**IV. Сравнение реализации самых популярных альтернативных коллекций в разных библиотеках**

**Содержание**

**4.1 Реализация мультимножества (MultiSet/Bag) в библиотеках guava, Apache Commons Collections и GS  
Примеры использования мультимножества (MultiSet/Bag) для подсчета слов в тексте**

1. разных вариантов Multiset от Guava:

HashMultiset , TreeMultiset, LinkedHashMultisetTest, ConcurrentHashMultiset

2. разных вариантов Bag от Apache Commons Collections:

HashBag, TreeBag, SynchronizedBag, SynchronizedSortedBag

3. разных вариантов Bag от GS Collections:

MutableBag, MutableSortedBag

4. Ну и наконец, посмотрим как можно сделать тоже самое в чистом JDK :

Эмуляция multiSet через HashMap.

**4.2 Реализация Multimap в библиотеках guava, Apache Commons Collections и GS Collections**

Примеры использования Multimap для сохранения всех вхождений слов в тексте

1. разных вариантов Multimap от Guava:

HashMultimap, ArrayListMultimapTest, LinkedHashMultimapTest, LinkedListMultimapTest, TreeMultimapTest

2. разных вариантов MultiValueMap от Apache Commons Collections:

MultiValueMap, MultiValueMap, оборачивающий TreeMap<String, TreeSet>(), MultiValueMap, оборачивающий LinkedHashMap<String, LinkedHashSet>()

3. разных вариантов Multimap от GS Collections:

FastListMultimap, HashBagMultimap, TreeSortedSetMultimap

4. Ну и наконец, посмотрим как можно сделать тоже самое в чистом JDK:

Эмуляция multiMap через HashMap.

**4.3 Реализация BiMap в библиотеках guava, Apache Commons Collections ,GS Collections и JDK**

**Примеры использования BiMap для создания русско-английского «переводчика», действующего в обе стороны**

1. разных вариантов BiMap от Guava:

BiMap, EnumBiMap, EnumHashBiMap

2. C помощью BidiMap от Apache Commons Collections:

DualHashBidiMap

3. C помощью HashBiMap от GS Collections:

HashBiMap

4. Ну и наконец, посмотрим как можно сделать тоже самое в чистом JDK:

Используем две HashMap для эмуляции BiMap

**4.1 Реализация мультимножества (MultiSet/Bag) в библиотеках guava, Apache Commons Collections и GS  
Collections**

Итак, мультимножество это множество, которое сохраняет не только факт наличие элементов в множестве, но и количество  
вхождений в него. В JDK его можно эмулировать конструкцией Map  
<T, Integer>, но, естественно, специализированные коллекции позволяют использовать значительно меньше кода. Сравним  
какие реализации данной коллекции предлагают разные библиотеки:  
  
**Внимание**: если таблица не помещается целиком, попробуйте уменьшить масштаб страницы или открыть в другом браузере.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Тип коллекции** | **Guava** | **Apache Commons Collections** | **GS Collections** | **JDK** |
| Порядок коллекции не определен | [HashMultiset](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/HashMultiset.html) | [HashBag](http://commons.apache.org/proper/commons-collections/javadocs/api-3.2.1/org/apache/commons/collections/bag/HashBag.html) | [HashBag](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/bag/mutable/HashBag.html) | [HashMap<String, Integer>](https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html) |
| Отсортированная в заданном или натуральном порядке | [TreeMultiset](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/TreeMultiset.html) | [TreeBag](http://commons.apache.org/proper/commons-collections/javadocs/api-3.2.1/org/apache/commons/collections/TreeBag.html) | [TreeBag](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/bag/sorted/mutable/TreeBag.html) | [TreeMap<String, Integer>](https://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html) |
| В порядке добавления | [LinkedHashMultiset](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/LinkedHashMultiset.html) | - | - | [LinkedHashMap<String, Integere>](https://docs.oracle.com/javase/8/docs/api/java/util/LinkedHashMap.html) |
| Многопоточные | [ConcurrentHashMultiset](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/ConcurrentHashMultiset.html) | [SynchronizedBag](http://commons.apache.org/proper/commons-collections/javadocs/api-3.2.1/org/apache/commons/collections/bag/SynchronizedBag.html) | [SynchronizedBag](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/bag/mutable/SynchronizedBag.html) | [Collections.synchronizedMap(HashMap<String, Integer>)](https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html) |
| Многопоточные и отсортированные | - | [SynchronizedSortedBag](http://commons.apache.org/proper/commons-collections/javadocs/api-3.2.1/org/apache/commons/collections/bag/SynchronizedSortedBag.html) | [SynchronizedSortedBag](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/bag/sorted/mutable/SynchronizedSortedBag.html) | [Collections.synchronizedSortedMap(TreeMap<String, Integer>)](https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html) |
| Не изменяемые | [ImmutableMultiset](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/ImmutableMultiset.html) | [UnmodifiableBag](http://commons.apache.org/proper/commons-collections/javadocs/api-3.2.1/org/apache/commons/collections/bag/UnmodifiableBag.html) | [UnmodifiableBag](http://www.goldmansachs.com/gs-collections/javadoc/6.1.0/com/gs/collections/impl/bag/mutable/UnmodifiableBag.html) | [Collections.unmodifiableMap(HashMap<String, Integer>)](https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html) |
| Не изменяемые и отсортированные | [ImmutableSortedMultiset](http://google.github.io/guava/releases/snapshot/api/docs/com/google/common/collect/ImmutableSortedMultiset.html) | [UnmodifiableSortedBag](http://commons.apache.org/proper/commons-collections/javadocs/api-3.2.1/org/apache/commons/collections/bag/UnmodifiableSortedBag.html) | [UnmodifiableSortedBag](http://unmodifiablesortedbag/) | [Collections.unmodifiableSortedMap(TreeMap<String, Integer>)](https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html) |

**Примеры использования мультимножества (MultiSet/Bag) для подсчета слов в тексте**

**Есть задача**: дана строчка текста «Hello World! Hello All! Hi World!», нужно разобрать её на отдельные  
слова где разделитель только пробел, сохранить в какую-нибудь коллекцию и вывести количество вхождений каждого  
слова, общее количество слов в тексте и количество уникальных слов.  
  
Посмотрим как это сделать с помощью  
**1. разных вариантов Multiset от Guava:**

Используем HashMultiset от guava для подсчета слов

Обратите внимание, что порядок вывода в System.out.println(multiset) и в  
System.out.println(multiset.elementSet()) — произвольный, то есть не определен.

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

Multiset<String> multiset = HashMultiset.create(Arrays.asList(INPUT\_TEXT.split(" ")));

*// Выводим кол-вом вхождений слов*

System.out.println(multiset); *// напечатает [Hi, Hello x 2, World! x 2, All!] - в произвольном порядке*

*// Выводим все уникальные слова*

System.out.println(multiset.elementSet()); *// напечатает [Hi, Hello, World!, All!] - в произвольном порядке*

*// Выводим количество по каждому слову*

System.out.println("Hello = " + multiset.count("Hello")); *// напечатает 2*

System.out.println("Empty = " + multiset.count("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(multiset.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiset.elementSet().size()); *//напечатает 4*

Используем TreeMultiset от guava для подсчета слов

Обратите внимание, что порядок вывода для TreeMultiset в System.out.println(multiset) и в  
System.out.println(multiset.elementSet()) — натуральный, то есть слова отсортированы по алфавиту.

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

Multiset<String> multiset = TreeMultiset.create(Arrays.asList(INPUT\_TEXT.split(" ")));

*// Выводим кол-вом вхождений слов*

System.out.println(multiset); *// напечатает [All!, Hello x 2, Hi, World! x 2]- в алфавитном порядке*

*// Выводим все уникальные слова*

System.out.println(multiset.elementSet()); *// напечатает [All!, Hello, Hi, World!]- в алфавитном порядке*

*// Выводим количество по каждому слову*

System.out.println("Hello = " + multiset.count("Hello")); *// напечатает 2*

System.out.println("Empty = " + multiset.count("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(multiset.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiset.elementSet().size()); *//напечатает 4*

Используем LinkedHashMultisetTest от guava для подсчета слов

Обратите внимание, что порядок вывода для LinkedHashMultiset в System.out.println(multiset) и в System.out.println(multiset.elementSet()) — в порядке первого добавления элемента

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

Multiset<String> multiset = LinkedHashMultiset.create(Arrays.asList(INPUT\_TEXT.split(" ")));

*// Выводим кол-вом вхождений слов*

System.out.println(multiset); *// напечатает [Hello x 2, World! x 2, All!, Hi]- в порядке первого*

добавления элемента

*// Выводим все уникальные слова*

System.out.println(multiset.elementSet()); *// напечатает [Hello, World!, All!, Hi] - в порядке первого*

добавления элемента

*// Выводим количество по каждому слову*

System.out.println("Hello = " + multiset.count("Hello")); *// напечатает 2*

System.out.println("Empty = " + multiset.count("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(multiset.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiset.elementSet().size()); *//напечатает 4*

Используем ConcurrentHashMultiset от guava для подсчета слов

Обратите внимание, что порядок вывода в System.out.println(multiset) и в  
System.out.println(multiset.elementSet()) — произвольный, то есть не определен, так как это по сути многопоточная версия HashMultiset

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

Multiset<String> multiset = ConcurrentHashMultiset.create(Arrays.asList(INPUT\_TEXT.split(" ")));

*// Выводим кол-вом вхождений слов*

System.out.println(multiset); *// напечатает [Hi, Hello x 2, World! x 2, All!] - в произвольном порядке*

*// Выводим все уникальные слова*

System.out.println(multiset.elementSet()); *// напечатает [Hi, Hello, World!, All!] - в произвольном*

порядке

*// Выводим количество по каждому слову*

System.out.println("Hello = " + multiset.count("Hello")); *// напечатает 2*

System.out.println("Empty = " + multiset.count("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(multiset.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiset.elementSet().size()); *//напечатает 4*

**2. разных вариантов Bag от Apache Commons Collections:**

Использование HashBag из Apache Commons Collections

Обратите внимание, что порядок вывода в System.out.println(multiset) и в  
System.out.println(multiset.elementSet()) — произвольный, то есть не определен.

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

Bag bag = new HashBag(Arrays.asList(INPUT\_TEXT.split(" ")));

*// Выводим кол-вом вхождений слов*

System.out.println(bag); *// напечатает [1:Hi,2:Hello,2:World!,1:All!] - в произвольном порядке*

*// Выводим все уникальные слова*

System.out.println(bag.uniqueSet()); *// напечатает [Hi, Hello, World!, All!] - в произвольном порядке*

*// Выводим количество по каждому слову*

System.out.println("Hello = " + bag.getCount("Hello")); *// напечатает 2*

System.out.println("Empty = " + bag.getCount("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(bag.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(bag.uniqueSet().size()); *//напечатает 4*

Использование TreeBag из Apache Commons Collections

Обратите внимание, что порядок вывода в System.out.println(multiset) и в  
System.out.println(multiset.elementSet()) — натуральный, то есть слова отсортированы по алфавиту.

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

Bag bag = new TreeBag(Arrays.asList(INPUT\_TEXT.split(" ")));

*// Выводим кол-вом вхождений слов*

System.out.println(bag); *// напечатает [1:All!,2:Hello,1:Hi,2:World!]- в алфавитном порядке*

*// Выводим все уникальные слова*

System.out.println(bag.uniqueSet()); *// напечатает [All!, Hello, Hi, World!]- в алфавитном порядке*

*// Выводим количество по каждому слову*

System.out.println("Hello = " + bag.getCount("Hello")); *// напечатает 2*

System.out.println("Empty = " + bag.getCount("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(bag.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(bag.uniqueSet().size()); *//напечатает 4*

Использование SynchronizedBag из Apache Commons Collections

Обратите внимание, что порядок вывода в System.out.println(multiset) и в  
System.out.println(multiset.elementSet()) — произвольный, то есть не определен, так как это по сути  
многопоточная версия HashBag

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

Bag bag = **S**ynchronizedBag.**s**ynchronizedBag(new HashBag(Arrays.asList(INPUT\_TEXT.split(" "))));

*// Выводим кол-вом вхождений слов*

System.out.println(bag); *// напечатает [1:Hi,2:Hello,2:World!,1:All!] - в произвольном порядке*

*// Выводим все уникальные слова*

System.out.println(bag.uniqueSet()); *// напечатает [Hi, Hello, World!, All!] - в произвольном порядке*

*// Выводим количество по каждому слову*

System.out.println("Hello = " + bag.getCount("Hello")); *// напечатает 2*

System.out.println("Empty = " + bag.getCount("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(bag.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(bag.uniqueSet().size()); *//напечатает 4*

Использование SynchronizedSortedBag из Apache Commons Collections

Обратите внимание, что порядок вывода в System.out.println(multiset) и в  
System.out.println(multiset.elementSet()) — натуральный, то есть слова отсортированы по алфавиту, так как  
это по сути многопоточная версия SortedBag

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

Bag bag = SynchronizedSortedBag.synchronizedBag(new TreeBag(Arrays.asList(INPUT\_TEXT.split(" "))));

*// Выводим кол-вом вхождений слов*

System.out.println(bag); *// напечатает [1:All!,2:Hello,1:Hi,2:World!]- в алфавитном порядке*

*// Выводим все уникальные слова*

System.out.println(bag.uniqueSet()); *// напечатает [All!, Hello, Hi, World!]- в алфавитном порядке*

*// Выводим количество по каждому слову*

System.out.println("Hello = " + bag.getCount("Hello")); *// напечатает 2*

System.out.println("Empty = " + bag.getCount("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(bag.size()); *// напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(bag.uniqueSet().size()); *// напечатает 4*

**3. разных вариантов Bag от GS Collections:**

Использование MutableBag из GS Collections

Обратите внимание, что порядок вывода в System.out.println(bag) и в System.out.println(bag.toSet()) — не определен

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

MutableBag<String> bag = HashBag.newBag(Arrays.asList(INPUT\_TEXT.split(" ")));

*// Выводим кол-вом вхождений слов*

System.out.println(bag); *// напечатает [Hi, World!, World!, Hello, Hello, All!]- в произвольном порядке*

*// Выводим все уникальные слова*

System.out.println(bag.toSet()); *// напечатает [Hi, Hello, World!, All!] - в произвольном порядке*

*// Выводим количество по каждому слову*

System.out.println("Hello = " + bag.occurrencesOf("Hello")); *// напечатает 2*

System.out.println("Empty = " + bag.occurrencesOf("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(bag.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(bag.toSet().size()); *//напечатает 4*

Использование MutableSortedBag из GS Collections

Обратите внимание, что порядок вывода в System.out.println(bag) и в System.out.println(bag.toSortedSet()) — будет натуральным, т.е. по алфавиту в данном случае

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Создаем Multiset*

MutableSortedBag<String> bag = TreeBag.newBag(Arrays.asList(INPUT\_TEXT.split(" ")));

*// Выводим кол-вом вхождений слов*

System.out.println(bag); *// напечатает [All!, Hello, Hello, Hi, World!, World!]- в натуральном порядке*

*// Выводим все уникальные слова*

System.out.println(bag.toSortedSet()); *// напечатает [All!, Hello, Hi, World!]- в натуральном порядке*

*// Выводим количество по каждому слову*

System.out.println("Hello = " + bag.occurrencesOf("Hello")); *// напечатает 2*

System.out.println("Empty = " + bag.occurrencesOf("Empty")); *// напечатает 0*

*// Выводим общее количества всех слов в тексте*

System.out.println(bag.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(bag.toSet().size()); *//напечатает 4*

**4. Ну и наконец, посмотрим как можно сделать тоже самое в чистом JDK с помощью эмуляции multiSet через HashMap**

Эмуляция multiSet через HashMap

Как вы легко можете заметить, кода потребовалось, естественно, больше чем у любых реализаций multiSet или  
Bag.

*// Разберем текст на слова*

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

List<String> listResult = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем эмуляцию Multiset c помощью HashMap и заполняем*

Map<String, Integer> fakeMultiset = new HashMap<String,Integer>(listResult.size());

for(String word: listResult) {

Integer cnt = fakeMultiset.get(word);

fakeMultiset.put(word, cnt == null ? 1 : cnt + 1);

}

*// Выводим кол-вом вхождений слов*

System.out.println(fakeMultiset); *// напечатает {World!=2, Hi=1, Hello=2, All!=1}- в произвольном*

порядке

*// Выводим все уникальные слова*

System.out.println(fakeMultiset.keySet()); *// напечатает [World!, Hi, Hello, All!] - в произвольном порядке*

*// Выводим количество по каждому слову*

System.out.println("Hello = " + fakeMultiset.get("Hello")); *// напечатает 2*

System.out.println("Empty = " + fakeMultiset.get("Empty")); *// напечатает null*

*// Выводим общее количества всех слов в тексте*

Integer cnt = 0;

for (Integer wordCount : fakeMultiset.values()){

cnt += wordCount;

}

System.out.println(cnt); *//напечатает 6*

*// Выводим общее количество уникальных слов*

System.out.println(fakeMultiset.size()); *//напечатает 4*

**4.2 Реализация Multimap в библиотеках guava, Apache Commons Collections и GS Collections**

Итак, Multimap это map, у которой у каждого ключа есть набор значений. Давайте сравним какие реализации есть данной  
коллекции в разных библиотеках. В таблице ниже порядок ключей и порядок значений показывает как будет происходить  
итерирования по ключам и значениям соответственно, дубликаты — может ли коллекция значений содержать дубликаты,  
аналог ключей и значений — на каких коллекциях построены ключи и значения, JDK показывает аналог коллекции с помощью  
JDK коллекций.  
  
**Внимание**: если таблица не помещается целиком, попробуйте уменьшить масштаб страницы или открыть в другом браузере.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Порядок ключей** | **Порядок значений** | **Дуб- лика- ты** | **Аналог ключей** | **Аналог значе- ний** | **Guava** | **Apache Commons Collections** | **GS Collections** | **JDK** |
| не задан | в порядке добавле- ния | да | HashMap | ArrayList | ArrayList- Multimap | MultiValueMap | FastList- Multimap | HashMap<K, ArrayList<V>> |
| не задан | не задан | нет | HashMap | HashSet | HashMultimap | MultiValueMap. multiValueMap( new HashMap<K, Set>(), HashSet.class); | UnifiedSet- Multimap | HashMap<K, HashSet<V>> |
| не задан | отсорти- рован | нет | HashMap | TreeSet | Multimaps. newMultimap( HashMap, Supplier <TreeSet>) | MultiValueMap. multiValueMap( new HashMap<K, Set>(), TreeSet.class); | TreeSortedSet- Multimap | HashMap<K, TreeSet<V>> |
| в порядке добавле- ния | в порядке добавле- ния | да | Linked HashMap | ArrayList | LinkedList- Multimap | MultiValueMap. multiValueMap(new LinkedHashMap<K, List>(), ArrayList.class); |  | LinkedHashMap< K, ArrayList<V>> |
| в порядке добавле- ния | в порядке добавле- ния | нет | LinkedHash- Multimap | Linked- HashSet | LinkedHash- Multimap | MultiValueMap. multiValueMap(new LinkedHashMap<K, Set>(), LinkedHashSet.class); |  | LinkedHashMap<K, LinkedHashSet<V>> |
| отсорти- рован | отсорти- рован | нет | TreeMap | TreeSet | TreeMultimap | MultiValueMap. multiValueMap( new TreeMap<K, Set>(),TreeSet.class); |  | TreeMap<K, TreeSet<V>> |

Как видно из таблицы, в Apache Commons Collections есть лишь одна реализация данного вида коллекции, остальные можно получить оборачивая стандартные коллекции при создании.

В guava намного больше уже определенных коллекций, при этоместь возможность реализовать обертку над любыми map'ами и любыми коллекциями значений.

В GS Collections есть так же специальные коллекция multimap основанная на Bag (HashBagMultimap), см. multiset и multimap.

**Примеры использования Multimap для сохранения всех вхождений слов в тексте**

**Есть задача**: дана строчка текста «Hello World! Hello All! Hi World!», нужно разобрать её на отдельные  
слова где разделитель только пробел и теперь нам нужно знать не только сколько каждых слов в тексте, но и все  
индекс вхождения слова в тесте, то есть что Hello это первое и третье слово в тексте и т.д.  
  
Посмотрим как это сделать с помощью  
**1. разных вариантов Multimap от Guava:**

Используем HashMultimap от guava

Обратите внимание, что порядок хранения данных произвольный как для ключей, так и для значений (для ключей поведение аналогичное HashMap, для значений HashSet). Повторяющиеся значения для одного ключа игнорируются.

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

Multimap<String, Integer> multiMap = HashMultimap.create();

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hi=[4], Hello=[0, 2], World!=[1, 5], All!=[3]} - в произвольном порядке*

*// Выводим все уникальные слова*

System.out.println(multiMap.keySet()); *// напечатает [Hi, Hello, World!, All!] - в произвольном порядке*

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает []*

*// Выводим общее количества всех слов в тексте*

System.out.println(multiMap.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiMap.keySet().size()); *//напечатает 4*

Используем ArrayListMultimapTest от guava

Обратите внимание, что порядок хранения данных произвольный для ключей, так и в порядке добавления для  
значений (для ключей поведение аналогичное HashMap, для значений ArrayList). Повторяющиеся значения для  
одного ключа сохраняются.

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

Multimap<String, Integer> multiMap = ArrayListMultimap.create();

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hi=[4], Hello=[0, 2], World!=[1, 5], All!=[3]} - ключи в*

произвольном порядке, значения в порядке добавления

*// Выводим все уникальные слова*

System.out.println(multiMap.keySet()); *// напечатает [Hello, World!, All!, Hi]- в произвольном порядке*

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает []*

*// Выводим общее количества всех слов в тексте*

System.out.println(multiMap.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiMap.keySet().size()); *//напечатает 4*

Используем LinkedHashMultimapTest от guava

Обратите внимание, что порядок хранения данных в порядке добавления для ключей и для значений (для ключей  
поведение аналогичное LinkedHashMap, для значений LinkedHashSet). Повторяющиеся значения для одного ключа  
игнорируются.

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List

<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

Multimap<String, Integer> multiMap = LinkedHashMultimap.create();

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hello=[0, 2], World!=[1, 5], All!=[3], Hi=[4]}-в порядке*

добавления

*// Выводим все уникальные слова*

System.out.println(multiMap.keySet()); *// напечатает [Hello, World!, All!, Hi]- в порядке добавления*

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает []*

*// Выводим общее количества всех слов в тексте*

System.out.println(multiMap.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiMap.keySet().size()); *//напечатает 4*

Используем LinkedListMultimapTest от guava

Обратите внимание, что порядок хранения данных в порядке добавления для ключей и для значений (для ключей  
поведение аналогичное LinkedHashMap, для значений LinkedList). Повторяющиеся значения для одного ключа  
сохраняются.

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List

<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

Multimap<String, Integer> multiMap = LinkedListMultimap.create();

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hello=[0, 2], World!=[1, 5], All!=[3], Hi=[4]}-в порядке*

добавления

*// Выводим все уникальные слова*

System.out.println(multiMap.keySet()); *// напечатает [Hello, World!, All!, Hi]- в порядке добавления*

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает []*

*// Выводим общее количества всех слов в тексте*

System.out.println(multiMap.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiMap.keySet().size()); *//напечатает 4*

Используем TreeMultimapTest от guava

Обратите внимание, что порядок хранения данных отсортированный для ключей и для значений (для ключей  
поведение аналогичное TreeMap, для значений TreeSet).

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

Multimap<String, Integer> multiMap = TreeMultimap.create();

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hello=[0, 2], World!=[1, 5], All!=[3], Hi=[4]}-в*

натуральном порядке

*// Выводим все уникальные слова*

System.out.println(multiMap.keySet()); *// напечатает [Hello, World!, All!, Hi]- в натуральном порядке*

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает []*

*// Выводим общее количества всех слов в тексте*

System.out.println(multiMap.size()); *//напечатает 6*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiMap.keySet().size()); *//напечатает 4*

**2. разных вариантов MultiValueMap от Apache Commons Collections:**

Используем MultiValueMap от Apache Commons Collections

Обратите внимание, что порядок хранения данных произвольный для ключей и значений (используется HashMap для  
ключей и ArrayList для значений)

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

MultiMap<String, Integer> multiMap = new MultiValueMap<String, Integer>();

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hi=[4], Hello=[0, 2], World!=[1, 5], All!=[3]} - в*

произвольном порядке

*// Выводим все уникальные слова*

System.out.println(multiMap.keySet()); *// напечатает [Hi, Hello, World!, All!] - в произвольном порядке*

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает null*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiMap.keySet().size()); *//напечатает 4*

Используем MultiValueMap, оборачивающий TreeMap<String, TreeSet>()

Обратите внимание, что порядок хранения данных отсортированный для ключей и значений

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

MultiMap<String, Integer> multiMap = MultiValueMap.multiValueMap(new TreeMap<String, Set>(), TreeSet.class);

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {All!=[3], Hello=[0, 2], Hi=[4], World!=[1, 5]} -в*

натуральном порядке

*// Выводим все уникальные слова*

System.out.println(multiMap.keySet()); *// напечатает [All!, Hello, Hi, World!] в натуральном порядке*

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает null*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiMap.keySet().size()); *//напечатает 4*

Используем MultiValueMap, оборачивающий LinkedHashMap<String, LinkedHashSet>()

Обратите внимание, что порядок хранения данных по первому добавлению для ключей и значений

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

MultiMap<String, Integer> multiMap = MultiValueMap.multiValueMap(new LinkedHashMap<String, Set>(), LinkedHashSet.class);

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hello=[0, 2], World!=[1, 5], All!=[3], Hi=[4]} - в порядке*

добавления

*// Выводим все уникальные слова*

System.out.println(multiMap.keySet()); *// напечатает [Hello, World!, All!, Hi] - в порядке добавления*

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает null*

*// Выводим общее количество всех уникальных слов*

System.out.println(multiMap.keySet().size()); *//напечатает 4*

**3. разных вариантов Multimap от GS Collections:**

Использование FastListMultimap

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

MutableListMultimap<String, Integer> multiMap = new FastListMultimap<String, Integer>();

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hi=[4], World!=[1, 5], Hello=[0, 2], All!=[3]}- в*

произвольном порядке

*// Выводим все уникальные слова*

System.out.println(multiMap.keysView()); *// напечатает [Hi, Hello, World!, All!] - в произвольном*

порядке

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает []*

*// Выводим общее количество всех слов в тексте*

System.out.println(multiMap.size()); *//напечатает 6*

*// Выводим общее количество уникальных слов в тексте*

System.out.println(multiMap.keysView().size()); *//напечатает 4*

Использование HashBagMultimap

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

MutableBagMultimap<String, Integer> multiMap = new HashBagMultimap<String, Integer>();

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hi=[4], World!=[1, 5], Hello=[0, 2], All!=[3]}- в*

произвольном порядке

*// Выводим все уникальные слова*

System.out.println(multiMap.keysView()); *// напечатает [Hi, Hello, World!, All!] - в произвольном*

порядке

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает []*

*// Выводим общее количество всех слов в тексте*

System.out.println(multiMap.size()); *//напечатает 6*

*// Выводим общее количество уникальных слов в тексте*

System.out.println(multiMap.keysView().size()); *//напечатает 4*

Использование TreeSortedSetMultimap

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

MutableSortedSetMultimap<String, Integer> multiMap = new TreeSortedSetMultimap<String, Integer>();

*// Заполним Multimap*

int i = 0;

for(String word: words) {

multiMap.put(word, i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(multiMap); *// напечатает {Hi=[4], World!=[1, 5], Hello=[0, 2], All!=[3]}- в*

произвольном порядке

*// Выводим все уникальные слова*

System.out.println(multiMap.keysView()); *// напечатает [Hi, Hello, World!, All!] - в произвольном*

порядке

*// Выводим все индексы вхождения слова в текст*

System.out.println("Hello = " + multiMap.get("Hello")); *// напечатает [0, 2]*

System.out.println("World = " + multiMap.get("World!")); *// напечатает [1, 5]*

System.out.println("Empty = " + multiMap.get("Empty")); *// напечатает []*

*// Выводим общее количество всех слов в тексте*

System.out.println(multiMap.size()); *//напечатает 6*

*// Выводим общее количество уникальных слов в тексте*

System.out.println(multiMap.keysView().size()); *//напечатает 4*

**4. Ну и наконец, посмотрим как можно сделать тоже самое в чистом JDK с помощью эмуляции multiMap через HashMap**

Эмуляция multiMap через HashMap

final int LIST\_INDEXES\_CAPACITY = 50;

String INPUT\_TEXT = "Hello World! Hello All! Hi World!";

*// Разберем текст на слова и индексы*

List<String> words = Arrays.asList(INPUT\_TEXT.split(" "));

*// Создаем Multimap*

Map<String, List<Integer>> fakeMultiMap = new HashMap<String, List<Integer>>(words.size());

*// Заполним map*

int i = 0;

for(String word: words) {

List<Integer> indexes = fakeMultiMap.get(word);

if(indexes == null) {

indexes = new ArrayList<Integer>(LIST\_INDEXES\_CAPACITY);

fakeMultiMap.put(word, indexes);

}

indexes.add(i);

i++;

}

*// Выводим все вхождения слов в текст*

System.out.println(fakeMultiMap); *// напечатает {Hi=[4], Hello=[0, 2], World!=[1, 5],*

All!=[3]} - в произвольном порядке

*// Выводим все уникальные слова*

System.out.println(fakeMultiMap.keySet()); *// напечатает [Hi, Hello, World!, All!] - в*

произвольном порядке

*// Выводим все индексы вхождения слова в текст*System.out.println("Hello = " + fakeMultiMap.get("Hello"));*//напечатает [0, 2]*System.out.println("World = " +fakeMultiMap.get("World!"));*//напечатает [1, 5]* System.out.println("Empty = " + fakeMultiMap.get("Empty")); *// напечатает null*

*// Выводим общее количество всех слов в тексте*

int cnt = 0;

for(List<Integer> lists: fakeMultiMap.values()) {

cnt += lists.size();

}

System.out.println(cnt); *//напечатает 6*

*// Выводим общее количество уникальных слов в тексте*

System.out.println(fakeMultiMap.keySet().size()); *//напечатает 4*

**4.3 Реализация BiMap в библиотеках guava, Apache Commons Collections ,GS Collections и JDK**

Реализация BiMap во всех библиотеках достаточно похожа, за исключением названия HashBiMap в guava и GS Collections и  
BidiMap в Apache Commons Collections. Кроме простейшей HashBiMap, у guava есть отдельные коллекции для работы с Enum  
в качестве ключей или значений, такие как EnumHashBiMap или EnumBiMap, у Apache Commons Collections есть ряд  
коллекций, где ключи упорядочены по добавлению или отсортированы.

**Примеры использования BiMap для создания русско-английского «переводчика», действующего в обе стороны**

**Есть задача**: есть массивы английских и русских слов соответствующие друг другу, нужно реализовать  
коллекцию русско-английского словаря, с возможностью перевода в обе стороны.  
  
Посмотрим как это сделать с помощью  
**1. разных вариантов BiMap от Guava:**

Используем BiMap от guava

String[] englishWords = {"one", "two", "three", "ball", "snow"};

String[] russianWords = {"один", "два", "три", "мяч", "cнег"};

*// Создаем Multiset*

BiMap<String, String> biMap = HashBiMap.create(englishWords.length);

*// создаем англо-русский словарь*

int i = 0;

for(String englishWord: englishWords) {

biMap.put(englishWord, russianWords[i]);

i++;

}

*// Выводим кол-вом вхождений слов*

System.out.println(biMap); *// напечатает {two=два, three=три, snow=cнег, ball=мяч, one=один} - в*

произвольном порядке

*// Выводим все уникальные слова*

System.out.println(biMap.keySet()); *// напечатает [two, three, snow, ball, one] - в произвольном порядке*

System.out.println(biMap.values()); *// напечатает [два, три, cнег, мяч, один]- в произвольном порядке*

*// Выводим перевод по каждому слову*

System.out.println("one = " + biMap.get("one")); *// напечатает one = один*

*// Выводим общее количество переводов в словаре*

System.out.println(biMap.size()); *//напечатает 5*

Используем EnumBiMap от guava

enum ENGLISH\_WORD {

ONE, TWO, THREE, BALL, SNOW

}

enum POLISH\_WORD {

JEDEN, DWA, TRZY, KULA, SNIEG

}

***// Задача даны массивы польско-английского перевода, сделать коллекцию для перевода слова в двух направлениях***

public static void main(String[] args) {

ENGLISH\_WORD[] englishWords = ENGLISH\_WORD.values();

POLISH\_WORD[] polishWords = POLISH\_WORD.values();

*// Создаем Multiset*

BiMap<ENGLISH\_WORD, POLISH\_WORD> biMap = EnumBiMap.create(ENGLISH\_WORD.class, POLISH\_WORD.class);

*// создаем англо-польский словарь*

int i = 0;

for(ENGLISH\_WORD englishWord: englishWords) {

biMap.put(englishWord, polishWords[i]);

i++;

}

*// Выводим кол-вом вхождений слов*

System.out.println(biMap); *// напечатает {ONE=JEDEN, TWO=DWA, THREE=TRZY, BALL=KULA, SNOW=SNIEG}*

*// Выводим все уникальные слова*

System.out.println(biMap.keySet()); *// напечатает [ONE, TWO, THREE, BALL, SNOW]*

System.out.println(biMap.values()); *// напечатает [JEDEN, DWA, TRZY, KULA, SNIEG]*

*// Выводим перевод по каждому слову*

System.out.println("one = " +biMap.get(ENGLISH\_WORD.ONE));*//напеч. one = JEDEN*

System.out.println("kula = " + biMap.inverse().get(POLISH\_WORD.KULA)); *// напечатает kula = BALL*

System.out.println("empty = " + biMap.get("empty")); *// напеч. empty = null*

*// Выводим общее количество переводов в словаре*

System.out.println(biMap.size()); *//напечатает 5*

}

Используем EnumHashBiMap от guava

enum ENGLISH\_WORD {

ONE, TWO, THREE, BALL, SNOW

}

*// Задача даны массивы руссков-английского перевода, сделать коллекцию для перевода слова в двух*

напрявлениях

public static void main(String[] args) {

ENGLISH\_WORD[] englishWords = ENGLISH\_WORD.values();

String[] russianWords = {"один", "два", "три","мяч","cнег"};

*// Создаем Multiset*

BiMap<ENGLISH\_WORD, String> biMap = EnumHashBiMap.create(ENGLISH\_WORD.class);

*// создаем англо-русский словарь*

int i = 0;

for(ENGLISH\_WORD englishWord: englishWords) {

biMap.put(englishWord, russianWords[i]);

i++;

}

*// Выводим кол-вом вхождений слов*

System.out.println(biMap); *// напечатает {ONE=один, TWO=два, THREE=три, BALL=мяч, SNOW=cнег}*

*// Выводим все уникальные слова*

System.out.println(biMap.keySet()); *// напечатает [ONE, TWO, THREE, BALL, SNOW]*

System.out.println(biMap.values()); *// напечатает [один, два, три, мяч, cнег]*

*// Выводим перевод по каждому слову*

System.out.println("one = " + biMap.get(ENGLISH\_WORD.ONE));*//напеч. one = один*

System.out.println("мяч = " + biMap.inverse().get("мяч")); *//напеч. мяч = BALL*

System.out.println("empty = " + biMap.get("empty")); *// напеч. empty = null*

*// Выводим общее количество переводов в словаре*

System.out.println(biMap.size()); *//напечатает 5*

}

**2. C помощью BidiMap от Apache Commons Collections:**

Используем DualHashBidiMap от Apache Commons Collections

String[] englishWords = {"one", "two", "three","ball","snow"};

String[] russianWords = {"один", "два", "три","мяч","cнег"};

*// Создаем Multiset*

BidiMap<String, String> biMap = new DualHashBidiMap();

*// создаем англо-русский словарь*

int i = 0;

for(String englishWord: englishWords) {

biMap.put(englishWord, russianWords[i]);

i++;

}

*// Выводим кол-вом вхождений слов*

System.out.println(biMap); *// напечатает {ball=мяч, snow=cнег, one=один, two=два, three=три}- в произвольном*

порядке

*// Выводим все уникальные слова*

System.out.println(biMap.keySet()); *// напечатает [ball, snow, one, two, three]- в произвольном порядке*

System.out.println(biMap.values()); *// напечатает [мяч, cнег, один, два, три]- в произвольном порядке*

*// Выводим перевод по каждому слову*

System.out.println("one = " + biMap.get("one")); *// напечатает one = один*

System.out.println("мяч = " + biMap.getKey("мяч")); *// напечатает мяч = ball*

System.out.println("empty = " + biMap.get("empty")); *// напеч. empty = null*

*// Выводим общее количество переводов в словаре*

System.out.println(biMap.size()); *//напечатает 5*

*3. C помощью HashBiMap от GS Collections:*

Используем HashBiMap от GS Collections

String[] englishWords = {"one", "two", "three","ball","snow"};

String[] russianWords = {"один", "два", "три","мяч","cнег"};

*// Создаем Multiset*

MutableBiMap<String, String> biMap = new HashBiMap(englishWords.length);

*// создаем англо-русский словарь*

int i = 0;

for(String englishWord: englishWords) {

biMap.put(englishWord, russianWords[i]);

i++;

}

*// Выводим кол-вом вхождений слов*

System.out.println(biMap); *// напечатает {two=два, ball=мяч, one=один, snow=cнег, three=три} - в*

произвольном порядке

*// Выводим все уникальные слова*

System.out.println(biMap.keySet()); *// напечатает [snow, two, one, three, ball] - в произвольном порядке*

System.out.println(biMap.values()); *// напечатает [два, мяч, один, cнег, три] - в произвольном порядке*

*// Выводим перевод по каждому слову*

System.out.println("one = " + biMap.get("one")); *// напечатает one = один*

System.out.println("мяч = " +biMap.inverse().get("мяч"));*//напечатает мяч=ball*

System.out.println("empty = " + biMap.get("empty")); *//напечатает empty = null*

*// Выводим общее количество переводов в словаре*

System.out.println(biMap.size()); *//напечатает 5*

**4. Ну и наконец, посмотрим как можно сделать тоже самое в чистом JDK**

Используем две HashMap для эмуляции BiMap

String[] englishWords = {"one", "two", "three","ball","snow"};

String[] russianWords = {"один", "два", "три","мяч","cнег"};

*// Создаем аналог BiMap*

Map<String, String> biMapKeys = new HashMap(englishWords.length);

Map<String, String> biMapValues = new HashMap(russianWords.length);

*// создаем англо-русский словарь*

int i = 0;

for(String englishWord: englishWords) {

biMapKeys.put(englishWord, russianWords[i]);

biMapValues.put(russianWords[i], englishWord);

i++;

}

*// Выводим кол-вом вхождений слов*

System.out.println(biMapKeys); *// напечатает {ball=мяч, two=два, three=три, snow=cнег, one=один}- в*

произвольном порядке

*// Выводим все уникальные слова*

System.out.println(biMapKeys.keySet()); *// напечатает [ball, two, three, snow, one] - в произвольном порядке*

System.out.println(biMapValues.keySet()); *// напечатает [два, три, мяч, cнег, один] - в произвольном порядке*

*// Выводим перевод по каждому слову*

System.out.println("one = " + biMapKeys.get("one")); *// напечатает one = один*

System.out.println("мяч = " + biMapValues.get("мяч")); *//напечатает мяч = ball*

System.out.println("empty = " +biMapValues.get("empty"));*//напеч. empty = null*

*// Выводим общее количество переводов в словаре*

System.out.println(biMapKeys.size()); *//напечатает 5*