Professor Hans Noodles

41 уровень

23.11.2019 56398 8 + 3

Ответы на самые популярные вопросы об интерфейсе Map

Привет! Сегодня мы дадим ответы на самые распространенные вопросы о Map, но для начала давай вспомним, что это такое.

Map — это структура данных, которая содержит набор пар “ключ-значение”. По своей структуре данных напоминает словарь, поэтому ее часто так и называют. В то же время, Map является **интерфейсом**, и в стандартном jdk содержит основные реализации: Hashmap, LinkedHashMap, Hashtable, TreeMap. Самая используемая реализация — Hashmap, поэтому и будем ее использовать в наших примерах.

Вот так выглядит стандартное создание и заполнение мапы:

```
Map<Integer, String> map = new HashMap<>();
map.put(1, "string 1");
map.put(2, "string 2");
map.put(3, "string 3");
```

А так — получение значений по ключу:

```
String string1 = map.get(1);
String string2 = map.get(2);
```

Если все из вышесказанного понятно, приступим к нашим ответам на популярные вопросы!

0. Как перебрать все значения Map

Перебор значений — самая частая операция, которую вы выполняете с мапами. Все пары (ключ-значение) хранятся во внутреннем интерфейсе Map.Entry, а чтобы получить их, нужно вызвать метод `entrySet()`. Он возвращает множество (Set) пар, которые можно перебрать в цикле:

```
for(Map.Entry<Integer, String> entry: map.entrySet()) {
   // get key
   Integer key = entry.getKey();
   // get value
   String value = entry.getValue();
}

Или используя итератор:
Iterator<Map.Entry<Integer, String>> itr = map.entrySet().iterator();
while(itr.hasNext()) {
   Map.Entry<Integer, String> entry =  itr.next();
   // get key
   Integer key = entry.getKey();
   // get value
   String value = entry.getValue();
}
```

1. Как конвертировать Map в List

У интерфейса Map существует 3 метода, которые возвращают перечень элементов:

- keySet() — возвращает множество(Set) ключей;
- values() — возвращает коллекцию(Collection) значений;
- entrySet() — возвращает множество(Set) наборов “ключ-значение”.

Если заглянуть в конструкторы класса `ArrayList`, можно заметить, что существует конструктор с аргументом типа Collection. Так как Set является наследником Collection, результаты всех вышеупомянутых методов можно передать в конструктор класса `ArrayList`. Таким образом, мы создадим новые списки и заполним их значениями из `Map`:

```
// key list
List<Integer> keyList = new ArrayList<>(map.keySet());
// value list
List<String> valueList = new ArrayList<>(map.values());
// key-value list
List<Map.Entry<Integer, String>> entryList = new ArrayList<>(map.entrySet());
```

2. Как отсортировать ключи мапы

Сортировка мап — тоже довольно частая операция в программировании. Сделать это можно несколькими способами:

1. Поместить Map.Entry в список и отсортировать его, используя Comparator.

   В компараторе будем сравнивать исключительно ключи пар:

```
List> list = new ArrayList(map.entrySet());
Collections.sort(list, new Comparator<Map.Entry<Integer, String>>() {
```

```
        return o1.getKey() - o2.getKey();
    }
});
```

Если разобрался с лямбдами, эту запись можно существенно сократить:

```
Collections.sort(list, Comparator.comparingInt(Map.Entry::getKey));
```

2. Использовать `SortedMap`, а точнее, ее реализацию — `TreeMap`, которая в конструкторе принимает Comparator. Данный компаратор будет применяться к ключам мапы, поэтому ключами должны быть классы, реализующие интерфейс `Comparable`:

```
SortedMap<Integer, String> sortedMap = new TreeMap<>(new Comparator<Integer>() {
    @Override
    public int compare(Integer o1, Integer o2) {
        return o1 - o2;
    }
});
```

И, конечно, все можно переписать, используя лямбды:

```
SortedMap<Integer, String> sortedMap = new TreeMap<>(Comparator.comparingInt(o -> o));
```

В отличие от первого способа, используя SortedMap, мы всегда будем хранить данные в отсортированном виде.

3. Как отсортировать значения мапы

Здесь стоит использовать подход, аналогичный первому для ключей — получать список значений и сортировать их в списке:

```
List <Map.Entry<Integer, String>> valuesList = new ArrayList(map.entrySet());
Collections.sort(list, new Comparator<Map.Entry<Integer, String>>() {
    @Override
    public int compare(Map.Entry<Integer, String> o1, Map.Entry<Integer, String> o2) {
        return o1.getValue().compareTo(o2.getValue());
    }
});
```

И лямбда для этого выглядит так:

```
Collections.sort(list, Comparator.comparing(Map.Entry::getValue));
```

4. В чем разница между HashMap, TreeMap, и Hashtable

Как упоминалось ранее, существуют 3 основные реализации интерфейса Map. У каждой из них есть свои особенности:

1. **Порядок элементов.** `HashMap` и `Hashtable` не гарантируют, что элементы будут храниться в порядке добавления. Кроме того, они не гарантируют, что порядок элементов не будет меняться со временем. В свою очередь, `TreeMap` гарантирует хранение элементов в порядке добавления или же в соответствии с заданным компаратором.

2. **Допустимые значения.** `HashMap` позволяет иметь ключ и значение null, `HashTable` — нет. TreeMap может использовать значения null только если это позволяет компаратор. Без использования компаратора (при хранении пар в порядке добавления) значение null не допускается.

И общее сравнение реализаций:

	HashMap	HashTable	TreeMap
Упорядоченность элементов	нет	нет	да
null в качестве значения	да	нет	да/нет
Потокобезопасность	нет	да	нет
Алгоритмическая сложность поиска элементов	O(1)	O(1)	O(log n)
Структура данных под капотом	хэш-таблица	хэш-таблица	красно-чёрное дерево

5. Как создать двунаправленную мапу

Иногда появляется необходимость использовать структуру данных, в которой и ключи, и значения будут уникальными, то есть мапа будет содержать пары “ключ-ключ”.

Такая структура данных позволяет создать "инвертированный просмотр/поиск" по мапе. То есть, мы можем найти ключ по его значению.Эту структуру данных называют двунаправленной мапой, которая, к сожалению, не поддерживается JDK. Но, к счастью, ее реализацию можно найти в библиотеках Apache Common Collections или Guava. Там она называется BidiMap и BiMap соответственно. Эти реализации вводят ограничения на уникальность ключей и значений. Таким образом получаются отношения one-to-one.

6. Как создать пустую Map

Создать пустую мапу можно двумя способами:

1. Обычная инициализация объекта:

```
1  Map<Integer, String> emptyMap = new HashMap<>();
```

2. Создание неизменяемой (immutable) пустой мапы:

```
1  Map<Integer, String> emptyMap =  Collections.emptyMap();
```

При попытке добавления данных в такую мапу мы получим:

`UnsupportedOperationException` исключение.

В этой статье мы рассмотрели самые частые вопросы, которые могли возникнуть у тебя при использовании интерфейса Map.

Что еще почитать:

- Для чего в Java нужны интерфейсы
- HashMap — что за карта такая?

+119

Комментарии (8) + 3 популярные новые старые

JavaCoder

Введите текст комментария

Сергей Уровень 41, Санкт-Петербург, Russian Federation 2 августа, 19:22

Исправьте, пожалуйста, код в разделе «2. Как отсортировать ключи мапы»
с

```
List> list = new ArrayList(map.entrySet());
```

на

```
List<Map.Entry<Integer, String>> list = new ArrayList<>(map.entrySet());
```

Проверьте, пожалуйста, код в разделе "3. Как отсортировать значения мапы", в нём надо переименовать переменную.

Ответить 0

Dmytryi Shubchynskyi Уровень 40, Мариуполь, Украина 22 февраля, 15:40

для чего нужно Создание неизменяемой (immutable) пустой мапы ?

Ответить 0

Sergey Kornilov Уровень 39, Petropavlovsk, Казахстан 5 июля 2021, 13:41

В закладки.

Ответить +1

Иван Уровень 41, Москва 8 января 2021, 18:34

Так же есть ошибка в пункте 4, подпункт 3, где говорится про потокобезопасность HashTable. В тексте нужно поменять местами HashMap и HashTable и тогда таблица будет верно читаться.

Ответить +1

Yuliia Boklah Уровень 41, Киев, Украина 6 сентября 2020, 18:56

В свою очередь, TreeMap гарантирует хранение элементов в порядке добавления или же в соответствии с заданным компаратором.

Указана совершенно неправильная информация. Речь идет об LinkedHashMap. В TreeMap элементы хранятся в есттественном порядке (natural ordering) или же в соответствии с заданным компаратором. Исправьте ошибку и не вводите учеников в заблуждение.

Ответить +9

Кирилл Уровень 13, Москва, Россия 21 июля 2020, 21:35

А зачем нужна мапа,в которую даже нельзя добавить какие либо данные ?

Ответить 0

fog Уровень 18 22 мая 2021, 17:43

Очевидно же: для того, чтобы *не-добавлять* туда данные ;)

Ответить 0

DiHASTRO Уровень 4, Kazakhstan 20 августа, 13:32

https://stackoverflow.com/questions/14846920/collections-emptymap-vs-new-hashmap

Ответить 0

НАЧАТЬ ОБУЧЕНИЕ

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.

ПОДПИСЫВАЙТЕСЬ

ЯЗЫК ИНТЕРФЕЙСА

Русский