# Java Week Six

1)

**public** **void** add(**int** index, T value) **throws** ListAccessError

{

//temp node equals the value

Node<T> temp = **new** Node<T>(value);

//if it is empty

**if**(isEmpty())

{

//head is the value

head = temp;

//tail equals what the head did

tail = head;

}

**else**

{

//newNode will be the starting point, the head

Node<T> newNode = head;

**int** i = 1;

//go through the list until you are at the node before the one you want

**while**(i < index)

{

newNode = newNode.getNext();

i++;

}

//set the new node and the one before it

temp.setNext(newNode.getNext());

newNode.setNext(temp);

}

// add to the number of nodes

noOfNodes ++;

}

@Override

**public** T remove(**int** index) **throws** ListAccessError

{

Node<T> temp = head;

//if its node empty

**if**(!isEmpty())

{

**int** i = 1;

//go through the list to the one before you want

**while**(i != index)

{

temp = temp.getNext();

i++;

}

//shift the nodes

temp.setNext(temp.getNext().getNext());

noOfNodes --;

}

**return** **null**;

}

@Override

//get the value of the node at index

**public** T get(**int** index) **throws** ListAccessError

{

**return** getNode(index).getValue();

}

**private** Node<T> getNode(**int** index) **throws** ListAccessError

{

**if** (index < 0 || index >= noOfNodes)

{ // invalid index

**throw** **new** ListAccessError("Index out of bounds");

}

Node<T> node = head; // start at head of list

**for** (**int** i = 0; i < index; i++)

{ // walk through list to desired index

node = node.getNext(); // by following next references

}

**return** node; // return the node at the required index

}

2)

@Test

**public** **void** randomArrayTestLow() **throws** ListAccessError

{

**long** startTime = System.*nanoTime*();

SinglyLinkedLists<Integer> list = **new** SinglyLinkedLists<Integer>();

RandomIntegerArray rand = **new** RandomIntegerArray(500);

Integer[] temp = rand.randomArray(500);

**for**(**int** i = 0; i < temp.length; i++)

{

list.add(i, temp[i].intValue());

}

**long** endTime = System.*nanoTime*();

System.***out***.println("Array Position: " + temp[375].toString() + " List Position: " + list.get(375) + " Time Taken in Microseconds: " + (endTime-startTime)/10000);

*assertEquals*(temp[375],list.get(375));

}

Result:

Array Position: 174 List Position: 174 Time Taken in Microseconds: 61

@Test

**public** **void** randomArrayTestHigh() **throws** ListAccessError

{

**long** startTime = System.*nanoTime*();

SinglyLinkedLists<Integer> list = **new** SinglyLinkedLists<Integer>();

RandomIntegerArray rand = **new** RandomIntegerArray(50000);

Integer[] temp = rand.randomArray(50000);

**for**(**int** i = 0; i < temp.length; i++)

{

list.add(i, temp[i].intValue());

}

**long** endTime = System.*nanoTime*();

System.***out***.println("Array Position: " + temp[4756].toString() + " List Position: " + list.get(4756) + " Time Taken in Microseconds: " + (endTime-startTime)/10000);

*assertEquals*(temp[4756],list.get(4756));

}

Result:

Array Position: 31283 List Position: 31283 Time Taken in Microseconds: 220797

