COMPSCI 105: Principles of Computer Science
Summer 2007

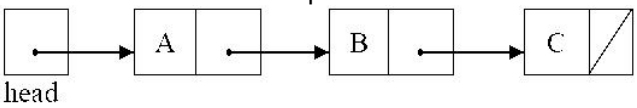
Tutorial Seven: Linked list, Stack, and Queue

Helen Gu and Slobodan Vukanovic
{ygu029, svuk002}@ec.auckland.ac.nz

This tutorial is not being assessed. It provides you with an opportunity to become familiar with concepts introduced in lectures.

Question One: Linked list

Write Java statements that create a linked list pictured as follows.



The Node class (see also file `Node.java`) is given as:

```
public class Node {
      private Object item;
      private Node next;
      public Node(Object item) { this.item = item; next = null; }
      public Node(Object item, Node next) {
            this.item = item; this.next = next; }
      public Object getItem() { return item; }
      public void setItem(Object item) { this.item = item; }
      public Node getNext() { return next; }
      public void setNext(Node next) { this.next = next; }
      public static void printList(Node list) {
            if (list != null) {
                  System.out.println(list.getItem());
                  printList(list.getNext());   }
      }
}
```

a) Write a method called *createLinkedList1* to create the above linked list beginning with an empty linked list, first create and attach a node for A, then create and attach a node for B, and finally create and attach a node for C.

```
public static Node createLinkedList1() {//complete in T07Q1.java, plz
```

b) Similar to Part a), write a method called *createLinkedList2* to create the above linked list, but instead create and attach nodes in the order C, B, A.

```
public static Node createLinkedList2() {//complete in T07Q1.java, plz
```

Question Two: Linked list

Complete *howMany* method to count how many times a data object is contained in linked list.

```
public class ListReferenceBased implements ListInterface {
      private Node head; // head of the linked list
      private int numItems; // number of items in list
      // some other methods …
      public int howMany(Object item) { //complete this here on paper
```

Question Three: Stack and Queue

Trace the execution of the following program. Draw the contents of the Stack and the Queue every time there is a change in contents. Please also write the program output in the end.

```
QueueInterface q = new QueueReferenceBased();
StackInterface s = new StackReferenceBased();
s.push(new Integer(2));
s.push(new Integer(1));
q.enqueue(s.pop());
s.push(new Integer(3));
q.enqueue(new Integer(5));
q.enqueue(new Integer(0));
System.out.print(q.dequeue());
s.push(q.dequeue());
System.out.print(q.dequeue());
System.out.print(s.pop());
```

Stack s:

Queue q:

Output:

Question Four: Stack and Queue

Please complete following code to implement a queue using two stacks s1 and s2. You may use **only** the operations: *pop(), push(Object o)* and *isEmpty()*, of the standard stack ADT. (Note that the queued objects are enqueued into one stack and dequeued from the other.)

```
public class TwoStacksQueue {
      private StackReferenceBased s1; // Enqueueing stack
      private StackReferenceBased s2; // Dequeueing stack
      public TwoStacksQueue {
           s1=new StackReferenceBased ( );
           s2=new StackReferenceBased ( );
      }
      public void enQueue(Object o) {
           //complete this code here on paper



      }
      public Object deQueue( ) {
           //complete this code here on paper









      }
}
```