

CSE 459.23 Lab3

Due November, 14 midnight (Monday)

Exercise 1: Extending HashMap

Your goal is to extend the java defined HashMap class to add the ability to store multiple objects. To do this, you will edit MultiHashMap.java to both add new methods and override two methods from HashMap.

- a) (1) You need to implement four new methods (see details in the MultiHashMap.java file)
 - b) `getAll(Object key)`
 - c) `putAll(Object key, Collection values)`
 - d) `contains(Object key, Object value)`
 - e) `remove(Object key, Object value)`
- (2) You need to override two methods in the HashMap class (means methods with the same signature and return type).
 - a) `get`
 - b) `put`

The external behavior of the `get` and `put` methods should be the same as in `HashMap`, but internally they should facilitate the ability to keep track of multiple associations. The only place where `MultiHashMap` should differ from `HashMap` externally is in its compliance with the MultiMap interface (it means that `MultiHashMap` should implement `MultiMap`).

- (3) To test your implementation, compile and run the file MultiHashMapTest.java.

Read the description for each method you are supposed to implement (inside the source code of MultiHashMap.java) carefully before you start! Here is an additional example to show the expected effects of these methods:

(NOTE: the format `*-*-*` stands for a collection of objects. Choose an appropriate class to maintain this collection)

Suppose `testObj` is an object of the `MultiHashMap` class, and it contains no key-value pairs initially.

- 1) `testObj.put("A", "red")`
this method returns NULL and now `testObj` contains an entry: "A" "red"
- 2) `testObj.put("A", "blue")`
this method returns the string "red" and now `testObj` contains an entry: "A" "red"- "blue"
- 3) `testObj.get("A")`
this method returns the string "blue"
- 4) `testObj.getAll("A")`
this method returns "red"- "blue"
- 5) `testObj.putAll("A", "green"- "black"- "grey")`
now `testObj` contains an entry: "A" "red"- "blue"- "green"- "black"- "grey"
- 6) `testObj.get("A")`
this method returns the string "grey"
- 7) `testObj.contains("A", "green")`
this method returns TRUE
- 8) `testObj.remove("A", "black")`
this method returns TRUE and now `testObj` contains an entry: "A" "red"- "blue"- "green"- "grey"

HINT:

- 1) Multiple objects can be organized inside a SINGLE object which is a member of the java collection framework.
- 2) Iterator can NOT guarantee the order of the objects returned from a collection. Consider how to reserve the ordering of the objects associated with a key.

Exercise 2: Sorting made easy

The class `java.util.Collections` (do not confuse this with the interface `java.util.Collection`) provides many useful static methods for use with `Collections`, `Lists`, and `Sets`, such as `sort`, and `binarySearch`.

In this exercise, you will utilize the `Collections.sort()` method on a `List` of your own abstract data type. Normally, `Collections.sort(List)` takes as argument one `List` of `Comparable` elements and modifies the `List` so that the elements are sorted. We wish to sort a collection `RatNum` objects, but `RatNum` does not implement the `Comparable` interface, so we cannot use `Collections.sort`.

We could simply modify the `RatNum.java` code, but assume that option is not available (often you will not be able to decide to modify a code because other people depend on the current version). The alternative is to call `Collections.sort(List, Comparator)`. The `Comparator` is a class you implement that only needs a single method, `public static int compare(Object o1, Object o2)`, which returns a negative number if $o1 > o2$, 0 if $o1 == o2$, and a positive number if $o1 < o2$.

Now the `Comparator` defined inside `RatNumSortingTest.java` simply returns 0 for its `compare()` method, treating all `RatNums` as equal values in the sorting. So your task is to modify the `compare()` method of `RatNumSortingTest.java` so that it follows the appropriate specifications of a `Comparator`.

Go to <http://www.cse.ohio-state.edu/~xut/459.23> for .java files needed in this lab!