Name: Ryan Linden & Brandon Ngo

UWNetID: \_\_\_\_\_\_ & ngo12

Section: \_\_\_\_\_\_ & BC

1. Describe your approach to implementing the hash table. (separate arrays for keys and values vs one array with objects that represent pairs.)

In order to implement the has table, we used one array with objects that represented pairs. We first created a class called HashEntry as seen in Figure 1. The class HashEntry consists of a ColorKey object called myKey as well as a long called myValue. The ColorKey is used to establish the key of the dictionary pair while the long is used to establish the value of the dictionary pair. The ColorHash class then has an array of HashEntry objects called hashTable.

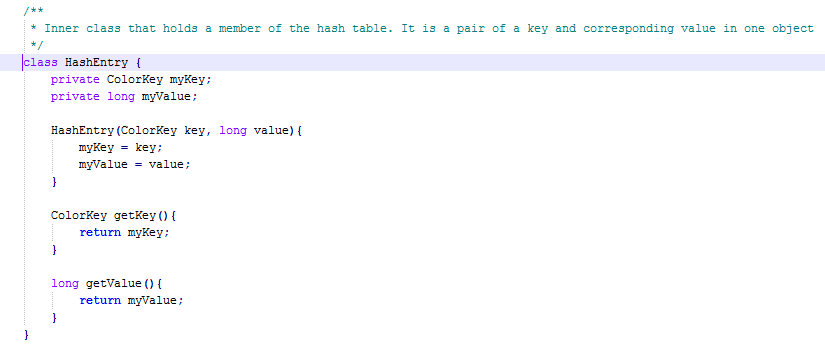


Figure 1. The HashEntry class with properties of an object ColorKey and a long to create dictionary pairs that will go into the hash table.

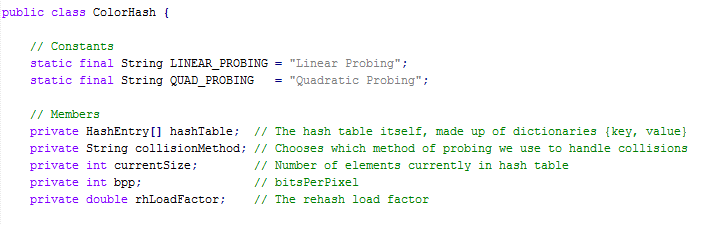


Figure 2. The ColorHash class with an array of HashEntry objects.

2. If you implemented any methods other than those in the specification, describe them briefly here.

1. private int**[]** probing**(**ColorKey key**):** Method seen in Figure 3 that performs probing in order to resolve collisions. The method records the point of insertion and the number of collisions while probing using a while loop of conditional statements. The type of probing depends on the probing type of the hash table. An array of int is returned with the int stored at index 0 being the index of the point to insert the key, value pair. The int stored at index 1 is the number of collisions that occurred while probing.

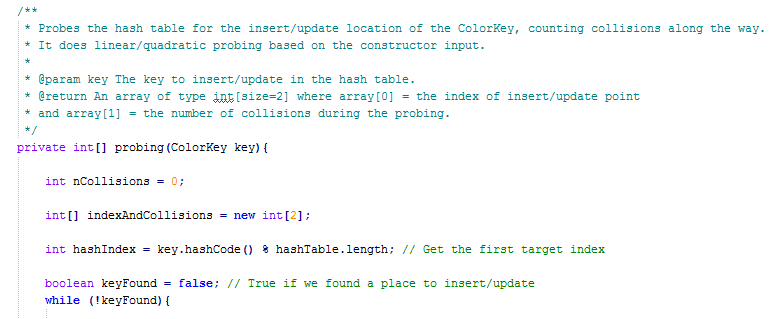


Figure 3. A snippet of the Private int[] probing (ColorKey key) with the array of int highlighted.

1. private boolean checkRehashing**():** Method seen in Figure 4 that checks if the load factor has exceeded its threshold value. If so, the resize() method is called in order to resize the table and insert the old elements from the original table into the new resized table.

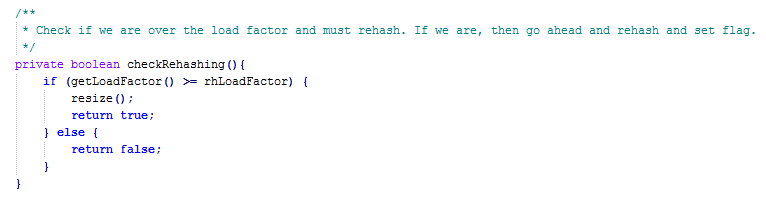


Figure 4. A snippet of the private boolean checkRehasing() method.

3. When you use 6 bits per pixel, how many black pixels are there in the Mona Lisa image? (These are the pixels whose ColorKey bits value equals 0.)

There are 62007 black pixels in the Mona Lisa Image.

4. Provide a copy of the table of counts produced by your collisionTests method.

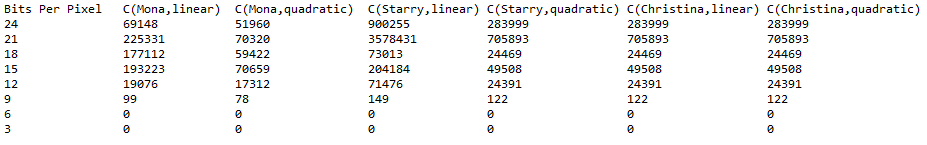


Figure 5. The table of counts produced by our collisionTests method. The likelihood of collisions decreased with the smaller key space as expected.

5. Provide a copy of the table of similarity values produced by your fullSimilarityTests method.

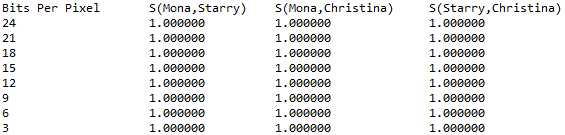


Figure 6. The table of similarity values produced by our fullSimilarityTests method. The similarity values increased to 1.0 with the smaller key space as expected.

6. Examine the hashCode method of class ColorKey. What types of images might tend to cause lots of collisions relative to other images?

7. (Answer this only if you are doing the optional makeup-credit item)