

Сетове - Колекции , които пазят в себе си само уникални елементи.

1. Sets

- **HashSet<E>**
- **TreeSet<E>**
- **LinkedHashSet<E>**

Сетове доставят функции за добавяне/премахване/търсене на елементи и са МНОГО БЪРЗИ. Варианти на Сетове са:

HashSet<E> -> При него елементите са подредени на случаен принцип. Числото, което се получи при хеширането, елемента отива на тази позиция в сета.

TreeSet<E> -> Тук елементите са подредени по азбучен ред.

LinkedHashSet<E> -> Работи подобно на опашките.. първият елемент си остава първи.

Как се инициализира ?

■ Initialization

```
HashSet<String> hash = new HashSet<String>();
```

Може също и по следния начин:

■ For easy reading you can use diamond inference syntax

```
TreeSet<String> tree = new TreeSet<>();
```

Полезни методи на всеки един от сетовете са:

.size(); -> връща броят на елементите в сета

.isEmpty(); -> проверява дали сета е празен. Връща true or false.

```
HashSet<String> hash = new HashSet<>();  
System.out.println(hash.size()); // 0  
System.out.println(hash.isEmpty()); // True
```

ПРИ СЕТОВЕТЕ НЯМАМЕ ИНДЕКСАЦИЯ. Елементите се обхожда с foreach.

HashSet Е НАЙ-БЪРЗ ОТ ВСИЧКИТЕ СЕТОВЕ.

Мапове(Maps или Associative arrays) - Имаме две стойности вътре - key и value

2. Maps

- `HashMap<K, V>`
- `TreeMap<K, V>`
- `LinkedHashMap<K, V>`



Java Advanced - Sets and Maps - януари 2017 - Венцислав Иванов



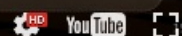
Associative Arrays (Maps)

- Associative arrays are arrays indexed by keys
 - Not by the numbers 0, 1, 2, ...
- Hold a set of pairs **<key, value>**
- Traditional array
- Associative array

| key | 0 | 1 | 2 | 3 | 4 |
|-------|---|----|----|-----|----|
| value | 8 | -3 | 12 | 408 | 33 |

| key | value |
|------------|-------------|
| John Smith | +1-555-8976 |
| Lisa Smith | +1-555-1234 |
| Sam Doe | +1-555-5030 |

1:28:18 / 2:13:51



Как се инициализира ?

Initialization

```
HashMap<String, Integer> hash = new HashMap<>();
```

Притежават същите методи като сетовете - `size()` и `isEmpty()`

- `.size()`
- `.isEmpty()`

```
HashSet<String> hash = new HashSet<>();  
System.out.println(hash.size()); // 0  
System.out.println(hash.isEmpty()); // True
```

Задължително е в един мап всички ключове да са от един тип както и стойностите.

В `HashMap` елементи се добавят с `put()`, а не с `add()`.

`HashMap` не хешира стойностите, а само ключовете.

Общо взето освен присвояването на нова стойност на `value-to`, за което ни трябва новата стойност на `value-to`, всичко останало се прави само върху ключовете(`keys`).

Обхождане на HashMap:

```
HashMap<String, Integer> vehicles = new HashMap<>();
vehicles.put("BMW", 5);
vehicles.put("Mercedes", 3);
vehicles.put("Audi", 4);
vehicles.put("BMW", 10);
for(String key: vehicles.keySet())
    System.out.println(key + " - " + vehicles.get(key));
```

На картинката е показано итериране на мап по ключове(keys). Достъпването на value-то става чрез vehicles.get(key);

Java Advanced - Sets and Maps - януари 2017 - Венцислав Иванов

Looping Through Maps - Example

SOFTWARE UNIVERSITY FOUNDATION

```
HashMap<String, Integer> vehicles = new HashMap<>();
vehicles.put("BMW", 5);
vehicles.put("Mercedes", 3);
vehicles.put("Audi", 4);
vehicles.put("BMW", 10);
for(String key: vehicles.keySet())
    System.out.println(key + " - " + vehicles.get(key));
```

Override first value

Return set of all keys

Return value for key

```
Audi - 4
Mercedes - 3
BMW - 10
```

1:37:23 / 2:13:51

YouTube

Друг вариант за обхождане е:

```
for (Map.Entry<String, Integer> entries : results.entrySet()) {
    System.out.println(entries.getKey() + " - " + entries.getValue() + " times");
}
```