Java → Collections → Map

Java → Multiset

Hard 19 minutes Share:

115 users solved this problem. Latest completion was **2 days ago**.

Code Challenge — Write a program

This problem is a bit challenged. You can just skip it if you are a beginner or don't have enough time.

A **multiset** is a generalization of the concept of a set. Unlike sets, it can store duplicate elements. The number of instances of an element is the **multiplicity.**

For example, given the following multiset:

```
{a, a, b, b, b, c}
```

The multiplicity of **a** is 2, the multiplicity of **b** is 3, the multiplicity of **c** is 1. If a multiset does not have an element, the multiplicity of it is 0.

Write an implementation of the provided generic interface **Multiset**. The template for your generic class named **HashMultiset** is given as well. You should implement all methods of the class, according to its interface. You also can add additional methods for helping.

Read the given interface to understand the common multiset operations (add, remove, union, intersection and so one).

Do not forget to test your class. If your implementation is not correct, the testing system gives you a hint throwing an exception with a text, like:

```
Exception in thread "main" java.lang.AssertionError: size() returned an incorrect result
```

When you pass your solution to the submit form, do not remove the interface and do not make it and the class public.

Code Editor IDE

Java

```java
interface Multiset<E> {

    /**
     * Add an element to the multiset.
     * It increases the multiplicity of the element by 1.
     */
    void add(E elem);

    /**
     * Remove an element from the multiset.
     * It decreases the multiplicity of the element by 1.
     */
    void remove(E elem);

    /**
     * Union this multiset with another one. The result is the modified multiset (this).
     * It will contain all elements that are present in at least one of the initial multisets.
     * The multiplicity of each element is equal to the maximum multiplicity of
     * the corresponding elements in both multisets.
     */
    void union(Multiset<E> other);

    /**
     * Intersect this multiset with another one. The result is the modified multiset (this).
     * It will contain all elements that are present in the both multisets.
     * The multiplicity of each element is equal to the minimum multiplicity of
     * the corresponding elements in the intersecting multisets.
     */
    void intersect(Multiset<E> other);

    /**
     * Returns multiplicity of the given element.
     * If the set doesn't contain it, the multiplicity is 0
     */
    int getMultiplicity(E elem);

    /**
     * Check the multiset contains an element,
     * i.e. the multiplicity > 0
     */
    boolean contains(E elem);

    /**
     * The number of unique elements
     */
    int numberOfUniqueElements();

    /**
     * The size of the multiset, including repeated elements
     */
    int size();

    /**
     * The set of unique elements (without repeating)
     */
    Set<E> toSet();
}

class HashMultiset<E> implements Multiset<E> {

    private Map<E, Integer> map = new HashMap<>();

    @Override
    public void add(E elem) {
        Integer value = map.putIfAbsent(elem, 1);
        if (value != null) {
            map.put(elem, value + 1);
        }
    }

    @Override
    public void remove(E elem) {
        Integer value = map.get(elem);
        if (value == null) {
            return;
        }

        if (value.equals(1)) {
            map.remove(elem);
            return;
        }

        map.put(elem, value - 1);
    }

    @Override
    public void union(Multiset<E> other) {
        for (E e : other.toSet()) {
            int otherValue = other.getMultiplicity(e);
            map.merge(e, otherValue, Integer::max);
        }
    }

    @Override
    public void intersect(Multiset<E> other) {
        for (E e : other.toSet()) {
            int otherValue = other.getMultiplicity(e);
            if (map.get(e) != null) {
                map.merge(e, otherValue, Integer::min);
            }
        }

        Set<E> keySet = new HashSet<>(map.keySet());
        for (E e : keySet) {
            if (!(other.contains(e))) {
                map.remove(e);
            }
        }
    }

    @Override
    public int getMultiplicity(E elem) {
        return map.getOrDefault(elem, 0);
    }

    @Override
    public boolean contains(E elem) {
        return map.containsKey(elem);
    }

    @Override
    public int numberOfUniqueElements() {
        return map.size();
    }

    @Override
    public int size() {
        return map.values().stream().reduce(Integer::sum).orElse(0);
    }

    @Override
    public Set<E> toSet() {
        return map.keySet();
    }
}
```

✓ **Correct**

25 users liked this problem. **5** didn't like it. **What about you?**

Continue   Solve again

Time limit: 8 seconds   Memory limit: 256 MB

Comments (18)   Hints (0)   Useful links (0)   Solutions (1)

Share something, Sergey Kubatko

Post   Please do not post solutions here

Sort by: **Last posted**

A **Albert** about 1 month ago Report

I hate that stupid annotations, srsly.
No, srsly, no jokes, for real, it is realy realy bad (for those union/intersection methods at least).

For you guys, who think, that union means that you should UNITE multiplicity of both multisets, you are wrong.
Look for this simple yet stupid example:

First multiset contains (key : multiplicity) :
aaa : 1
bbb : 10
ccc : 100
ddd : 1000

Second multiset contains (key : multiplicity) :
aaa : 10
bbb : 1
ccc : 100
eee : 1000

And if we'll try to execute first.union(second), the correct answer MUST BE:
aaa : 10
bbb : 10
ccc : 100

ddd : 1000
eee : 1000

And a small tip for you, who still tries to understand what INTERSECTION is (as an example we'll take those two sets from above):

Result of first.intersect(second) MUST BE:
aaa : 1
bbb : 1
ccc : 100

Other methods are piece of cake and natively understandable.
Hope it helped at least anyone here.
Cheers.

1 Reply

AZ **ALEKSEI ZIMIN** about 2 months ago Report

When implementing intersect don`t forget to clear map, before updating!

0 Reply

LA **Lens Apperkot** about 2 months ago Report

I don't understand what I should do if both maps doesn't have interseccting elements. For example, map1: a=1, b=2, c=3; map2: x=1, y=2.
Should intersection look like a=0, b=0, c=0, x=0, y=0 ?? Can someone explain implementation of intersection?

0 Reply

U1 **User 1672156** 3 months ago Report

I had a problem with intersect implementation. Note that it should update out multiset (this) to only contain values that are present in both. So after intersect our map should only have values that where present in other with updated multiplicity

0 Reply

DM **Dmitrij Morozov** 3 months ago Report

Intersect - items must be in two lists at the same time.
[a=2, b=3, c=1] - unique items three, not one;

0 Reply

M **MxWild** 3 months ago Report

Yohooo! Three hours.
Read task many times. Especially union and intersect methods.

1 Reply

NS **Natal'ja Shvetsova** 3 months ago Report

There is very unclear problem statement! Clumsy English with illogical use of words and accepted concepts (in comments)... Rewrite it, please!

0 Reply

VP **Vasya Pupkin** 3 months ago Report

It will contain all elements that are **present in the both** multisets.

0 Reply

K **KateEllycott** 3 months ago Report

Guys, before implementing Union method, make sure you understand the concept of unification of multisets!

0 Reply

PW **Paweł Walczuk** 4 months ago Report

Failed. Runtime error
Exception in thread "main" java.lang.AssertionError: getMultiplicity(elem) returned an incorrect result
    at
MultisetDemo.lambdacompareWithMapOrThrowAssertionErrorcompareWithMapOrThrowAssertionError5(Main.java:371)
    at java.base/java.util.HashMap.forEach(HashMap.java:1336)

0 Show all Reply

LA **LAURENT APICELLA** 5 months ago Report

Maybe a clearer definition of the number of unique elements should be given.
This is the correct output based on the method which has passed the tests:
testMS: [a=2, b=3, c=1]
unique elements in testMS: 3

0 Show all Reply

S **Sigavax** 5 months ago Report

After three hours of reading about multisets, going through Guava source code, tweaking on my own code and, of course, contemplating life choices - here I am - I finally solved this "peculiar" task, to say the least. Yet all I feel is rage.

0 Reply

O **olosh_ami** 5 months ago Report

This should be a mandatory problem! But the assertion results are misleading. I had problems in Union method, the error was showing problem with getMultiplicity! Had to look into the logtrace carefully to find out.

0 Reply

I **I_J_K** 6 months ago Report

can work improper methods union(...) and intersect(...), and the system will issue an error method getMultiplicity

0 Show original ↑ Reply

RM **Renat Mukhametshin** 6 months ago Report

oh!!! my code pass the check, but I very tired

0 Reply

RM **Renat Mukhametshin** 6 months ago Report

don't understand why this code wrong for getMultiplicity:

```
public int getMultiplicity(E elem) {
      try{
         return map.get(elem);
```

0 Show all Reply

RM **Renat Mukhametshin** 6 months ago Report

please, why getMultiplicity wrong? my code for this Multiset {aa=2, cc=2, bb=2, Aa=2, A=1, b=2, r=1, c=3, d=3, e=1}

return this (it's ok):

GetMultiplicity for aa 2

0 Show all Reply

U **usr** 10 months ago Report

There is a mistake in "numberOfUniqueElements" method. Here should be size of the set from "toSet" method.

2 Reply