MIDTERM EXAMINATION
CSC 212 ♦ FALL 2013

You may use one 8.5"x11" sheet of notes on this exam. You may not consult other sources of information. You will have the entire period (80 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)
Consider the short program below to answer the questions that follow. (The line numbers are for reference but are not part of the program.)

```
1     public class SphereVolumes {
2         public static double volume(double r) {
3             double result = (4.0/3.0)*Math.PI*r*r*r;
4             return result;
5         }
6         public static void main(String[] args) {
7             for (int i = 0; i < args.length; i++) {
8                 double r, v;
9                 r = Double.valueOf(args[i]);
10                v = volume(r);
11                System.out.println("Radius="+r+", volume="+v+".");
12            }
13        }
14    }
```

a.) List all tokens appearing in the program above that are names of classes.
b.) List all tokens appearing in the program above that are fields of a class.
c.) List all the local variables used in the main method.
d.) List the numbers of all lines containing method calls.
e.) List the numbers of all lines containing a variable declaration (excluding method parameters).
f.) List the numbers of all lines that perform memory allocation on the heap.
g.) List the numbers of all lines containing variable assignments.
h.) If the volume method had not been declared static, how could it be called from main? (Write a code fragment to replace line 10.)

Graphical User Interfaces (16 points)
Identify a class or interface from either the **java.awt**, **java.awt.event** or **javax.swing** packages that matches the descriptions below.

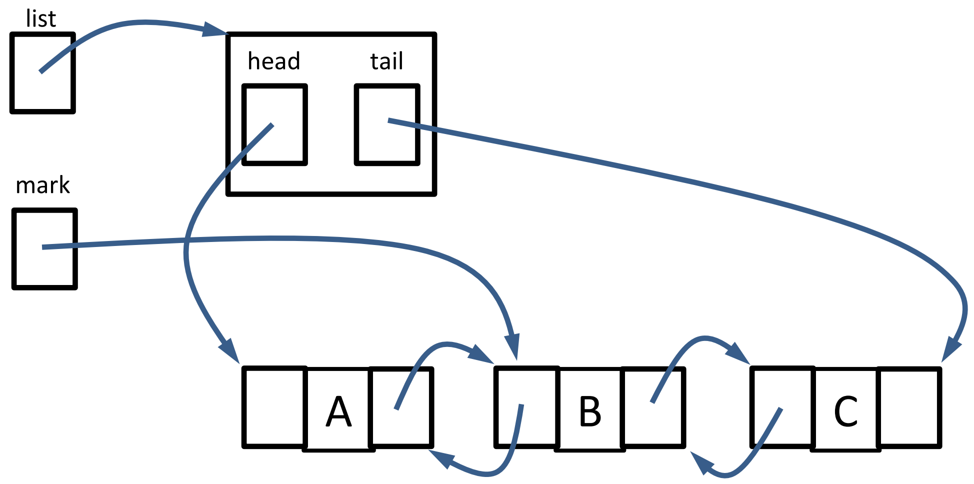
a.) Represents a top-level window with a title and border
b.) The parent class for all objects that can appear in a content pane
c.) Represents a hue in terms of its red, green, and blue components
d.) Holds the event handler for a clicked button
e.) Defines empty handler methods for mouse clicks, drags, wheel motions, etc.
f.) Can arrange window components in a 3x3 grid
g.) Allows you to fit one component layout within another in a GUI
h.) Provides methods for drawing lines, circles, rectangles, etc. within a given area of the screen

Stacks (16 points)
Consider the following pseudocode:

```
given string T, stack S initially empty
err = 0;
n ← length(T)
loop for i from 0 to n-1
   if i is less than (n-1)/2
      S.push(T[i])
   elseif i is greater than (n-1)/2
      if S.pop() not equal to T[i]
         err = err+1;
      endif
   endif
endloop
```

a.) If **T** = "REVERSE" what would be the value of **err** at the end?
b.) If **T** = "RACECAR" what would be the value of **err** at the end?
c.) Draw the contents of the stack after the completion of each loop iteration for **T** = "STOPS".
d.) If **T** = "POPS" and **S** is not empty but holds 'S' initially, what would be the value of **err** at the end?

Linked Lists (20 points)
Consider the diagram below, which visualizes the memory structure of a doubly linked list of the sort developed in class. The fields shown in each node are from left to right **previous**, **data**, and **next**. For each part below, draw the state of the memory structure as it would look after the commands given. If an exception would occur then you should draw no picture but identify the line that would cause the exception. Finally, for all the items where you have drawn a picture, state whether or not the structure referred to by **list** represents an internally consistent structure for a linked list.



a.) list.tail = list.head = null;
b.) list.tail = mark; list.head.next.next = null;
c.) mark.next.previous = mark.previous; mark.previous.next = mark.next;
d.) mark.next = mark.previous = null;
e.) list.head = list.head.next; list.head.previous = null; list.head.previous.next = null;

Complexity Analysis (12 points)
The method below implements bubble sort on an array of integers. How many times would the comparison inside the inner loop be executed? Give an exact formula in terms of **n**, the array length, and also relate this to a standard complexity class (big-O bound).

```
    public static void sort(int[] arr) {
        for (int i = 1; i < arr.length; i++) {
            for (int j = 0; j < arr.length-i; j++) {
                if (arr[j] > arr[j+1]) {
                    int tmp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = tmp;
                }
            }
        }
    }
```

Consider the version below, which follows the same implementation using a **LinkedList<Integer>**. What, if anything, will change about the complexity analysis as compared to the case above?

```
    public static void sort(LinkedList<Integer> list) {
        for (int i = 1; i < list.size(); i++) {
            for (int j = 0; j < list.size()-i; j++) {
                if (list.get(j) > list.get(j+1)) {
                    int tmp = list.get(j);
                    list.set(j,list.get(j+1));
                    list.set(j+1,tmp);
                }
            }
        }
    }
```

Class Design (20 points)
Write a few short paragraphs describing the role of constructors, accessors, and manipulators. Include in your discussion their relationship to the fields of a class, and the considerations that apply when deciding which constructors, accessors, and manipulators should be included in the class definition. Refer to specific examples from class where possible.