NOTE: This is intended to be harder than the actual exam. If you feel comfortable with this, you should be okay with the exam.

Overview

Abstract classes and interfaces
- Explain what the difference is between an abstract class and an interface (though you don't have to necessarily know when to use which)
- Write Java classes that extend abstract classes and implement interfaces
- Identify when errors arise in overriding abstract classes and implementing interfaces (i.e., you must implement all abstract methods).

Inner classes
- Apply scoping rules in tracing code involving inner classes
  - if an inner class uses a variable, be able to state *which* variable it refers to--i.e. in which class, and when it would cause an error
- Explain the general difference between inner classes and subclasses.
- Explain how we have used inner classes in building our event-driven GUIs.

Java graphics and event-driven programming
- Write code that uses listeners to implement event-driven GUIs. This includes writing listeners and registering them with the appropriate GUI components.
- Identify errors in code that implements event-driven GUIs
  - forgetting to register a listener, implementing the wrong kind of listener, etc
- Trace code that uses listeners to implement event-driven programming.
- Explain when methods get called in response to different events.
  - if the user clicks on a button, how do you know what method gets called?

Recursion
- Write simple recursive functions
- Identify base cases and recursive steps for simple recursive problems
- Trace recursive methods
  - Draw and count stack frames as recursive methods run
Example Questions

Recursive Question 1
int myFactorial(int integer)
{
    int recRes=0; //recursive result
    if(integer == 1)
        return 1;
    recRes=myFactorial(integer-1);
    System.out.println("my integer is:" + integer +" my recRes is:"+recRes);
    return integer*recRes;
}
> System.out.println("3! = "+myFactorial(3)); //called from main

1. In the above code, What is the base case in the above code? If there is no base case, please describe what the base case should be.
2. Why do we need a base case at all?
3. What is the recursive call?
4. Assuming the base case is now correct (either because it was there or you added it) draw the run-time stack with all the local variables (remember the parameters are considered local variables) when myFactorial(3) is called from main as shown above
5. What gets printed when running myFactorial(3) from main (keep in mind the recursive function has a println too)

Recursive Question 2
Write a recursive method (no loops!) that prints every element in a given array backwards on a new line (ie. use System.out.println for all printing). For example the array {1,2,3,4} will print
4
3
2
1
We have started this method for you below:
public void printArrayBackwards(int arr[], int currentIndex){
class MousePanel extends JPanel
{
    private int clickCounter;
    MyMouseListener implements MouseListener
    {
        public void mouseClicked(MouseEvent e)
        {
            System.out.println("Mouse has been clicked: "+ (++clickCounter));
        }
        public void mousePressed(MouseEvent e){}
        public void mouseReleased(MouseEvent e){}
        public void mouseEntered(MouseEvent e){}
        public void mouseExited(MouseEvent e){{}
    }
    }
    MousePanel()
    {
        //add code to add the MyMouseListener as the MouseListener for this JPanel
    }
}

1. Fill in the needed code in the MousePanel constructor to add a new MyMouseListener as the MouseListener for the MousePanel (whew that was a Mouse full...err Mouth full)
2. When does the mouseClicked(MouseEvent e) get called (ie what event causes it)?
3. Micky decides to add a MousePanel to his JFrame. When he runs his program, what the value of “clickCounter” after the first time he clicks on the MousePanel
4. How is it that MyMouseListener can access MousePanel's private instance variable clickCounter?
1. Does this compile? If not what needs to be fixed?
2. Assuming you've correctly fixed any issues. What does main print?
3. On line 8 we assign a value to variable "y". On what line is “y” declared? Which class is “y” a member of?
4. On line 21 we assign 4 to “outer.inner.x”. On what line is the “x” variable declared?
5. Do non-static inner classes always have access to their outer classes member variables?
6. Do outer classes always have access to inner class’ variables?
7. Why do we need to create a new InnerClass object on line 15 above?