/\*\*
 \* A Node (containing an int) in a singly-linked
 \* data structure
 \*
 \* @version 1.0
 \* @author Prof. David Bernstein, James Madison University
 \*/
public class IntNode
{
 public int value;
 public IntNode next;
}

/\*\*
 \* A Node (containing an Object) in a singly-linked
 \* data structure
 \*
 \* @version 1.0
 \* @author Prof. David Bernstein, James Madison University
 \*/
public class Node
{
 public Object value;
 public Node next;
}

/\*\*
 \* A Pile of int values
 \*
 \* This implementation uses linked memory
 \*
 \* @version 1.0
 \* @author Prof. David Bernstein, James Madison University
 \*/
 public class Pile
 {
 private IntNode last;

 /\*\*
 \* Construct a new (empty) Pile
 \*/
 public Pile()
 {
 last = null;
 }
 /\*\*
 \* Push an int onto this Pile
 \*
 \* @param anInt The int to push
 \*/
 public void push(int anInt)
 {
 IntNode temp;

 temp = new IntNode();

 temp.value = anInt;
 temp.next = last;

 last = temp;
 }
 }

/\*\*
 \* A Quack of objects
 \*
 \* This implementation uses linked memory
 \*
 \* @version 1.0
 \* @author Prof. David Bernstein, James Madison University
 \*/
 public class Quack
 {
 private Node first, last;

 /\*\*
 \* Construct a new (empty) Quack
 \*/
 public Quack()
 {
 first = null;
 last = null;
 }
 /\*\*
 \* Pep an Object off of this Quack
 \*
 \* @return The Object
 \*/
 public Object pep()
 {
 Node temp;
 Object value;

 value = null;

 if (first != null)
 {
 value = first.value;

 if (first == last)
 {
 first = null;
 }
 else
 {
 temp = last;
 while (temp.next != first)
 {
 temp = temp.next;
 }

 first = temp;
 first.next = null;
 }
 }

 if (first == null) last = null;

 return value;
 }
 /\*\*
 \* Pip an Object off of this Quack
 \*
 \* @return The Object
 \*/
 public Object pip()
 {
 Object value;

 if (last != null)
 {
 value = last.value;
 last = last.next;
 }
 else
 {
 value = null;
 }
 if (last == null) first = null;

 return value;
 }

 /\*\*
 \* Push an Object onto this Quack
 \*
 \* @param anObject The Object to push
 \*/
 public void push(Object anObject)
 {
 Node temp;

 temp = new Node();

 temp.value = anObject;
 temp.next = last;

 last = temp;
 if (first == null) first = last;
 }
 }

/\*\*
 \* A driver that uses a Quack
 \*
 \* @version 1.0
 \* @author Prof. David Bernstein, James Madison University
 \*/
 public class QuackDriver
 {
 /\*\*
 \* The entry point
 \*
 \* @param args The command-line arguments
 \*/
 public static void main(String[] args)
 {
 Quack quack;
 Object pdOff;

 quack = new Quack();

 quack.push("Bob");
 quack.push("and");
 quack.push("Carol");
 quack.push("like");
 quack.push("Ted");
 quack.push("and");
 quack.push("Alice");

 System.out.println("\n\npip:\n");

 pdOff = quack.pip();
 while (pdOff != null)
 {
 System.out.println(pdOff);
 pdOff = quack.pip();
 }
 }
 }