MIDTERM EXAMINATION
CSC 212 ♦ FALL 2016

You may use one double-sided 8.5"x11" sheet of notes on this exam. You may not consult other sources of information. You will have the entire period (110 minutes) to complete your work. All work should be written in the exam booklet. Partial credit will be granted where appropriate if intermediate steps are shown.

Vocabulary (16 points)
Identify the technical term or concept corresponding to each of the descriptions given below.

a.) An operation in a Java program that specifies the type associated with a variable name.
b.) An operation in a Java program that earmarks a discrete piece of memory on the heap.
c.) An operation that copies a value into the memory storage associated with a variable.
d.) A way of copying a data structure where part of the structure is copied and part is shared.
e.) This is a special method that is responsible for initializing all the fields of a new object.
f.) A piece of memory reserved for the storage of all the nonstatic fields of a particular example of a class.
g.) Storage for multiple copies of a single data type, arranged as a sequential block in memory.
h.) The term that describes a variable's value when it is actually the address of a block of memory on the heap.

Java Language (12 points)

The program below will run without errors, although it is not well written. Identify as many instances as you can find where the program below deviates from the style/programming guidelines for this class. If a particular shortcoming appears more than once, list it once and note how many instances you count.

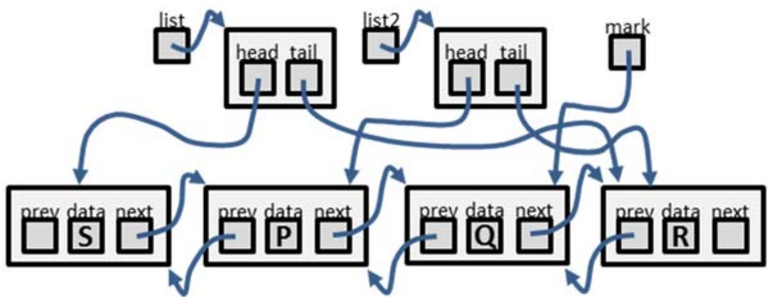
```
import java.awt.*;
public class BadCode {
    public static final double defaultradius = 1.0;
    public double qqq;
    public double ComputeAreaOrVolume(int dim) {
   if (dim == 2) return Math.PI*qqq*qqq;
   return 4*Math.PI*qqq*qqq*qqq/3;
    }
    public static void main(String[] args) {
   for (int i = 0; i < args.length; i++) {
       double qq;
       try {
      qq = Double.valueOf(args[i]);
       } catch (Exception e) {
      continue;
       }
       BadCode ggg = new BadCode();
       ggg.qqq = qq;
       System.out.print(ggg.ComputeAreaOrVolume(2)+" ");
       System.out.println(ggg.ComputeAreaOrVolume(3));
   }
    }
}
```

Data Structures (8 points)

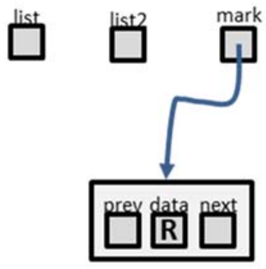
Identify a data structure we have studied that is suggested by each of the scenarios below.

a.) Train cars being rearranged and assembled into trains in a switching yard.
b.) Milk cartons in the dairy case at the supermarket (loaded from the back).
c.) People waiting to go on a ride at an amusement park.
d.) Tree rings in a core sample, moving from the bark inwards.
e.) Characters in a word processor, where the user can edit, add, and delete anywhere in the document.
f.) Parking spaces along the side of a road.
g.) Coats piled on a bed as people arrive at a party.
h.) Desks in a classroom with fixed seating.

Linked Lists (16 points)
Consider the diagram below. In the questions that follow, draw a diagram that shows the changes to the picture, assuming that each one begins with the configuration shown below. Do not draw any memory objects that are no longer accessible. An example is done for you.



a.) mark = list2.tail; mark.prev = null; list1 = list2 = null;



b.) list.head.prev = list.tail; list.tail.next = list.head; list2 = null; mark = null;
c.) mark.next.prev = mark.prev; mark.next.prev = mark.prev; mark = null; list2 = null;
d.) list.head.next.next.next = tail.prev.prev.prev.prev; list.tail = list2.tail = mark;
e.) list = list2; list2.head = list.head.prev; list1 = null;

Stacks and Queues (16 points)
It is often useful to make the distinction between the abstract interface of a data structure and its specific implementation. This problem takes this principle to an extreme, using stacks to implement a queue interface, and a queue to implement a stack interface.

Consider the two incomplete classes below (with deliberately omitted Javadoc). Based on the parts of each class already written, give an implementation of the two missing methods. You may write Java code directly, or give detailed pseudocode. When you are done, **explain in words how/why each implementation works.**

```
import java.util.*;
/** Implements stack operations using a queue as the underlying storage
*/
public class StackFromQueue<E> {
    private Queue<E> q = new LinkedList<E>();
    private int numberStored = 0;
    private void cycle(int n) {
         for (int i = 0; i < n; i++) {
             q.add(q.remove());
         }
    }
    public void push(E data) {
         q.add(data);
         numberStored++;
    }
    // Write the implementation of pop
}

import java.util.*;
/** Implements queue operations using a stack as the underlying storage
*/
public class QueueFromStack<E> {
    private Stack<E> stackA = new Stack<E>();
    private Stack<E> stackB = new Stack<E>();
    private void transfer(Stack<E> from, Stack<E> to) {
         while (!from.isEmpty()) {
             to.push(from.pop());
         }
    }
    public void in(E data) {
         stackA.push(data);
    }
    // Write the implementation of out
}
```

Class Design (16 points)
Suppose that you are writing java code to implement a weather station app. Your code will work with a line of home weather station hardware kits; each kit may or may not include various pieces of equipment, the details of which are not important for this question. The hardware comes with a set of library functions that your program can call to collect data from the hardware. Your program should display the current status of the various weather instruments, updating itself periodically. There will also be a simple interface to allow the user to change display from Celsius to Fahrenheit, etc.

Discuss the class structure you would set up to implement this project. What classes would you need to create, and what would their jobs be? Answer this in words, not in code. As you write, try point out how your answer exemplifies any of the programming design concepts we have discussed in class.

Graphical User Interfaces (16 points)

Consider the fragment of code below for setting up an interface with three buttons, and the listener class definition that follows. Assuming that these objects are set up as shown, what would be the expected output from pressing the buttons in the following sequence: B1, B2, B3; B1, B2, B3? (Note: when a component has more than one listener registered, the most recently registered listener runs first.)

```
JButton b1 = new JButton("B1");
JButton b2 = new JButton("B2");
JButton b3 = new JButton("B3");
TestListener h1 = new TestListener();
h1.output = "X";
b1.addActionListener(h1);
TestListener h2 = new TestListener();
h2.output = "Y";
b2.addActionListener(h2);
h1.output = "Z";
b2.addActionListener(h1);
h2.edit = h1;
b3.addActionListener(new TestListener());
```

The piece above would appear within a setup method. Below is the listener class definition:

```
public static class TestListener implements ActionListener {
      private String output = "A";
      private TestListener edit = this;
      public void actionPerformed(ActionEvent e) {
            System.out.println(output);
            edit.output = edit.output+output;
      }
}
```