Kristine Trinh

Nlt895

11190412

1. Implementing the test method for **isEmpty()**

public boolean isEmpty() {

return false;

}

@Test

public void testIsEmpty() {

System.out.println("isEmpty");

LinkedList instance = new LinkedList();

boolean expResult = true;

boolean result = instance.isEmpty();

assertEquals(expResult, result);

}

Test result: testIsEmpty Failed: expected: <true> but was:<false>

public boolean isEmpty() {

return (info == null && nextList == null);

}

Test result: passed!

1. Implementing the test for **insert()**

public void insert() {

return;

}

@Test

public void testInsert() {

System.out.println("insert");

LinkedList instance = new LinkedList();

instance.insert("A");

instance.insert("B");

instance.insert("C");

assertFalse(instance.isEmpty());

}

Test result: testInsert caused an ERROR: uncompilable source code

public void insert(String info) {

lList oldList = new lList();

oldList.info = this.info;

oldList.nextList = this.nextList;

this.info = info;

this.nextList = oldList;

}

Test result: passed!

1. Implementing the test for **nextList()**

public LinkedList nextList() {

return null;

}

@Test

public void testNextList() {

System.out.println("nextList");

LinkedList instance = new LinkedList();

instance.insert("A");

instance.insert("B");

String expResult = "A";

LinkedList result = instance.nextList();

assertEquals(expResult, result);

}

Test result: testNextList Failed: expected <A> but was:<null>

public lList nextList() {

if (!isEmpty())

return nextList;

return null;

}

@Test

public void testNextList() {

System.out.println("nextList");

lList instance = new lList();

String expResult = null;

lList result = instance.nextList();

assertEquals(expResult, result);

}

Test Result: passed!

@Test

public void testNextList() {

System.out.println("nextList");

lList instance = new lList();

String expResult = null;

lList result = instance.nextList();

assertEquals(expResult, result);

lList instance2 = new lList();

instance2.insert("A");

instance2.insert("B");

String expResult2 = "A" ;

lList result2 = instance2.nextList();

assertEquals (expResult2,result2.info);

}

Test Result: passed!

1. Implementing the test for **lSize()**

public int lSize() {

return 0;

}

@Test

public void testLSize() {

System.out.println("lSize");

LinkedList instance = new LinkedList();

int expResult = 0;

int result = instance.lSize();

assertEquals(expResult, result);

}

Test result: The test case is a prototype.

public int lSize() {

int result = 0;

LinkedList temp = new LinkedList();

temp.nextList = nextList;

while (temp.nextList != null) {

result++;

temp = temp.nextList;

}

return result;

}

@Test

public void testLSize() {

System.out.println("lSize");

lList instance = new lList();

int expResult = 0;

int result = instance.lSize();

assertEquals(expResult, result);

lList instance2 = new lList();

instance2.insert("K");

instance2.insert("Z");

instance2.insert("P");

int expResult2 = 3;

int result2 = instance2.lSize();

assertEquals(expResult2, result2);

}

Test result: passed!

1. Implementing the test for deleteInfo()

public void deleteInfo(String outInfo) {

return;

}

@Test

public void testDeleteInfo() {

System.out.println("deleteInfo");

LinkedList instance = new LinkedList();

instance.insert("V");

instance.deleteInfo("V");

assertEquals(null,instance);

}

Test result: expected:<null> but was:[linearstructures.LinkedList@6d9c638](mailto:linearstructures.LinkedList@6d9c638)

@Test

public void testDeleteInfo() {

System.out.println("deleteInfo");

LinkedList instance = new LinkedList();

instance.insert("V");

instance.deleteInfo("V");

assertEquals(null,instance.info);

}

Test result: passed!

@Test

public void testDeleteInfo() {

System.out.println("deleteInfo");

lList instance = new lList();

instance.insert("V");

instance.deleteInfo("V");

assertEquals(null,instance.info);

instance.insert("H");

instance.insert("I");

instance.insert("S");

instance.insert("U");

instance.insert("N");

instance.deleteInfo("I");

instance.deleteInfo("U");

int expResult2 = 3;

int result2 = instance.lSize();

assertEquals(expResult2,result2);

}

Test result: passed!

1. Implementing the test for traverse()

public String tranverse() {

return null;

}

}

@Test

public void testTranverse() {

System.out.println("tranverse");

LinkedList instance = new LinkedList();

String expResult = "";

String result = instance.tranverse();

assertEquals(expResult, result);

}

Test result: expected:<> but was:<null> junit.framework.AssertionFailedError

public String tranverse() {

String result = "";

if (!isEmpty()) {

LinkedList temp = new LinkedList();

temp.nextList = nextList;

temp.info = info;

while (temp.nextList != null) {

result += temp.info + ",";

temp = temp.nextList;

}

result = result.substring(0, (result.length() - 1));

}

return result;

}

@Test

public void testTranverse() {

System.out.println("tranverse");

LinkedList instance = new LinkedList();

String expResult = "";

String result = instance.tranverse();

assertEquals(expResult, result);

}

Test result: passed!

@Test

public void testTraverse() {

System.out.println("tranverse");

lList instance = new lList();

String expResult = "";

String result = instance.traverse();

assertEquals(expResult, result);

lList instance2 = new lList();

instance2.insert("A");

instance2.insert("F");

instance2.insert("G");

String expResult2 = "G,F,A";

assertEquals(expResult2,instance2.traverse());

}

Test result: passed!

1. Test all of the function

@Test

public void testAll() {

lList list = new lList();

list.insert("A");

list.insert("V");

list.insert("A");

list.insert("J");

System.out.println("The content is " + list.traverse());

System.out.println("The size is " + list.nextList().lSize());

list.deleteInfo("V");

list.deleteInfo("J");

System.out.println("The content is " + list.traverse());

System.out.println("The size is " + list.nextList().lSize());

}

Test result: passed!