PhoneBook FAQ

WHY CAN'T YOU MAKE A GENERIC ARRAY?! AM ANGRY.
You'll have to Google the actual implementation details of why. A high level explanation is that generics were introduced with Java version 5; Java engineers had to make them work with what they already had.

What should I do if my hash table is full?
Usually you would double the size of the table (similar to what you did when writing an array-backed List and Stack). Of course this would require you to re-hash the table's contents if you rely on modulus in your hash function, as some keys would hash to different table locations with a larger table.

For example, with table size of 10, a key of 35 would hash as such: `h(35) = 35 % 10 = 5`. If the table is doubled in size: `h(35) = 35 % 20 = 15`.

One way around this is to use prime numbers for table length. When doubling the size of the array, you round to the next prime above twice the current size. You can also cache these values ahead of time (borrowed from SO):

```
private final int[] SIZES = { 1019, 2027, 4079, 8123, 16267, 32503, 65011, 130027, 260111, 520279,
1040387, 2080763, 4161539, 8323151, 16646323 };
```

Modding by primes will allow you to copy elements to the new array without re-hashing them.

When should I resize my array?
Usually the array is doubled if a particular *load factor* is reached. Load factor is a measure of how many data buckets in the array are full (*not* the same as the size of the array). A common load factor benchmark is 0.75 (i.e. if 75% of data buckets are full, resize the array to avoid excessive collisions).

The hashing labs ignore load factor (and table resizing) for the sake of expediency; an enterprise-level hash table implementation would have to take care of all of these details (and create Iterators, implement Serializable, etc.).