**V. Сравнение операций работы с коллекциями**

Давайте кратко посмотрим какие дополнительные методы, операции и алгоритмы предлагают альтернативные библиотеки по сравнению со стандартными возможностями JDK. Цель этого, естественно, не перечислить все возможные методы всех библиотек (это невозможно), а скорее дать краткое представление о философии и синтаксисе разных библиотек, чтобы каждый мог выбрать то что больше нравится именно ему.  
  
**Содержание**

5.1 Сравним создание коллекций с помощью методов различных библиотек в **JDK, guava, gs-collections**.

5.1.1) Создание списка (List)

5.1.2) Создание множества (set)

5.1.3) Создание Map

5.2 Сравним методы поиска из различных библиотек в **JDK, guava, apache, gs-collections**.

1) Найти количество вхождений объекта

2) Вернуть первый элемент коллекции

3) Вернуть последний элемент коллекции

4) Вернуть максимальный элемент

5) Вернуть минимальный элемент

6) вернуть единственный элемент коллекции

7) найти элемент в отсортированном списке

8) найти элемент в неотсортированной коллекции

9) выбрать все элементы по условию

5.3 Сравним методы сравнений, объединений и пересечений коллекций в **JDK, guava, apache, gs-collections**.

1) Проверить полное соответствие двух коллекций

2) Проверить наличие хотя бы одного общего элемента у двух коллекций

3) Найти все общие элементы (пересечение) у двух коллекций

4) Найти все элементы, которые есть в одной коллекции и нет в другой (difference)

5) Найти все различные элементы (symmetric difference) у двух коллекций

6) Получить объедение двух коллекций

5.4 Сравним методы изменения коллекции в **JDK, guava, apache, gs-collections**.

1) Сортировка коллекции

2) Удалить все элементы соответствующие условию

3) Удалить все элементы не соответствующие условию

4) Преобразовать все элементы коллекции

5) Изменить свойства каждого элемента коллекции

5.1 Сравним создание коллекций с помощью методов различных библиотек.

Guava и gs-collections предлагают создание коллекций через статические методы утилиты вместо использование new, давайте посмотрим насколько это удобнее обычного способа jdk.  
  
5.1.1) Создание списка (List)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **JDK** | **guava** | **gs-collections** |
| Создание пустого **списка** | new ArrayList<>() | Lists.newArrayList() | FastList.newList() |
| Создание **списка** из значений | Arrays.asList(«1», «2», «3») | Lists.newArrayList(«1», «2», «3») | FastList.newListWith(«1», «2», «3») |
| Создать **список** с  определенным capacity | new ArrayList<>(100) | Lists.newArrayListWithCapacity(100) | FastList.newList(100) |
| Создать **список** из любой коллекции | new ArrayList<>(collection) | Lists.newArrayList(collection) | FastList.newList(collection) |
| Создать **список** из любого Iterable | - | Lists.newArrayList(iterable) | FastList.newList(iterable) |
| Создать **список**  из Iterator'а | - | Lists.newArrayList(iterator) | - |
| Создать **список** из массива | Arrays.asList(array) | Lists.newArrayList(array) | FastList.newListWith(array) |
| Создать **список**  c помощью фабрики | — | — | FastList.newWithNValues(10, () -> «1») |

Примеры создания списка

*// Простое создание пустых коллекций*

List<String> emptyGuava = Lists.newArrayList(); *// c помощью guava*

List<String> emptyJDK = new ArrayList<>(); *// аналог JDK*

MutableList<String> emptyGS = FastList.newList(); *// c помощью gs*

*// Создать список ровно со 100 элементами*

List < String > exactly100 = Lists.newArrayListWithCapacity(100); *// c помощью guava*

List<String> exactly100JDK = new ArrayList<>(100); *// аналог JDK*

MutableList<String> empty100GS = FastList.newList(100); *// c помощью gs*

*// Создать список в котором ожидается около 100 элементов (чуть больше чем 100)*

List<String> approx100 = Lists.newArrayListWithExpectedSize(100); *// c помощью guava*

List<String> approx100JDK = new ArrayList<>(115); *// аналог JDK*

MutableList<String> approx100GS = FastList.newList(115); *// c помощью gs*

*// Создать список из заданных элементов*

List<String> withElements = Lists.newArrayList("alpha", "beta", "gamma"); *// c помощью guava*

List<String> withElementsJDK = Arrays.asList("alpha", "beta", "gamma"); *// аналог JDK*

MutableList<String> withElementsGS = FastList.newListWith("alpha", "beta", "gamma"); *// c помощью gs*

System.out.println(withElements);

System.out.println(withElementsJDK);

System.out.println(withElementsGS);

*// Создать список из любого объекта Iterable интерфейса (то есть любой коллекции)*

Collection<String> collection = new HashSet<>(3);

collection.add("1");

collection.add("2");

collection.add("3");

List<String> fromIterable = Lists.newArrayList(collection); *// c помощью guava*

List<String> fromIterableJDK = new ArrayList<>(collection); *// аналог JDK*

MutableList<String> fromIterableGS = FastList.newList(collection); *// c помощью gs*

System.out.println(fromIterable);

System.out.println(fromIterableJDK);

System.out.println(fromIterableGS);

*/\* обратите внимание у JDK необходим объект Collection интерфейса, у guava и gs достаточно Iterable \*/*

*// Создать список из Iterator'а*

Iterator<String> iterator = collection.iterator();

List<String> fromIterator = Lists.newArrayList(iterator); *// c помощью guava, аналога в JDK нет*

System.out.println(fromIterator);

*// Создать список из массива*

String[] array = {"4", "5", "6"};

List<String> fromArray = Lists.newArrayList(array); *// c помощью guava*

List<String> fromArrayJDK = Arrays.asList(array); *// аналог JDK*

MutableList<String> fromArrayGS = FastList.newListWith(array); *// c помощью gs*

System.out.println(fromArray);

System.out.println(fromArrayJDK);

System.out.println(fromArrayGS);

*// Создать список из c помощью фабрики*

MutableList<String> fromFabricGS = FastList.newWithNValues(10, () -> String.valueOf(Math.random())); *// c помощью gs*

System.out.println(fromFabricGS);

5.1.2) Создание множества (set)

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **JDK** | **guava** | **gs-collections** |
| Создание пустого  **множества** | new HashSet<>() | Sets.newHashSet() | UnifiedSet.newSet() |
| Создать **множество** из  заданных элементов | new HashSet<>(Arrays.asList(«alpha», «beta», «gamma»)) | Sets.newHashSet(«alpha», «beta», «gamma») | UnifiedSet.newSetWith(«alpha», «beta», «gamma») |
| Создать **множество** из любой коллекции | new HashSet<>(collection) | Sets.newHashSet(collection) | UnifiedSet.newSet(collection) |
| Создать **множество** из любого Iterable | - | Sets.newHashSet(iterable) | UnifiedSet.newSet(iterable) |
| Создать **множество** из Iterator'а | - | Sets.newHashSet(iterator); | - |
| Создать **множество** из массива | new HashSet<>(Arrays.asList(array)) | Sets.newHashSet(array) | UnifiedSet.newSetWith(array) |

Примеры создания множества

*// Простое создание пустых коллекций*

Set<String> emptyGuava = Sets.newHashSet(); *// c помощью guava*

Set<String> emptyJDK = new HashSet<>(); *// аналог JDK*

Set<String> emptyGS = UnifiedSet.newSet(); *// c помощью gs*

*// Создать множество в котором ожидается около 100 элементов (чуть больше чем 100)*

Set<String> approx100 = Sets.newHashSetWithExpectedSize(100); *// c помощью guava*

Set<String> approx100JDK = new HashSet<>(130); *// аналог JDK*

Set<String> approx100GS = UnifiedSet.newSet(130); *// c помощью gs*

*// Создать множество из заданных элементов*

Set<String> withElements = Sets.newHashSet("alpha", "beta", "gamma"); *// c помощью guava*

Set<String> withElementsJDK = new HashSet<>(Arrays.asList("alpha", "beta", "gamma")); *// аналог JDK*

Set<String> withElementsGS = UnifiedSet.newSetWith("alpha", "beta", "gamma"); *// c помощью gs*

System.out.println(withElements);

System.out.println(withElementsJDK);

System.out.println(withElementsGS);

*// Создать множество из любого объекта Iterable интерфейса (то есть любой коллекции)*

Collection<String> collection = new ArrayList<>(3);

collection.add("1");

collection.add("2");

collection.add("3");

Set<String> fromIterable = Sets.newHashSet(collection); *// c помощью guava*

Set<String> fromIterableJDK = new HashSet<>(collection); *// аналог JDK*

Set<String> fromIterableGS = UnifiedSet.newSet(collection); *// c помощью gs*

System.out.println(fromIterable);

System.out.println(fromIterableJDK);

System.out.println(fromIterableGS);

*/\* обратите внимание у JDK необходим объект Collection интерфейса, у guava достаточно Iterable \*/*

*// Создать множество из Iterator'а*

Iterator<String> iterator = collection.iterator();

Set<String> fromIterator = Sets.newHashSet(iterator); *// c помощью guava, аналога в JDK нет*

System.out.println(fromIterator);

*// Создать множество из массива*

String[] array = {"4", "5", "6"};

Set<String> fromArray = Sets.newHashSet(array); *// c помощью guava*

Set<String> fromArrayJDK = new HashSet<>(Arrays.asList(array)); *// аналог JDK*

Set<String> fromArrayGS = UnifiedSet.newSetWith(array); *// c помощью gs*

System.out.println(fromArray);

System.out.println(fromArrayJDK);

System.out.println(fromArrayGS);

5.1.3) Создание Map

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **JDK** | **guava** | **gs-collections** |
| Создание пустой **map'ы** | new HashMap<>() | Maps.newHashMap() | UnifiedMap.newMap() |
| Создать **map'у** c  определенным capacity | new HashMap<>(130) | Maps.newHashMapWithExpectedSize(100) | UnifiedMap.newMap(130) |
| Создать map'у из другой **map'ы** | new HashMap<>(map) | Maps.newHashMap(map) | UnifiedMap.newMap(map) |
| Создать **map'у** из ключей | - | - | UnifiedMap.newWithKeysValues(«1», «a», «2», «b») |

Примеры создания map

// Простое создание пустых коллекций  
Map<String, String> emptyGuava = Maps.newHashMap(); // c помощью guava  
Map<String, String> emptyJDK = new HashMap<>(); // аналог JDK  
Map<String, String> emptyGS = UnifiedMap.newMap(); // c помощью gs  
  
// Создать map'у в котором ожидается около 100 элементов (чуть больше чем 100)  
Map<String, String> approx100 = Maps.newHashMapWithExpectedSize(100); // c помощью guava  
Map<String, String> approx100JDK = new HashMap<>(130); // аналог JDK  
Map<String, String> approx100GS = UnifiedMap.newMap(130); // c помощью gs  
  
// Создать map'у из другой map'ы  
Map<String, String> map = new HashMap<>(3);  
map.put(«k1»,«v1»);  
map.put(«k2»,«v2»);  
Map<String, String> withMap = Maps.newHashMap(map); // c помощью guava  
Map<String, String> withMapJDK = new HashMap<>(map); // аналог JDK  
Map<String, String> withMapGS = UnifiedMap.newMap(map); // c помощью gs  
  
System.out.println(withMap);  
System.out.println(withMapJDK);  
System.out.println(withMapGS);  
  
// Создать map'у из ключей  
Map<String, String> withKeys = UnifiedMap.newWithKeysValues(«1», «a», «2», «b»);  
System.out.println(withKeys);

5.2 Сравним методы поиска из различных библиотек.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **JDK** | **guava** | **apache** | **gs-collections** |
| Найти количество вхождений объекта | Collections.frequency(collection, «1») | Iterables.frequency(iterable, «1») | CollectionUtils.cardinality(«1», iterable) | mutableCollection.count((each) -> «a1».equals(each)) |
| Вернуть первый элемент коллекции или значение по умолчанию | collection.stream().findFirst().orElse(«1») | Iterables.getFirst(iterable, «1») | CollectionUtils.get(iterable, 0) | orderedIterable.getFirst() |
| Вернуть последний элемент коллекции или значение по умолчанию | collection.stream().skip(collection.size()-1).findFirst().orElse(«1»); | Iterables.getLast(iterable, «1») | CollectionUtils.get(collection, collection.size()-1) | orderedIterable.getLast() |
| Вернуть максимальный элемент | Collections.max(collection) | Ordering.natural().max(iterable) | - | orderedIterable.max() |
| Вернуть минимальный элемент | Collections.min(collection) | Ordering.natural().min(iterable) | - | orderedIterable.min() |
| Вернуть единственный элемент коллекции |  | Iterables.getOnlyElement(iterable) | CollectionUtils.extractSingleton(collection) |  |
| Найти элемент в отсортированном списке | Collections.binarySearch(list, «13») | Ordering.natural().binarySearch(list, «13») |  | mutableList.binarySearch(«13») |
| Найти элемент в неотсортированной коллекции | collection.stream().filter(«13»::equals).findFirst().get() | Iterables.find(iterable, «13»::equals) | CollectionUtils.find(iterable, «13»::equals) | mutableList.select(«13»::equals).get(0) |
| Выбрать все элементы по условию | collection.stream().filter((s) -> s.contains(«1»)).collect(Collectors.toList()) | Iterables.filter(iterable, (s) -> s.contains(«1»)) | CollectionUtils.select(iterable, (s) -> s.contains(«1»)) | mutableCollection.select((s) -> s.contains(«1»)) |

Обратите внимание что методы разных библиотек работают с разными сущностями, это можно определить по названию переменных: collection — любая реализация интерфейса Collection, iterable — интерфейса Iterable, list — интерфейса List, orderedIterable и mutableList соответствующих интерфейсов в GS (orderedIterable — интерфейс для всех коллекций у которых определен порядок элементов, mutableList — интерфейс для любых изменяемых списков)  
  
**Примеры**:

1) Найти количество вхождений объекта

Collection<String> collection = Lists.newArrayList("a1", "a2", "a3", "a1");

Iterable<String> iterable = collection;

MutableCollection<String> collectionGS = FastList.newListWith("a1", "a2", "a3", "a1");

*// Вернуть количество вхождений объекта*

int i1 = Iterables.frequency(iterable, "a1"); *// с помощью guava*

int i2 = Collections.frequency(collection, "a1"); *// c помощью JDK*

int i3 = CollectionUtils.cardinality("a1", iterable); *// c помощью Apache*

int i4 = collectionGS.count((s) -> "a1".equals(s));

long i5 = collection.stream().filter((s) -> "a1".equals(s)).count(); *// c помощью stream JDK*

System.out.println("count = " + i1 + ":" + i2 + ":" + i3 + ":" + i4 + ":" + i5); *// напечатает count = 2:2:2:2:2*

2) Вернуть первый элемент коллекции

Collection<String> collection = Lists.newArrayList("a1", "a2", "a3", "a1");

OrderedIterable<String> orderedIterable = FastList.newListWith("a1", "a2", "a3", "a1");

Iterable<String> iterable = collection;

*// вернуть первый элемент коллекции*

Iterator<String> iterator = collection.iterator(); *// c помощью JDK*

String jdk = iterator.hasNext() ? iterator.next(): "1";

String guava = Iterables.getFirst(iterable, "1"); *// с помощью guava*

String apache = CollectionUtils.get(iterable, 0); *// c помощью Apache*

String gs = orderedIterable.getFirst(); *// c помощью GS*

String stream = collection.stream().findFirst().orElse("1"); *// c помощью Stream API*

System.out.println("first = " + jdk + ":" + guava + ":" + apache + ":" + gs + ":" + stream); *// напечатает first = a1:a1:a1:a1:a1*

3) Вернуть последний элемент коллекции

Collection<String> collection = Lists.newArrayList("a1", "a2", "a3", "a8");

OrderedIterable<String> orderedIterable = FastList.newListWith("a1", "a2", "a3", "a8");

Iterable<String> iterable = collection;

*// вернуть последней элемент коллекции*

Iterator<String> iterator = collection.iterator(); *// c помощью JDK*

String jdk = "1";

while(iterator.hasNext()) {

jdk = iterator.next();

}

String guava = Iterables.getLast(iterable, "1"); *// с помощью guava*

String apache = CollectionUtils.get(collection, collection.size()-1); *// c помощью Apache*

String gs = orderedIterable.getLast(); *// c помощью GS*

String stream = collection.stream().skip(collection.size()-1).findFirst().orElse("1"); *// c помощью Stream API*

System.out.println("last = " + jdk + ":" + guava + ":" + apache + ":" + gs + ":" + stream); *// напечатает last = a8:a8:a8:a8:a8*

4) Вернуть максимальный элемент

Collection<String> collection = Lists.newArrayList("5", "1", "3", "8", "4");

OrderedIterable<String> orderedIterable = FastList.newListWith("5", "1", "3", "8", "4");

Iterable<String> iterable = collection;

*// вернуть максимальный элемент коллекции*

String jdk = Collections.max(collection); *// c помощью JDK*

String gs = orderedIterable.max(); *// c помощью GS*

String guava = Ordering.natural().max(iterable); *// с помощью guava*

System.out.println("max = " + jdk + ":" + guava + ":" + gs); *// напечатает max = 8:8:8*

5) Вернуть минимальный элемент

Collection<String> collection = Lists.newArrayList("5", "1", "3", "8", "4");

OrderedIterable<String> orderedIterable = FastList.newListWith("5", "1", "3", "8", "4");

Iterable<String> iterable = collection;

*// вернуть минимальный элемент коллекции*

String jdk = Collections.min(collection); *// c помощью JDK*

String gs = orderedIterable.min(); *// c помощью GS*

String guava = Ordering.natural().min(iterable); *// с помощью guava*

System.out.println("min = " + jdk + ":" + guava + ":" + gs); *// напечатает min = 1:1:1*

6) вернуть единственный элемент коллекции

Collection<String> collection = Lists.newArrayList("a3");

OrderedIterable<String> orderedIterable = FastList.newListWith("a3");

Iterable<String> iterable = collection;

*// вернуть единственный элемент коллекции*

String guava = Iterables.getOnlyElement(iterable); *// с помощью guava*

String jdk = collection.iterator().next(); *// c помощью JDK*

String apache = CollectionUtils.extractSingleton(collection); *// c помощью Apache*

assert(orderedIterable.size() > 1);*// c помощью GS*

String gs = orderedIterable.getFirst();

System.out.println("single = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает single = a3:a3:a3:a3*

7) найти элемент в отсортированом списке

List<String> list = Lists.newArrayList("2", "4", "13", "31", "43");

MutableList<String> mutableList = FastList.newListWith("2", "4","13", "31", "43");

*// найти элемент в отсортированом списке*

int jdk = Collections.binarySearch(list, "13");

int guava = Ordering.natural().binarySearch(list, "13");

int gs = mutableList.binarySearch("13");

System.out.println("find = " + jdk + ":" + guava + ":" + gs); *// напечатает find = 2:2:2*

8) найти элемент в неотсортированной коллекции

Collection<String> collection = Lists.newArrayList("a1", "a2", "a3", "a1");

MutableCollection<String> orderedIterable = FastList.newListWith("a1", "a2", "a3", "a1");

Iterable<String> iterable = collection;

*// вернуть третий элемент коллекции по порядку*

String jdk = collection.stream().skip(2).findFirst().get(); *// c помощью JDK*

String guava = Iterables.get(iterable, 2); *// с помощью guava*

String apache = CollectionUtils.get(iterable, 2); *// c помощью Apache*

System.out.println("third = " + jdk + ":" + guava + ":" + apache); *// напечатает third = a3:a3:a3*

9) выбрать все элементы по условию

Collection<String> collection = Lists.newArrayList("2", "14", "3", "13", "43");

MutableCollection<String> mutableCollection = FastList.newListWith("2", "14", "3", "13", "43");

Iterable<String> iterable = collection;

*// выбрать все элементы по шаблону*

List<String> jdk = collection.stream().filter((s) -> s.contains("1")).collect(Collectors.toList()); *// c помощью JDK*

Iterable<String> guava = Iterables.filter(iterable, (s) -> s.contains("1")); *// с помощью guava*

Collection<String> apache = CollectionUtils.select(iterable, (s) -> s.contains("1")); *// c помощью Apache*

MutableCollection<String> gs = mutableCollection.select((s) -> s.contains("1")); *// c помощью GS*

System.out.println("select = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает select = [14, 13]:[14, 13]:[14, 13]:[14, 13]*

5.3 Сравним методы сравнений, объединений и пересечений коллекций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **JDK** | **guava** | **apache** | **gs-collections** |
| Проверить полное соответствие двух коллекций | collection1.containsAll(collection2) | Iterables.elementsEqual(iterable1, iterable2) | CollectionUtils.containsAll(collection1, collection2) | mutableCollection1.containsAll(mutableCollection2) |
| Наличие хотя бы одного общего элемента | !Collections.disjoint(collection1, collection2) | !Sets.intersection(set1, set2).isEmpty() | CollectionUtils.containsAny(collection1, collection2) | !mutableSet1.intersect(mutableSet2).isEmpty() |
| Найти все общие элементы (пересечение) | Set<T> result = new HashSet<>(set1);  result.retainAll(set2) | Sets.intersection(set1, set2) | CollectionUtils.intersection(collection1, collection2) | mutableSet1.intersect(mutableSet2) |
| Отсутствие общих элементов | Collections.disjoint(collection1, collection2) | Sets.intersection(set1, set2).isEmpty() | !CollectionUtils.containsAny(collection1, collection2) | mutableSet1.intersect(mutableSet2).isEmpty() |
| Найти все элементы, которые есть в одной коллекции и нет в другой (difference) | Set<T> result = new HashSet<>(set1); result.removeAll(set2) | Sets.difference(set1, set2) | CollectionUtils.removeAll(collection1, collection2) | mutableSet1.difference(mutableSet2) |
| Найти все различные элементы (symmetric difference) |  | Sets.symmetricDifference(set1, set2) | CollectionUtils.disjunction(collection1, collection2) | mutableSet1.symmetricDifference(mutableSet2) |
| Получить объедение двух коллекций | Set<T> result = new HashSet<>(set1);  result.addAll(set2) | Sets.union(set1, set2) | CollectionUtils.union(collection1, collection2) | mutableSet1.union(mutableSet2) |

**Примеры**:

1) Проверить полное соответствие двух коллекций

Collection<String> collection1 = Lists.newArrayList("a1", "a2", "a3", "a1");

Collection<String> collection2 = Lists.newArrayList("a1", "a2", "a3", "a1");

Iterable<String> iterable1 = collection1;

Iterable<String> iterable2 = collection2;

MutableCollection<String> mutableCollection1 = FastList.newListWith("a1", "a2", "a3", "a1");

MutableCollection<String> mutableCollection2 = FastList.newListWith("a1", "a2", "a3", "a1");

*// Проверить полное соответствие двух коллекций*

boolean jdk = collection1.containsAll(collection2); *// c помощью JDK*

boolean guava = Iterables.elementsEqual(iterable1, iterable2); *// с помощью guava*

boolean apache = CollectionUtils.containsAll(collection1, collection2); *// c помощью Apache*

boolean gs = mutableCollection1.containsAll(mutableCollection2); *// c помощью GS*

System.out.println("containsAll = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает containsAll = true:true:true:true*

2) Проверить наличие хотя бы одного общего элемента у двух коллекций

Collection<String> collection1 = Lists.newArrayList("a1", "a2", "a3", "a1");

Collection<String> collection2 = Lists.newArrayList("a4", "a8", "a3", "a5");

Set<String> set1 = Sets.newHashSet("a1", "a2", "a3", "a1");

Set<String> set2 = Sets.newHashSet("a4", "a8", "a3", "a5");

MutableSet<String> mutableSet1 = UnifiedSet.newSetWith("a1", "a2", "a3", "a1");

MutableSet<String> mutableSet2 = UnifiedSet.newSetWith("a4", "a8", "a3", "a5");

*// Проверить наличие хотя бы одного общего элемента у двух коллекций*

boolean jdk = !Collections.disjoint(collection1, collection2); *// c помощью JDK*

boolean guava = !Sets.intersection(set1, set2).isEmpty(); *// с помощью guava*

boolean apache = CollectionUtils.containsAny(collection1, collection2); *// c помощью Apache*

boolean gs = !mutableSet1.intersect(mutableSet2).isEmpty(); *// c помощью GS*

System.out.println("containsAny = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает containsAny = true:true:true:true*

3) Найти все общие элементы (пересечение) у двух коллекций

Collection<String> collection1 = Lists.newArrayList("a1", "a2", "a3", "a1");

Collection<String> collection2 = Lists.newArrayList("a4", "a8", "a3", "a5");

Set<String> set1 = Sets.newHashSet("a1", "a2", "a3", "a1");

Set<String> set2 = Sets.newHashSet("a4", "a8", "a3", "a5");

MutableSet<String> mutableSet1 = UnifiedSet.newSetWith("a1", "a2", "a3", "a1");

MutableSet<String> mutableSet2 = UnifiedSet.newSetWith("a4", "a8", "a3", "a5");

*// Найти все общие элементы у двух коллекций*

Set<String> jdk = new HashSet<>(set1); *// c помощью JDK*

jdk.retainAll(set2);

Set<String> guava = Sets.intersection(set1, set2); *// с помощью guava*

Collection<String> apache = CollectionUtils.intersection(collection1, collection2); *// c помощью Apache*

Set<String> gs = mutableSet1.intersect(mutableSet2); *// c помощью GS*

System.out.println("intersect = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает intersect = [a3]:[a3]:[a3]:[a3]*

4) Найти все элементы, которые есть в одной коллекции и нет в другой (difference)

Collection<String> collection1 = Lists.newArrayList("a2", "a3");

Collection<String> collection2 = Lists.newArrayList("a8", "a3", "a5");

Set<String> set1 = Sets.newHashSet("a2", "a3");

Set<String> set2 = Sets.newHashSet("a8", "a3", "a5");

MutableSet<String> mutableSet1 = UnifiedSet.newSetWith("a2", "a3");

MutableSet<String> mutableSet2 = UnifiedSet.newSetWith("a8", "a3", "a5");

*// Найти все элементы, которые есть в одной коллекции и нет в другой (difference)*

Set<String> jdk = new HashSet<>(set1); *// c помощью JDK*

jdk.removeAll(set2);

Set<String> guava = Sets.difference(set1, set2); *// с помощью guava*

Collection<String> apache = CollectionUtils.removeAll(collection1, collection2); *// c помощью Apache*

Set<String> gs = mutableSet1.difference(mutableSet2); *// c помощью GS*

System.out.println("difference = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает difference = [a2]:[a2]:[a2]:[a2]*

5) Найти все различные элементы (symmetric difference) у двух коллекций

Collection<String> collection1 = Lists.newArrayList("a2", "a3");

Collection<String> collection2 = Lists.newArrayList("a8", "a3", "a5");

Set<String> set1 = Sets.newHashSet("a2", "a3");

Set<String> set2 = Sets.newHashSet("a8", "a3", "a5");

MutableSet<String> mutableSet1 = UnifiedSet.newSetWith("a2", "a3");

MutableSet<String> mutableSet2 = UnifiedSet.newSetWith("a8", "a3", "a5");

*// Найти все различные элементы (symmetric difference) у двух коллекций*

Set<String> intersect = new HashSet<>(set1); *// c помощью JDK*

intersect.retainAll(set2);

Set<String> jdk = new HashSet<>(set1);

jdk.addAll(set2);

jdk.removeAll(intersect);

Set<String> guava = Sets.symmetricDifference(set1, set2); *// с помощью guava*

Collection<String> apache = CollectionUtils.disjunction(collection1, collection2); *// c помощью Apache*

Set<String> gs = mutableSet1.symmetricDifference(mutableSet2); *// c помощью GS*

System.out.println("symmetricDifference = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает symmetricDifference = [a2, a5, a8]:[a2, a5, a8]:[a2, a5, a8]:[a2, a5, a8]*

6) Получить объедение двух коллекций

Set<String> set1 = Sets.newHashSet("a1", "a2");

Set<String> set2 = Sets.newHashSet("a4");

MutableSet<String> mutableSet1 = UnifiedSet.newSetWith("a1", "a2");

MutableSet<String> mutableSet2 = UnifiedSet.newSetWith("a4");

Collection<String> collection1 = set1;

Collection<String> collection2 = set2;

*// Получить объедение двух коллекций*

Set<String> jdk = new HashSet<>(set1); *// c помощью JDK*

jdk.addAll(set2);

Set<String> guava = Sets.union(set1, set2); *// с помощью guava*

Collection<String> apache = CollectionUtils.union(collection1, collection2); *// c помощью Apache*

Set<String> gs = mutableSet1.union(mutableSet2); *// c помощью GS*

System.out.println("union = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает union = [a1, a2, a4]:[a1, a2, a4]:[a1, a2, a4]:[a1, a2, a4]*

5.4 Сравним методы изменения коллекции

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **JDK** | **guava** | **apache** | **gs-collections** |
| Сортировка коллекции | Collections.sort(list); | Ordering.natural().sortedCopy(iterable) |  | mutableList.sortThis() |
| Удалить все элементы соответствующие условию | collection.removeIf((s) -> s.contains(«1»)) | Iterables.removeIf(iterable, (s) -> s.contains(«1»)) | CollectionUtils.filter(iterable, (s) -> !s.contains(«1»)) | mutableCollection.removeIf((Predicate<String>) (s) -> s.contains(«1»)) |
| Удалить все элементы не соответствующие условию | collection.removeIf((s) -> !s.contains(«1»)) | Iterables.removeIf(iterable, (s) -> !s.contains(«1»)) | CollectionUtils.filter(iterable, (s) -> s.contains(«1»)) | mutableCollection.removeIf((Predicate<String>) (s) -> !s.contains(«1»)) |
| Изменить все элементы коллекции | collection.stream().map((s) -> s + "\_1").collect(Collectors.toList()) | Iterables.transform(iterable, (s) -> s + "\_1") | CollectionUtils.transform(collection, (s) -> s + "\_1") | mutableCollection.collect((s) -> s + "\_1") |
| Изменить свойства каждого элемента | collection.stream().forEach((s) -> s.append("\_1")) | Iterables.transform(iterable, (s) -> s.append("\_1")) | CollectionUtils.transform(collection, (s) -> s.append("\_1")) | mutableCollection.forEach((Procedure<StringBuilder>) (s) -> s.append("\_1")) |

1) Сортировка коллекции

List<String> jdk = Lists.newArrayList("a1", "a2", "a3", "a1");

Iterable<String> iterable = jdk;

MutableList<String> gs = FastList.newList(jdk);

*// Сортировка коллекции*

Collections.sort(jdk); *// с помощью jdk*

List<String> guava = Ordering.natural().sortedCopy(iterable); *// с помощью guava*

gs.sortThis(); *// c помощью gs*

System.out.println("sort = " + jdk + ":" + guava + ":" + gs); *// напечатает sort = [a1, a1, a2, a3]:[a1, a1, a2, a3]:[a1, a1, a2, a3]*

2) Удалить все элементы соответствующие условию

Collection<String> jdk = Lists.newArrayList("a1", "a2", "a3", "a1");

Iterable<String> guava = Lists.newArrayList(jdk);

Iterable<String> apache = Lists.newArrayList(jdk);

MutableCollection<String> gs = FastList.newList(jdk);

*// Удалить все элементы соответствующие условию*

jdk.removeIf((s) -> s.contains("1")); *// с помощью jdk*

Iterables.removeIf(guava, (s) -> s.contains("1")); *// с помощью guava*

CollectionUtils.filter(apache, (s) -> !s.contains("1")); *// с помощью apache*

gs.removeIf((Predicate<String>) (s) -> s.contains("1")); *// c помощью gs*

System.out.println("removeIf = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает removeIf = [a2, a3]:[a2, a3]:[a2, a3]:[a2, a3]*

3) Удалить все элементы не соответствующие условию

Collection<String> jdk = Lists.newArrayList("a1", "a2", "a3", "a1");

Iterable<String> guava = Lists.newArrayList(jdk);

Iterable<String> apache = Lists.newArrayList(jdk);

MutableCollection<String> gs = FastList.newList(jdk);

*// Удалить все элементы не соответствующие условию*

jdk.removeIf((s) -> !s.contains("1")); *// с помощью jdk*

Iterables.removeIf(guava, (s) -> !s.contains("1")); *// с помощью guava*

CollectionUtils.filter(apache, (s) -> s.contains("1")); *// с помощью apache*

gs.removeIf((Predicate<String>) (s) -> !s.contains("1")); *// c помощью gs*

System.out.println("retainIf = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает retainIf = [a1, a1]:[a1, a1]:[a1, a1]:[a1, a1]*

4) Преобразовать все элементы коллекции

Collection<String> collection = Lists.newArrayList("a1", "a2", "a3", "a1");

Iterable<String> iterable = collection;

Collection<String> apache = Lists.newArrayList(collection);

MutableCollection<String> mutableCollection = FastList.newList(collection);

*// Преобразовать все элементы коллекции в другие элементы*

List<String> jdk = collection.stream().map((s) -> s + "\_1").collect(Collectors.toList()); *// с помощью jdk*

Iterable<String> guava = Iterables.transform(iterable, (s) -> s + "\_1"); *// с помощью guava*

CollectionUtils.transform(apache, (s) -> s + "\_1"); *// с помощью apache*

MutableCollection<String> gs = mutableCollection.collect((s) -> s + "\_1"); *// c помощью gs*

System.out.println("transform = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// напечатает transform = [a1\_1, a2\_1, a3\_1, a1\_1]:[a1\_1, a2\_1, a3\_1, a1\_1]:[a1\_1, a2\_1, a3\_1, a1\_1]:[a1\_1, a2\_1, a3\_1, a1\_1]*

5) Изменить свойства каждого элемента коллекции

Collection<StringBuilder> jdk = Lists.newArrayList(new StringBuilder("a1"), new StringBuilder("a2"), new StringBuilder("a3"));

Iterable<StringBuilder> iterable = Lists.newArrayList(new StringBuilder("a1"), new StringBuilder("a2"), new StringBuilder("a3"));;

Collection<StringBuilder> apache = Lists.newArrayList(new StringBuilder("a1"), new StringBuilder("a2"), new StringBuilder("a3"));

MutableCollection<StringBuilder> gs = FastList.newListWith(new StringBuilder("a1"), new StringBuilder("a2"), new StringBuilder("a3"));

*// Изменить свойства каждого элемента коллекции*

jdk.stream().forEach((s) -> s.append("\_1")); *// с помощью jdk*

Iterable<StringBuilder> guava = Iterables.transform(iterable, (s) -> s.append("\_1")); *// с помощью guava*

CollectionUtils.transform(apache, (s) -> s.append("\_1")); *// с помощью apache*

gs.forEach((Procedure<StringBuilder>) (s) -> s.append("\_1")); *// c помощью gs*

System.out.println("change = " + jdk + ":" + guava + ":" + apache + ":" + gs); *// changeAll = [a1\_1, a2\_1, a3\_1]:[a1\_1, a2\_1, a3\_1]:[a1\_1, a2\_1, a3\_1]:[a1\_1, a2\_1, a3\_1]*