CSC 112 MIDTERM EXAM
FALL 2006

You will have 110 minutes to complete this exam. All work should be written in the exam booklet. Start with the questions that you know how to do, and try not to spend too long on any one question. Partial credit will be granted where appropriate. Good luck!

1. **Class Design** (12 points). Suppose that you are designing a program that will allow users to display, enlarge, rotate, and crop an image. The following classes are envisioned:

> **PhotoGUI**: Manages the GUI, including a text box that can load a new photo, and buttons to rotate left or right. The user can also drag the mouse to select a rectangle in the image area. In this case, a new window will appear with an enlarged view of the selected area.
>
> **PhotoView**: Displays a view of some specified portion of a photograph.
>
> **PhotoData**: Contains image data and other information about a photograph.

a.) Assuming that event handling will be processed using private nested classes as demonstrated in class, in which class should these nested classes be placed? Which events must be handled to achieve the described behavior?

b.) Explain the advantages of using separate **PhotoView** and **PhotoData** classes. In what circumstances might there be more instances of one class than the other?

c.) Which class should have a method that will determine the layout of the visible components in one of the program's windows?

2. **Programming Style** (12 points). Programming languages contain many features designed to eliminate the need for redundant code (i.e., code that is essentially the same except for minor differences). Give three examples of such mechanisms in Java, and three advantages that result from taking advantage of them.

3. **Sorting** (16 points). Consider the list of numbers below. Show the order of the numbers after **each** swap performed by the algorithms that follow.

> 5, 6, 2, 3, 4, 1

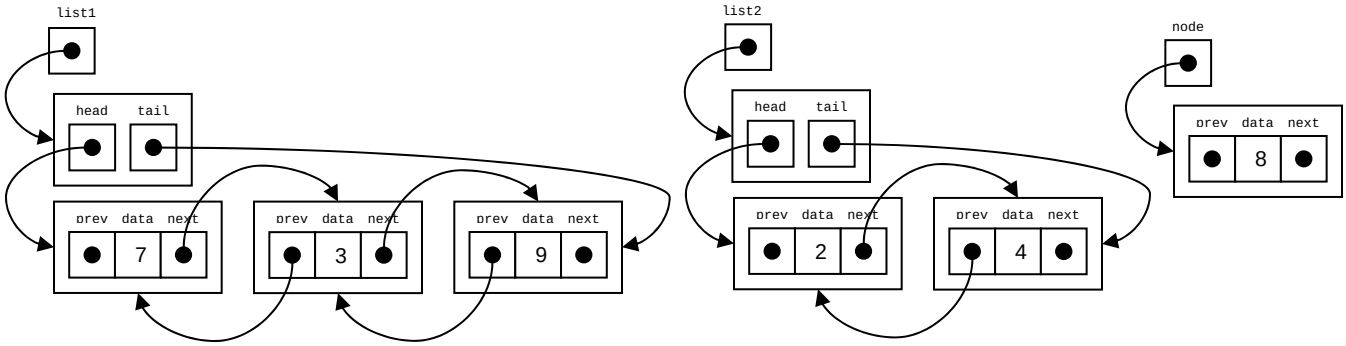
a.) Selection sort, array implementation, growing the sorted region from left to right.

b.) Insertion sort, array implementation, growing the sorted region from left to right.

4. **List Operations** (20 points). Consider the list structures shown in the figure below, and the operations that follow. Assume for this problem that all fields are public. For each part, you have two tasks.

I. Draw the changes to the structures made by the code.
II. Indicate that the results are all valid structures, or give a reason why they are not. Identify any pieces that would be eligible for garbage collection.

Assume that each part below begins with the state shown in the figure (changes are not cumulative). You only need to draw the structures that have changed.



a.)
```
list1.tail.next = list2.tail;
list2.tail.prev = list1.tail;
list2.head = list1.head;
list1.head = list1.tail = null;
```

b.)
```
Node mark = list1.head.next;
mark.next.prev = node;
node.next = mark.next;
mark.next = node;
node.prev = mark;
```

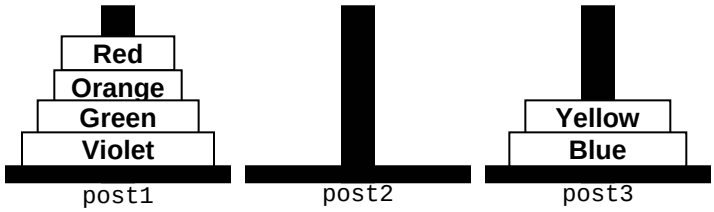
c.)
```
for (Node item = list1.head; item != null; item = item.next) {
    item.next.prev = null;
    item.next = null;
}
```

5. **Programming Practice** (12 points). Java includes a number of qualifiers that can be added to fields and methods, including `private`, `static`, and `final`. These qualifiers have specific effects within a program, are provided to help programmers achieve certain goals. Four each of the three qualifiers just mentioned, describe (a) the practical effect of including it before a field of a class, and (b) what motivation might cause a programmer to use include such a qualifier. (In other words, what does the qualifier do, and why would the programmer want to do that?)

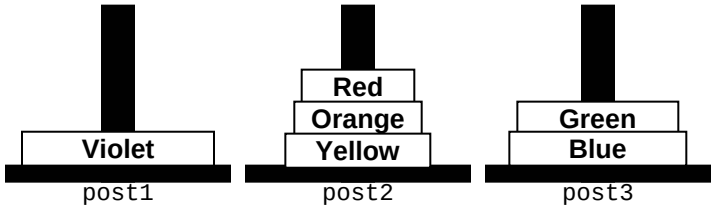
6. **Program Simulation** (12 points). Simulate execution of the following program. What would be its output?

```
public class Shadow {
    static int x = 0;

    static void method1(int x) {
        System.out.println(x);
        x = 1;
        System.out.println(x);
    }

    static void method2() {
        int x = 2;
        System.out.println(x);
    }

    static void method3() {
        System.out.println(x);
        x = 3;
        System.out.println(x);
    }

    public static void main(String[] args) {
        System.out.println(Shadow.x);
        int x = 4;
        Shadow.x = 5;
        System.out.println(x);
        System.out.println(Shadow.x);
        method1(x);
        System.out.println(x);
        method2();
        System.out.println(x);
        method3();
        System.out.println(x);
        System.out.println(Shadow.x);
    }
}
```

7. **Stacks** (8 points). There is a child's toy called the Towers of Hanoi consisting of colored plastic rings stacked upon a central post. The toy acts as a stack, since rings must be inserted or removed in LIFO order. Suppose that the following situation is simulated in a program using three stacks, (**`post1`**, **`post2`**, and **`post3`**).



If the initial state of the stacks is as shown above, can you devise a sequence of moves that will culminate in the desired goal state below? (For example, **`post2.push(post1.pop)`** would move the red ring to the middle post.)



8. **Generic Programming** (8 points). Write a single generic method that would replace the two shown below.

```
static void shift(Integer[] x) {
    Integer tmp = x[0];
    for (int i = 1; i < x.length; i++) {
        tmp[i-1] = tmp[i];
    }
    x[x.length-1] = tmp;
}
static void shift(String[] x) {
    String tmp = x[0];
    for (int i = 1; i < x.length; i++) {
        tmp[i-1] = tmp[i];
    }
    x[x.length-1] = tmp;
}
```