```
class ChainNode<K, V> {
  private K key;
  private V value;
  ChainNode<K, V> next;

  public ChainNode(K newKey, V newValue,
                   ChainNode<K, V> nextNode) {
    key = newKey;
    value = newValue;
    next = nextNode;
```

```
  } // end constructor

  public V getValue() {
    return value;
  } // end getValue

  public K getKey() {
    return key;
  } // end getKey

} // end ChainNode

// ********************************************************
// Hash table implementation.
// Assumption: A table contains at most one item with a
//             given search key at any time.
// Note: This code will compile with a warning about the use
// of unchecked or unsafe operations. This is due to the
// cast in method tableRetrieve.  Exercise X asks you to
// rewrite this implementation using ArrayList to avoid this
// warning.
// ********************************************************

public class HashTable<K, V> {
  public final int HASH_TABLE_SIZE = 101;
  private ChainNode[] table;     // hash table
  private int size = 0;          // size of ADT table
```

```
  public HashTable() {
    table = new ChainNode[HASH_TABLE_SIZE];
  }  // end default constructor

// table operations
  public boolean tableIsEmpty() {
    return size==0;
  }  // end tableIsEmpty

  public int tableLength() {
    return size;
  }  // end tableLength

// Programming Problem 4 asks you to implement the following
// methods.

  public void tableInsert(K key, V value)
                          throws HashException {
    // ...
  }  // end tableInsert

  public boolean tableDelete(K searchKey) {

    // ...
    return true;  // added for compilation
  }  // end tableDelete

  public V tableRetrieve(K searchKey) {
    // ...
    return null;  // added for compilation
  } // end tableRetrieve

  public int hashIndex(K key) {
    // ...
  }  // end hashIndex

}  // end HashTable
```

Items are stored in the table using `ChainNode`, which expects *<key, value>*