## Lab 9

In lab 9, you will use Java generics to implement an interface for maintaining a one-to-one mapping.

The name of your Eclipse project should be abc123-lab9, where you replace abc123 with your abc123 id. The name of the file that contains the main method should remain DualMapTest.java. The other files should also keep their original names. The file numbers.txt should be copied to the project directory (should not be in src or bin). To submit the project, export the project and upload the zip file to Blackboard.

### Task

Your task is to write the DualHashMap class so it works in combination with the code in [lab9.zip](http://cs.utsa.edu/~cs3443/laboratories/lab9.zip). DualHashMap should implement the DualMap interface so the main method in DualMapTest runs correctly.

The idea of the DualMap interface is to maintain a one-to-one mapping between two sets (the "keys" and the "values"). This is similar to the Map interface.

In the Map interface, each key is associated with one value, but duplicate values are allowed. For example, the keys might be id numbers and the values might be names. Two id numbers might map to the same name because two people might have the same name.

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

map.put(123456. "Queen Elizabeth");

map.put(654321. "Queen Elizabeth");

// prints Queen Elizabeth twice

System.out.println(map.get(123456));

System.out.println(map.get(654321));

In a DualMap, duplicate values are not allowed (so not a good choice for id numbers to names). A DualMap also has a reverseGet method that returns the key for a specified value.

DualMap<Integer,String> dualmap = new DualHashMap<Integer,String>();

dualmap.put(13, "thirteen");

dualmap.put(42, "forty-two");

// prints thirteen and 42

System.out.println(dualmap.get(13));

System.out.println(dualmap.reverseGet("forty-two"));

As with Maps, DualMaps have two generic type parameters (see the Pair2 class from the notes for an example). Your DualHashMap.java does not need to know the types for keys and values, but to work properly, both types should have an equals method and a hashcode method. You do not need to implement equals or hashcode as the types used in DualMapTest.java have these methods.

To implement DualHashMap, there should be two HashMap instance variables for the two mappings to be maintained: one from keys to values and the other from values back to keys. The constructor for DualHashMap needs to create these HashMaps. DualMap.java provides information on how the methods of a DualMap are supposed to operate. When you run DualMapTest you should get output like:

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

January February March April May June July August September October November December

null Feb null Apr null Jun null Aug null Oct null Dec

null February null April null June null August null October null December

U M T W R F S

Sunday Monday Tuesday Wednesday Thursday Friday Saturday

null null null null null null null

null null null null null null null

U M T W R F S

Sunday Monday Tuesday Wednesday Thursday Friday Saturday

[nine, thousand, eight, hundred]

[thirty-one, thousand, sixty]

[twenty-seven, thousand, two, hundred, ninety-nine]

[seventeen, thousand, four, hundred, eleven]

[twelve, thousand, sixty-five]

[ten, thousand, nine, hundred, ninety-six]

[eight, thousand, nine, hundred, twenty-four]

[three, thousand, three, hundred, eighty-three]

[twenty-six, thousand, five, hundred, eighty-seven]

[thirteen, thousand, three, hundred, seventy-two]

19986 3931 9843 18911 11453 16495 23820 14675 13274 18724

The first 10 lines of your output should be identical. The last 11 lines will very likely be different because the numbers are randomized.

### Comments

Add javadoc comments to your DualHashMap.java.

### UML Class Diagram

Create a UML class diagram using Violet.

### Rubric

* An incorrect submission will possibly get zero points. A project that does not compile will receive at most 50 points total.
* (80 pts.) If running DualMapTest produces correct output. This will proportional to the number of lines that are correctly printed out.
* (10 pts.) If there are javadoc comments for DualHashMap.java: at least one for the class and comments for each variable, method and constructor.
* (10 pts.) A UML class diagram is included that has all the classes (including the DualMapTest.java class) and all the constructors, methods, variables and constants of each class.