**Course exercise for Object oriented programming**

1. Implementing a singly-Linked lists.

Given:

|  |  |
| --- | --- |
| class SList{  public SListNode head;  public int size;  public SList(){  head = null;  size = 0; }  public int length() {  return size; }  public void insertFront(Object item){  head = new SListNode(item, head);  size++; }  public String toString() {  int i;  Object obj;  String result = "[ ";  SListNode cur = head;  while (cur != null) {  obj = cur.item;  result = result + obj.toString() + " ";  cur = cur.next; }  result = result + "]";  return result;  }  } | class SListNode{  public Object item;  public SListNode next;  public SListNode(Object o, SListNode n){  item = o;  next = n;  }  public SListNode(Object o){  item = o;  next = null;  }  } |

1. Add an *insertEnd*(Object item) method for the SList class. This method inserts the parameter "item" at the end of this list.

**public void insertEnd(Object item) {**

**SListNode newNode = new SListNode(item);**

**size++;**

**// Edge Case : If the LL is already empty , the node is set to head.**

**if (head == null) {**

**head = new SListNode(item);**

**return;**

**}**

**newNode.next = null; // Because this will be our last node.**

**SListNode last = head;**

**// Starting from head, traversing the LL looking for the last node.**

**while (last.next != null) {**

**last = last.next;**

**}**

**last.next = newNode; // Pointing the last node to our new node.**

**return;**

**}**

and explain your design:

The algorithm is pretty much self-explanatory. I've described what each line does

with comments.

The Algorithm - There are 2 cases. In both cases, we create a newNode at first. Then,

If the LL is already empty, then we create a new node and set that as head. If LL is

not empty, then Starting from head, we traverse the LL with a while loop looking for

the last node. when we find the last node, we set last node's next to our new node.

1. Add a *nth*(int position) method for the SList class. This method returns the item at the specified position. If position < 1 or position > this.length(), null is returned. The range of the parameter position is from 1 to length().

**public int nth(int position){**

**if (position < 1) {**

**return null;**

**}**

**int count = 1;**

**SListNode cur = head;**

**// Traversing the SList and looking for the nth position.**

**while (cur != null && count <= position) {**

**if (count == position) {**

**return cur.item.toString();**

**}**

**count += 1;**

**cur = cur.next;**

**}**

**return null;**

**}**

and explain your design:

At first, we assume that the term 'position' is 1-indexed, meaning it's range is from 1 to the length of the LL.

The Algorithm - At first we set a count variable to 1 and a cur node refer to head. now, while cur isn't null (to eliminate edge cases) and count <= passedPosition we loop through the LL looking for the n-th node. if count == position, we return the item. else we increment count and let cur refer to the next node. if we reach the end, that means that the parameter wasn't in the range. Thus, we return null at the end.

1. Write a Test class to test your code. In your test code, you can use the following line of code to output ints in your int list.

//Your code

**public class testClass {**

**public static void main(String[] args) {**

**// Creating a SList instance named 'll'.**

**SList ll = new SList();**

**ll.insertFront(10);**

**ll.insertFront(20);**

**ll.insertFront(30);**

**ll.insertFront(40);**

**ll.insertFront(50);**

**System.out.println("\nAt first, the SList is - " + ll.toString()+ ", it's size is - " + ll.length());**

**int curSize = ll.length();**

**System.out.println("Checking task 1, insertEnd's functionality :\n");**

**ll.insertEnd(13);**

**ll.insertEnd(97);**

**ll.insertEnd(11);**

**System.out.println("Inserted " + (ll.length() - curSize)+ " items at the end of SList.");**

**System.out.println("Now, the SList is - " + ll.toString()+ ", it's size is - " + ll.length());**

**System.out.println("\nChecking task 2, nth(int position)'s functionality : :\n");**

**for (int i = -1; i <= ll.length() + 1; i++) {**

**System.out.println("The item at " + i +"th position is "+ ll.nth(i));**

**}**

**// System.out.println(ll.toString());**

**}**

**}**

1. Run your program and take a screen shot and paste it here.

//image of your screen shot

