${\rm CSC\,148H\ L5101\ Midterm\ 2003}$ Duration — 50 minutes Aids allowed: none

Student Number:		
Lab day, time, room:		
Last Name:	First Name:	
Do not turn this page until you have received the signal to start. (Please fill out the identification section above, and read the instructions below.) Good Luck!		
This midterm consists of 3 questions on 5 pages you receive the signal to start, please make sure	,	# 1:/11
Comments are not required except where indicat	ed, although they may help	# 2:/10
us mark your answers. They may also get you pa out how to write the code.	art marks if you can't figure	# 3:/ 9
Write your student number at the bottom of page	ges 2-5 of this test.	
If you use any space for rough work, please indi-	cate clearly what you want	TOTAL: /30

TOTAL: _____/30

Question 1. [11 MARKS]

Complete the bodies of methods queueToList(Queue) and lastTenNonEmptyValues(Node) in the two classes Question1A and Question1B, according to their external and internal comments.

```
public interface Queue {
  void enqueue(Object o);
  String dequeue();
  String head();
  int size();
}
public class Node {
  public String value;
 public Node link;
 public Node(String value) { this.value = value; }
}
public class Question1A {
  /** Return a linked list containing the elements of 'q', in the same order.
    * (in particular, if q isn't empty the returned Node contains the value q.head()).
    * Requires: q != null.
    * Ensures: q.size() == 0.
    * @return the first Node (null if none) in the linked list of values. */
  public static Node queueToList(Queue q) {
    if (q.size() == 0) { return null; }
   Node first = new Node(q.dequeue()); // The first Node in the returned list.
   Node last = first;
                                        // The last Node in the returned list.
```

```
return first;
}
```

```
public class Q implements Queue {
    /** A Q that can hold up to 'capacity' elements.
    * Requires: capacity >= 0. */
    public Q(int capacity) { /* body not shown */ }

    // rest of class not shown
}

public class Question1B {

    /** Return a Queue containing, in any order, the last ten non-empty Strings
    * from the linked list 'list'.
    * If there are less than ten, return all of them.
    * @param list the first Node (null if empty) in a linked list. */
    public static Queue lastTenNonEmptyValues(Node list) {

        Queue q = new Q(10); // this is the only instance of Q you may use.

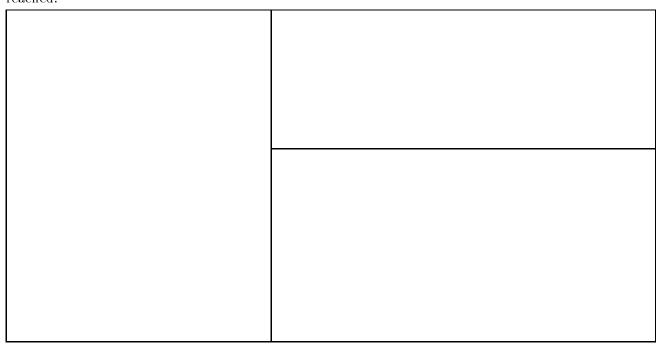
        // You may not traverse (loop over) the list more than once.
        // Hint: there's a reason we're using a Queue.
```

```
return q;
}
```

Question 2. [10 MARKS]

```
public class A {
                                        public class B extends A {
  public static void p(A a) {
                                          public static void p(B b) {
    System.out.println("A.p");
                                            System.out.println("B.p");
                                            b.m();
    a.m();
    a.r();
                                          public void m() {
  public void m() {
    System.out.println("A.m");
                                            System.out.println("B.m");
  private void r() {
                                          private void r() {
    System.out.println("A.r");
                                            System.out.println("B.r");
}
                                        }
public class M {
  public static void main(String[] args) {
    B b = new B();
   b.p(b);
    A.p(b);
}
```

Part (a) [5 MARKS] Draw the memory model when the beginning of line 1 of B's method p is first reached:



Part (b) [5 MARKS] Write the output from running the entire program M:

Question 3. [9 MARKS]

```
Part (a) [2 MARKS]
```

Write your student number at the bottom of every page of the midterm (except the front page).

```
Part (b) [7 MARKS]
```

Write the body of hasLine() in the following class.

You are not required to throw an exception if the user of hasLine() violates the precondition.

```
import java.io.*;

/** For reading lines from a BufferedReader without receiving IOExceptions. */
public class BRWrapper {

   private BufferedReader br;
   private String lastLine; // last line read by hasLine (if it was successful)

   /** A BRWrapper returning lines from 'br'.
        * Requires: br != null. */
   public BRWrapper(BufferedReader br) { this.br = br; }

   /** Attempt to read the next line and return whether reading was successful.
        * Requires: line() be called in-between calls to this method.
        * Oreturn true iff there is a line *and* there was no IOException while reading. */
   public boolean hasLine() {
```

```
/** Return the line read by hasLine().
    * Requires: hasLine() returned true.
    * Ensures: returned value != null. */
    public String line() {
        return lastLine;
    }
}
```

Total Marks = 30