Random Probing

Suppose that the hash table has b buckets (mTableSize is b). In linear open addressing the buckets are examined in the order (f(k)+i) % b, 0 <= i < b, where k is the key of the element being searched for. In random probing a pseudo-random number generator is used to obtain a random sequence R(i), 1 <= i < b where R(1), R(2), ... R(b-1) is a permutation of [1, 2, ..., b-1]. The buckets are examined in the order f(k), (f(k)+R(i)) % b, 1 <= i < b.

Note: the Random class is in **java.util**

The easiest way to generate R(i) is to instantiate a Random object (BEFORE the while loop in findPos), for which you pass f(k) (what myHash returns). Then if you iterate in the loop, call the Random object's (you already instantiated) nextInt() method to get the next "random" number (and similar to myHash, make sure that after you % mTableSize, that if it's < 0, you add mTableSize).

There is a possibility that after adding a particular random value (and after %, checking if < 0) , you may get the same location again. In a "safer" implementation, you should check if the new location is the same as the previous (and get the nextInt again if so).