SinglyLinkedList.java

Objective: To create a singly linked list and add functionality to manipulate the list.

Background:
A linked list is a linear collection of data that stores data in a simple, regular structure. It consists of nodes in which each node contains a value and a link (pointer) to the next node. The last node in the list points to **null**. The advantage of linked lists is they make it easy to insert, remove, and reorder data without the data being contiguous in memory. Linked lists can grow and shrink dynamically, and can be used to implement other data structures such as queues and stacks.

Assignment:
Download the zip file **SinglyLinkedList.zip** and unzip. It will create a directory **SinglyLinkedList** and do all of your work in this directory.

You will add the following functionality to the **SinglyLinkedList** class. Use the **ListNode** class for nodes. User **SinglyLinkedListTester.java** (provided) to test your class. The data type **E** must be a **Comparable** to work properly. The **Coordinate** class implements **Comparable** and has been provided for testing. Be sure to adjust your list's **head** and **tail** pointers in each method as necessary.

1. **boolean add (E value)** Adds **value** to the end of the list and returns **true** if the add is successful.

2. **boolean add (int index, E value)** Adds **value** to the **index** location in the list. If **index** is one greater than the largest index of the list, then the **value** is added at the end of the list. If **index** is not in the list, then it throws **java.util.NoSuchElementException**.

3. **void clear()** Clears the list of nodes.

4. **int size()** Returns the number of nodes in the list.

5. **ListNode<E> get (int index)** Returns the **ListNode** at the **index** location in the list. If index is not in the list, then it throws **java.util.NoSuchElementException**.

6. **E set (int index, E value)** Sets the **index** node to **value** and returns the original value in the node. If **index** is not in the list, then it throws **java.util.NoSuchElementException**.

7. **E remove (int index)** Removes the **index** node from the list and returns the **value** it removed. Throw **java.util.NoSuchElementException** if the list is empty.

8. **boolean isEmpty ()** Returns **true** if the list is empty; **false** otherwise.

9. **boolean contains (E value)** Returns **true** if the list contains at least one **value**; **false** otherwise.

10. **int indexOf (E value)** If found, returns the index of the node matching **value**. Otherwise, it returns **-1**.

11. **public SinglyLinkedList(SinglyLinkedList oldList)** Implement a copy constructor for **SinglyLinkedList**.