CSC301 HW4

Instructions:

1. Download all the files from the dropbox for HW4.
2. Start up Eclipse with the same workspace as you used for HW1.
3. Inside the src folder, create a hw4 package.
4. Select all the .java files you downloaded and copy and paste them into the hw4 package in Eclipse.
5. You should now be able to edit the LinearProbingHashST class. This is the same code as discussed in the book. Modify it so that implements lazy deletion as follows:
    - When a key is deleted, set the corresponding value for that key to null, but do not physically remove the key from the key array.
    - When searching for a key, remember that if the corresponding value is null, then that key is not in the symbol table.
    - When inserting, make use of any lazy deleted slots (slots whose keys are not null but whose values are null) whenever possible.
        - *Hint:* Scan the chain until you find a valid slot for insertion. If the slot is the key itself or an empty slot, you can insert as before. If the slot is a lazy deleted key, reuse the slot, but then you have to continue looking down the chain to try to lazy delete the old key if it appears further down the chain, otherwise the same key will appear twice in the keys array.
    - When returning a Key iterator with the keys() method, make sure that lazy deleted keys are not included.
    - When resizing, make sure not to copy over lazy deleted keys. NOTE: Besides lazy deleted slots being reused, this is the only time keys are removed the key array.
    - The field **n** which keeps track of the number of keys in the symbol table should reflect the number of logical keys. In other words, it should not include lazy deleted keys in the count. (It should not include keys whose corresponding value is null.)
6. Use the HW6Test class to create your own JUnit tests for this assignment. Note that I added additional methods so that you can look inside the LinearProbeHashST in order to test its behavior. Without these extra functions, you couldn't for example check if a particular slot holds a lazy deleted key, because this is not really part of the public API for the symbol table.

Restrictions:

- You may not add any fields to the LinearProbeHashST.
- You may only create new classes for testing purposes. Otherwise, your LinearProbeHashST class should be completely self-contained.

Hints:

Make sure your tests at least check the following:

- Various sequences of put, get, delete and keys still work correctly. (In other words, they have the same input/output behavior as before you made the changes.)
- Lazy deleted slots are reused by inserts.

To do this, you will want to use keys with predictable hashCode functions so that you can anticipate where those keys should be stored, and then check that they really are stored there. The built in Integer class is a good choice because the hashCode() for any Integer object is its value.

Grading:

Your grade will be based on how your code performs on a Junit test I will design. Like HW1, I will post my Junit test after the deadline and you will have until the resubmit deadline to make a resubmission. As usual, resubmissions will be graded with a 15 point penalty.