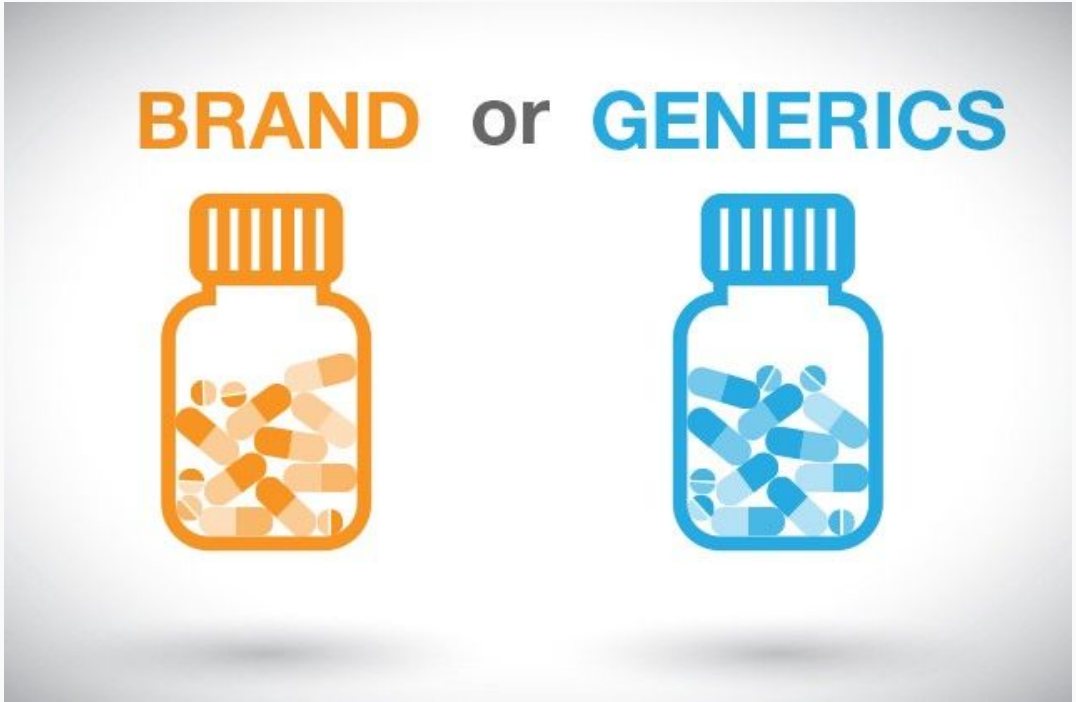
Лекция 5

Обобщения и коллекции

Generics



Проблема

```
public class Box
{
    private Object value;

    public Object getValue()
    {
        return value;
    }

    public void setValue(Object value)
    {
        this.value = value;
    }
}
```

```
Box box = new Box();
box.setValue("Hello");

String boxedValue = null;
if (box.getValue() instanceof  String) {
    boxedValue = (String)box.getValue();
}

System.out.println(boxedValue);
```

Обобщения

```java
public class GenericBox<T>
{
    private T value;

    public T getValue()
    {
        return value;
    }

    public void setValue(T value)
    {
        this.value = value;
    }
}
```

```java
GenericBox<String> genericBox = new GenericBox<>();
genericBox.setValue("Hello");
String genericBoxValue = genericBox.getValue();

System.out.println(genericBoxValue);
```

Имена параметров

- E - Element
- K - Key
- V - Value
- N - Number
- T,S,U,V - Type

Обобщенные методы

```
private static <T> int compare(GenericBox<T> first, GenericBox<T> second) {
    return first.getValue().toString().compareTo(second.getValue().toString());
}
```

Будет ли работать?

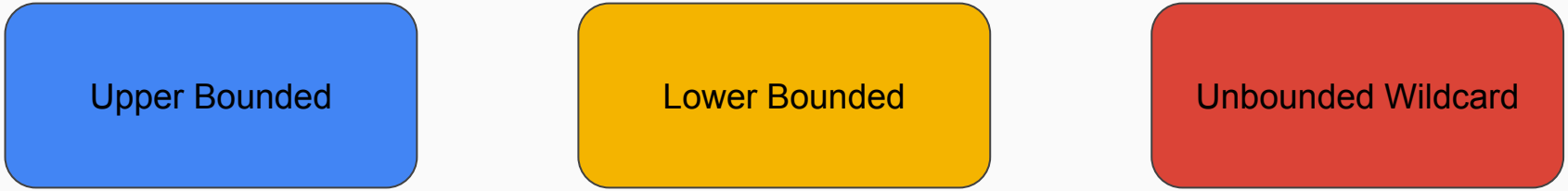
```
GenericBox<Number> numberBox = new GenericBox<>();
numberBox.setValue(Integer.valueOf("1"));
numberBox.setValue(Double.valueOf("1"));
```

Будет ли работать?

```
private static void testNumber() {
    GenericBox<Integer> boxedInt = new GenericBox<>();
    printNumber(boxedInt);
}

private static void printNumber(GenericBox<Number> boxedNumber) {
    System.out.println(boxedNumber.getValue());
}
```

Java Wildcard



Unbounded Wildcard

<?> - Неизвестный тип

Когда использовать:

- Когда используются только методы Object
- Когда выполнение метода не зависит от типа параметра

Lower Bounded

<? super Type> - нижняя граница

Consumer - потребитель данных

Upper Bounded

<? extends Type> - верхняя граница

Provider - поставщик данных

Producer Extends Consumer Super (PECS)

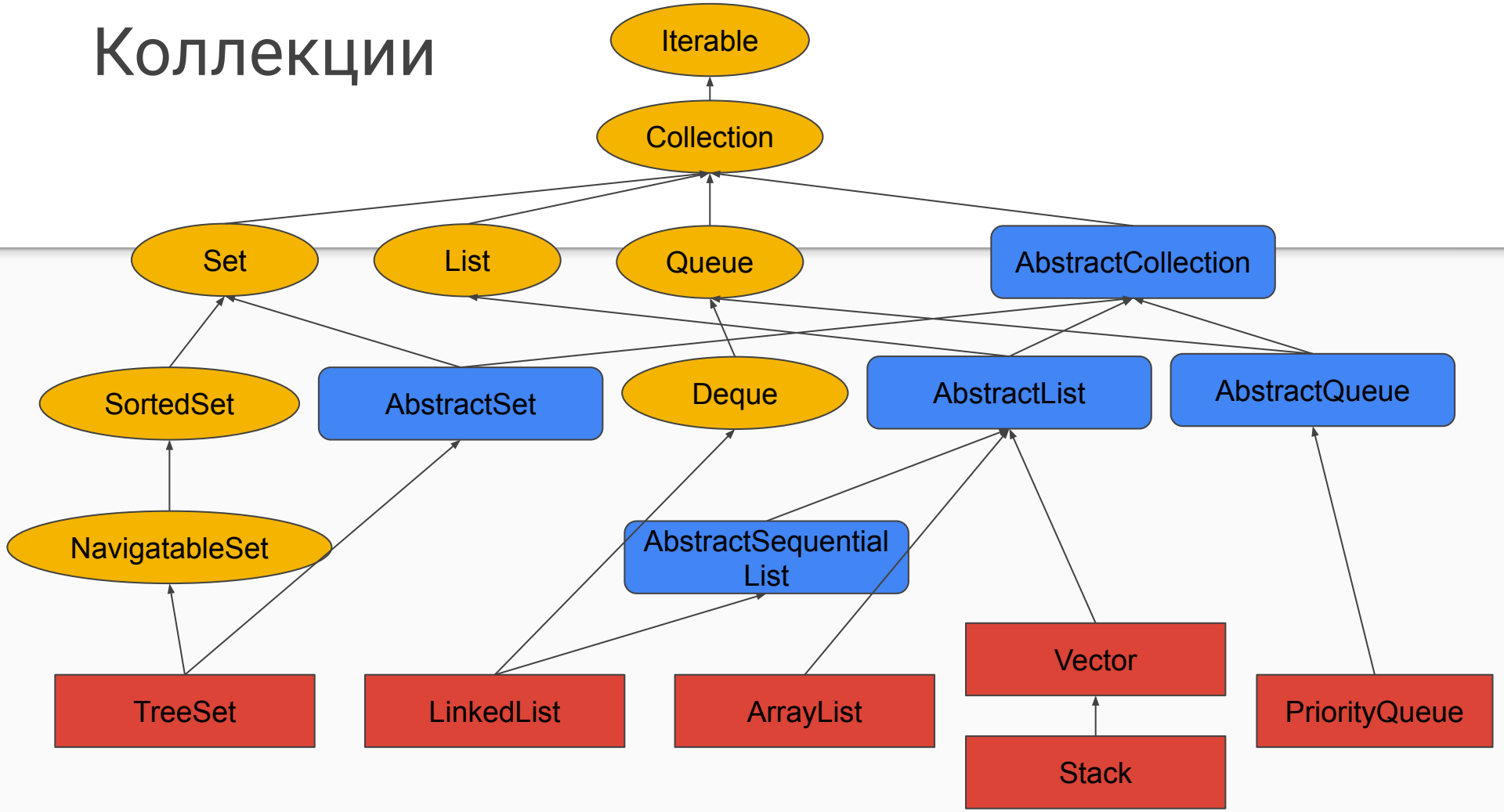
- Если метод имеет аргументы с параметризованным типом (например, Collection или Predicate), то в случае, если аргумент - производитель (producer), нужно использовать ? extends T, а если аргумент - потребитель (consumer), нужно использовать ? super T.
- Производитель и потребитель, кто это такие? Очень просто: если метод читает данные из аргумента, то этот аргумент - производитель, а если метод передаёт данные в аргумент, то аргумент является потребителем. Важно заметить, что определяя производителя или потребителя, мы рассматриваем только данные типа T.
- List<String> подтип List<? extends Object>
- List<Object> подтип List<? super String>

Что с чем может работать?

```
private static void printNumber1(GenericBox<? extends Number> boxedNumber) {
    System.out.println(boxedNumber.getValue().byteValue());
}

private static void printNumber2(GenericBox<? super Number> boxedNumber) {
    System.out.println(boxedNumber.getValue().byteValue());
}
```

Коллекции


Iterable
Collection
Set
List
Queue
AbstractCollection
SortedSet
AbstractSet
Deque
AbstractList
AbstractQueue
NavigatableSet
AbstractSequential List
Vector
TreeSet
LinkedList
ArrayList
PriorityQueue
Stack

Collection

- add(...)
- contains(...)
- remove(...)
- clear()
- iterator()
- size()
- isEmpty()

Iterator

- Collection<E> extends Iterable<E>
- Iterable<E>
  - Iterator<E> iterator();
- Iterator<E>
  - E next();
  - boolean hasNext();
  - void remove();

List

List<E> extends Collection<E>

- set(int index, E element)
- get(int index)
- indexOf(E element)

Реализации List

- ArrayList<>()
- LinkedList<>()

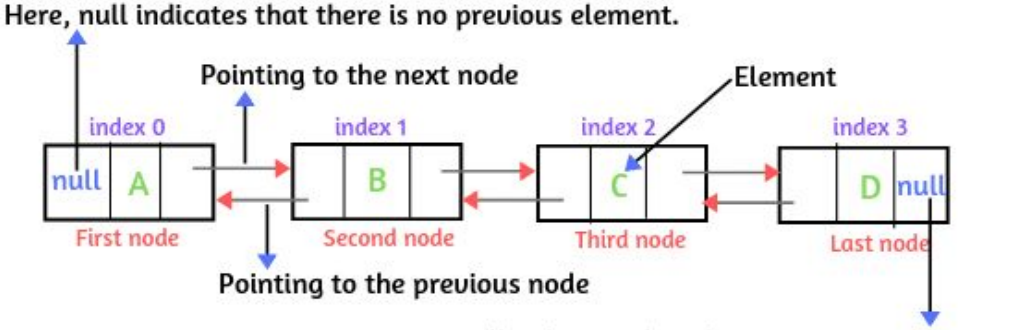
ArrayList

- Основан на массиве
- Инициализируется пустым массивом на 10 элементов
- Если массив заполнен то создаётся и заполняется новый массив в два раза больше предыдущего

LinkedList

- class LinkedList
  - Node head
  - Node tail
- private class Node
  - V value
  - Node next
  - Node prev

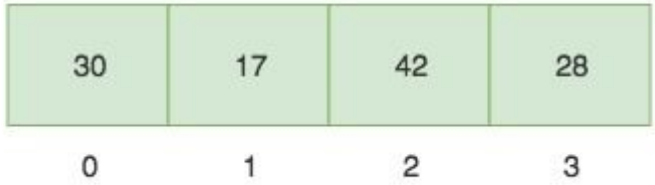
Representation of Java LinkedList Node



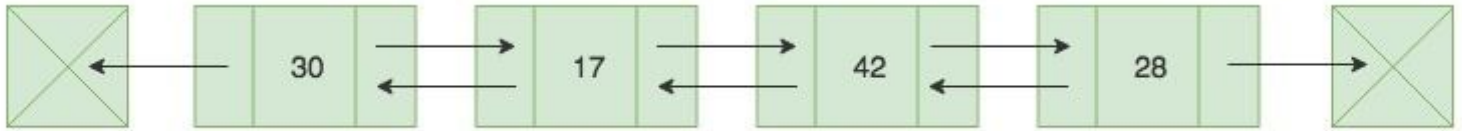
A array representation of linear Doubly LinkedList in Java

ArrayList / LinkedList

Java ArrayList Representation



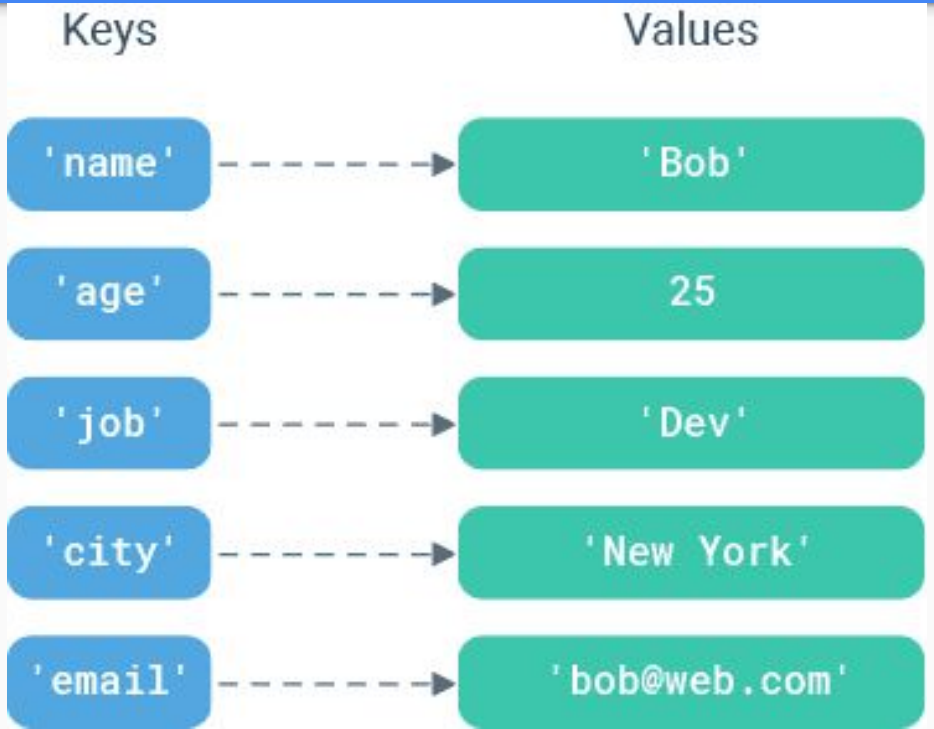
Java LinkedList Representation



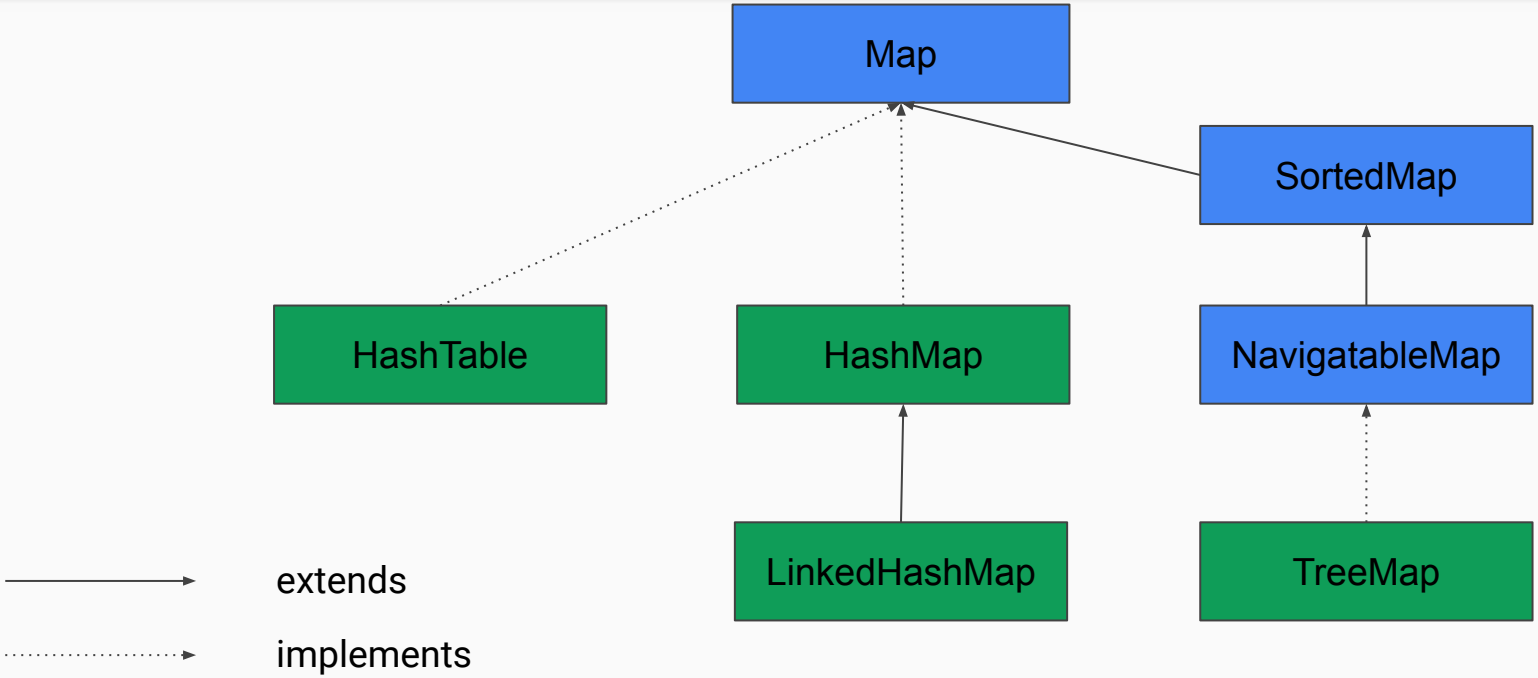
Set

Коллекция гарантирует уникальность элементов

Map

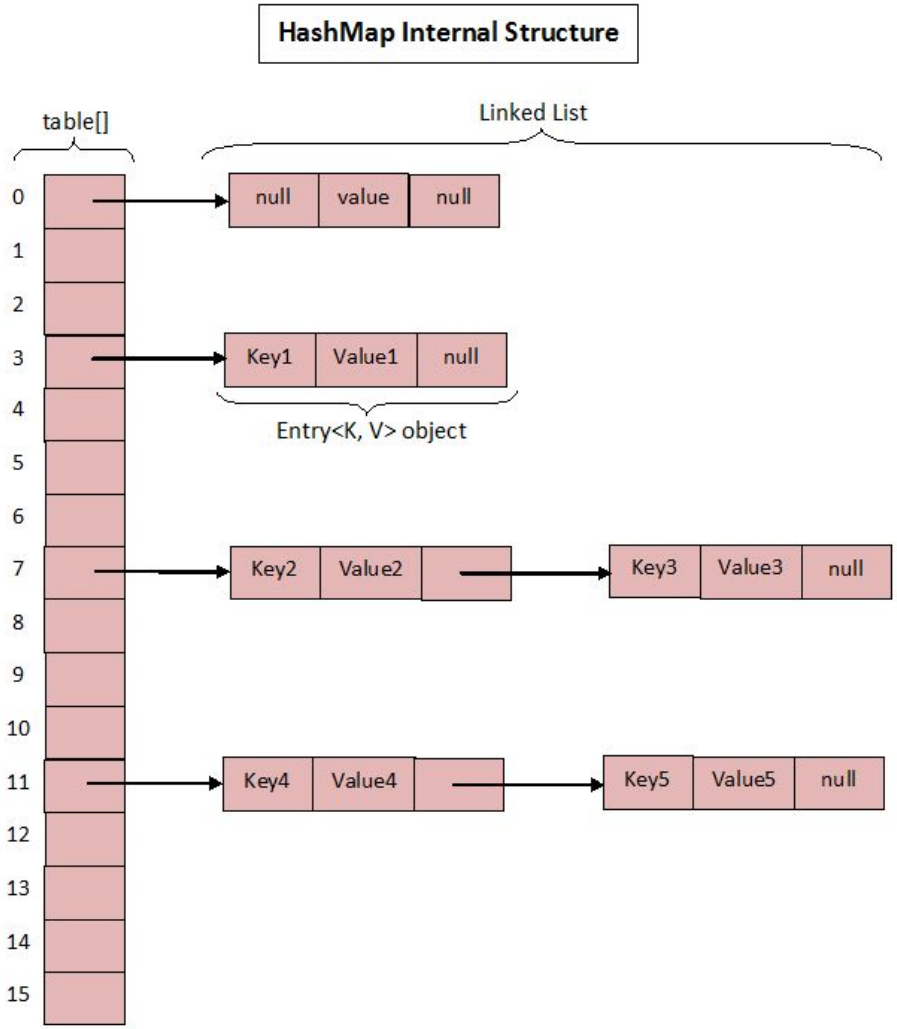

Keys
Values
'name'
'Bob'
'age'
25
'job'
'Dev'
'city'
'New York'
'email'
'bob@web.com'

Map<K, V>


Map
SortedMap
HashTable
HashMap
NavigatableMap
LinkedHashMap
TreeMap
extends
implements

HashMap

- table — Массив типа Entry[], который является хранилищем ссылок на списки (цепочки) значений;
- loadFactor — Коэффициент загрузки. Значение по умолчанию 0.75 является хорошим компромиссом между временем доступа и объемом хранимых данных;
- threshold — Предельное количество элементов, при достижении которого, размер хэш-таблицы увеличивается вдвое. Рассчитывается по формуле (capacity * loadFactor);
- size — Количество элементов HashMap-а;


HashMap Internal Structure
table[]
Linked List
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
null
value
null
Key1
Value1
null
Entry<K, V> object
Key2
Value2
Key3
Value3
null
Key4
Value4
Key5
Value5
null