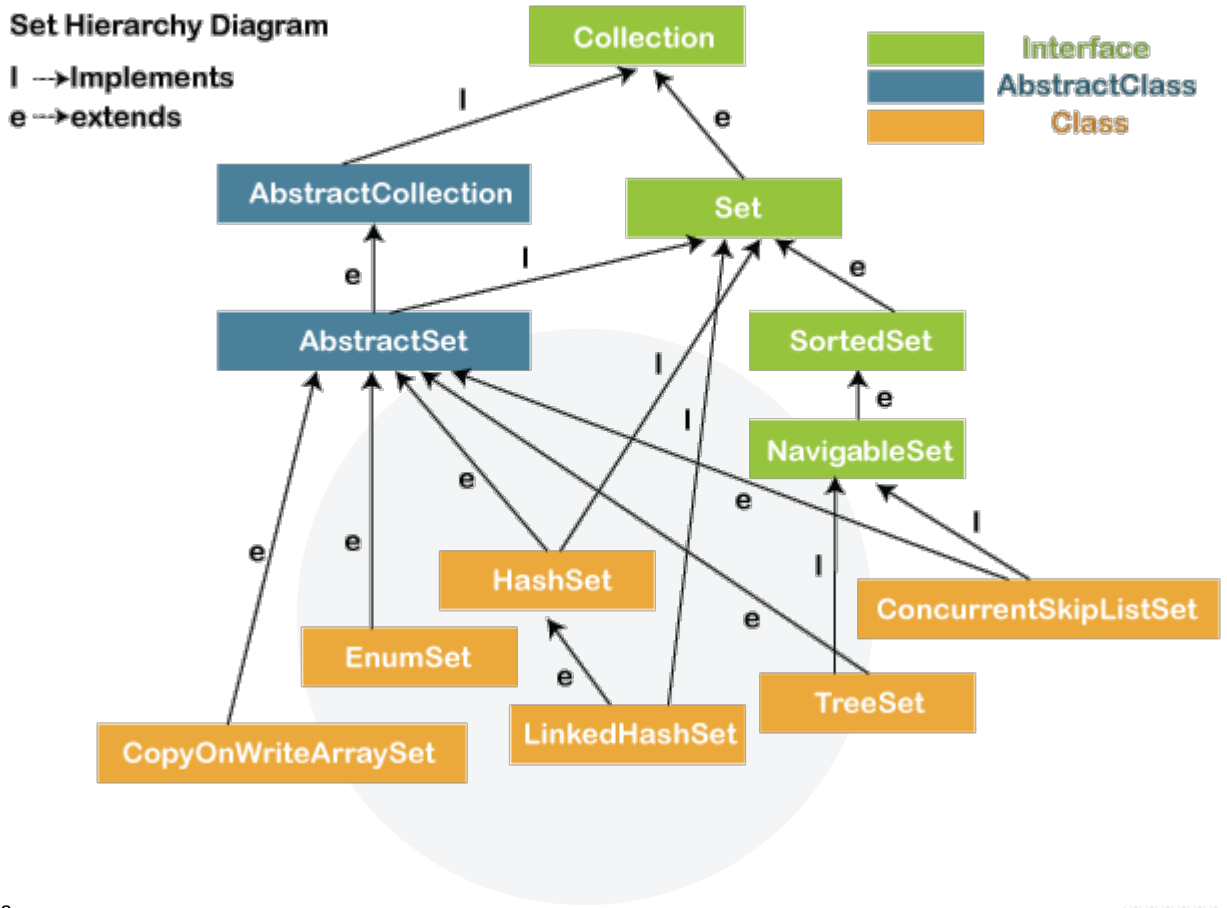


https://www.digitalocean.com/community/tutorials/java-list

Índice

- Collections
- List
- **Set**
- SortedSet
- Map
- SortedMap

Illustrations by Pixeltrue on icons8

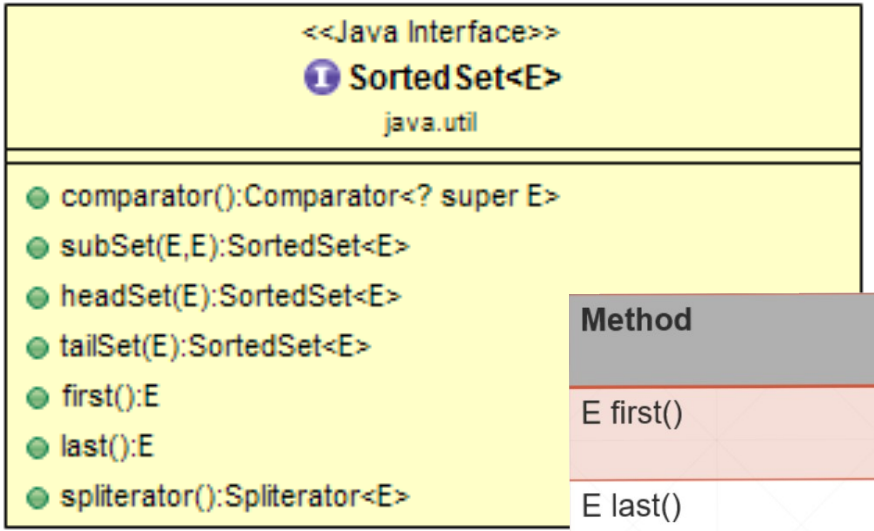


https://www.javatpoint.com/set-in-java

Índice

- Collections
- List
- Set
- **SortedSet**
- Map
- SortedMap

Illustrations by Pixeltrue on icons8



https://www.javatpoint.com/set-in-java

Method	Description
E first()	Returns First Element in the SortedSet.
E last()	Returns Last Element in the SortedSet.
SortedSet<E> HeadSet(E to)	Returns SortedSet in which all elements are lesser than the 'to' element.
SortedSet<E> TailSet(E from)	Returns SortedSet in which all elements are higher than the 'from' element or equal to 'from' element.
SortedSet<E> SubSet(E from, E to)	Returns SortedSet which is between 'from' & 'to'.

https://coding-examples.com/java/java-treeset-sortedset-explained/

compareTo and equals

- `compareTo` should generally be consistent with `equals`.
  - `a.compareTo(b) == 0` should imply that `a.equals(b)`.

- *equals-compareTo trick* - If your class needs to implement both `equals` and `compareTo`, you can take advantage:

```
public boolean equals(Object o) {
    if (o instanceof Employee) {
        Employee other = (Employee) o;
        return this.compareTo(other) == 0;
    } else {
        return false;
    }
}
```


https://slideplayer.com/slide/16337781/

Índice

- Collections
- List
- Set
- SortedSet
- **Map**
- SortedMap

Illustrations by Pixeltrue on icons8

Method Summary

void	**clear**() Removes all mappings from this map (optional operation).
boolean	**containsKey**(Object key) Returns true if this map contains a mapping for the specified key.
boolean	**containsValue**(Object value) Returns true if this map maps one or more keys to this value.
abstract Set	**entrySet**() Returns a set view of the mappings contained in this map.
boolean	**equals**(Object o) Compares the specified object with this map for equality.
Object	**get**(Object key) Returns the value to which this map maps the specified key.
int	**hashCode**() Returns the hash code value for this map.
boolean	**isEmpty**() Returns true if this map contains no key-value mappings.
Set	**keySet**() Returns a Set view of the keys contained in this map.
Object	**put**(Object key, Object value) Associates the specified value with the specified key in this map (optional operation).
void	**putAll**(Map t) Copies all of the mappings from the specified map to this map (optional operation).
Object	**remove**(Object key) Removes the mapping for this key from this map if present (optional operation).
int	**size**() Returns the number of key-value mappings in this map.
String	**toString**() Returns a string representation of this map.
Collection	**values**() Returns a collection view of the values contained in this map.

https://www.cs.cmu.edu/~pattis/15-1XX/15-200/lectures/collectionsiii/lecture.html

Índice

- Collections
- List
- Set
- SortedSet
- Map
- **SortedMap**

Illustrations by Pixeltrue on icons8

Inner classes inherited from class java.util.Map
Map.Entry

Method Summary

Comparator	**comparator**() Returns the comparator associated with this sorted map, or null if it uses its keys' natural ordering.
Object	**firstKey**() Returns the first (lowest) key currently in this sorted map.
SortedMap	**headMap**(Object toKey) Returns a view of the portion of this sorted map whose keys are strictly less than toKey.
Object	**lastKey**() Returns the last (highest) key currently in this sorted map.
SortedMap	**subMap**(Object fromKey, Object toKey) Returns a view of the portion of this sorted map whose keys range from fromKey, inclusive, to toKey, exclusive.
SortedMap	**tailMap**(Object fromKey) Returns a view of the portion of this sorted map whose keys are greater than or equal to fromKey.

Methods inherited from interface java.util.Map
clear, containsKey, containsValue, entrySet, equals, get, hashCode, isEmpty, keySet, put, putAll, remove, size, values

https://www.cs.cmu.edu/~pattis/15-1XX/15-200/lectures/collectionsiii/lecture.html

TAREA:

1. Leer comprensivamente el bloque 2 de teoría
2. Definir y testear el tipo Vuelo en sus dos versiones (mutable e inmutable).
3. Definir un tipo contenedor llamado Aeropuerto.