Exam-3: 95/100

Instructions stated to create one instance of an anonymous class that implements ChangeListen; instead, you created three instances (one for each slider) of a *named* class.

A3 (sliderListener).

B1. C1.

The GUI looks and functions exactly as required.

Homework-3: 100/100

Unused fields in Color class. Also, I think a spinner rather than a text field would be an easier and more natural way to specify the distance metric.

Wow...the force is strong with this one. Very impressive! I was not familiar with the Delaunay triangularization. Excellent job researching and coding.

Good object-oriented design. Documentation is clear and complete.

I am trying to complete the grading so I did not take time to figure out what you did to render the diagram so quickly. The painting is faster than I would have thought possible, given the pixel-by-pixel approach that we were taking. If you get a chance, please talk to me about this.

Homework-2: 92/100

I see that you have figured out how to use the visual editor. That's good, although I did not expect anyone to do this. In future assignments, check with me first before you do. The reason is that it is harder to evaluate student work that was partly written by the visual editor; also, using the visual editor a programmer can create code that works but which he does not really understand.

I don't understand what the MainMenu class is for. It's a main class, so I ran it; but all that appears is a frame with two text fields into which the grid dimensions are entered – there is no button or other control to make the selection. Nothing happens when entering this data, as far as I can tell. When I run the other program (MemoryGameGUI), I can play, although some features are missing.

- **Coding style and documentation: B1. B3. D3. D1.** I am a bit surprised that you are making these mistakes at this point. I don't think it's a question of understanding – it's a question of attention to detail. Remember, in software development, the details matter. Also: **B2!**

- **Functionality**

    - **Current game time:** ✓

    - **Best game time:** missing

    - **Duration control:** ✓

    - **Grid size options:** missing

    - **Component(s) not used in class:** slider, box layout, icons, images.

The audio does not work on my system. The application throws an exception:

```
Prism-ES2 Error : GL_VERSION (major.minor) = 1.4Exception in thread "AWT-EventQueue-0" MediaException: UNKNOWN : com.sun.media.jfxmedia.MediaException: Could not create player! : com.sun.media.jfxmedia.MediaException: Could not create player!
```

This may be a Linux problem – I may have to install some extra libraries. I want to give you the benefit of the doubt in assuming that the audio worked for you on your system, but since you did not document any of the missing features of your solution I have no way of knowing.

- **Object-orientation:** It looks like you did a good job in separating the code that manages the game state from the code that presents it to the user (i.e., the GUI), but see comments below.

In MemoryGame, you have two static fields to which you gave package access. Why are they not private. Also, why are they *static*? Wouldn't each instance of the game have its own delay?

Public instance variables, as in the Tile class, defeat the point of object-oriented programming. In doing this, you can no longer guarantee that Tile objects are in a meaningful state because external code can directly manipulate the state without going through the public interface of the class. It also makes the MemoryGame class dependent on the underlying implementation of Tile objects. You need to ask yourself this question: are Tiles and MemoryGames really two aspects of the same thing? If so, they should be expressed by a single class. But I think that is not the case here. They represent distinct concepts. So if MemoryGame objects need access to Tile data, the Tile class should provide suitable and carefully written public methods to provide this access.

Excellent coding on the whole, but a few documentation and conceptual problems as noted above.

Exam-1: 97/100

Cube: 20/20

B1 (even on exams).

Make those instance variables private!

Counter Game: 77/80

B1.

B6 (variables b1, b2, b3).

There is a lot of duplicated code. It could be factored out to simplify things. You do not need three separate classes implementing the *ActionListener* interface – one will suffice.

Instructions said to change the button text color to red, not background color.

Homework-1: 97/100

Public instance variables defeat the point of object-oriented programming. Why do you need those variables to be accessible to other classes? A class is supposed to encapsulate data and behaviors within a protective wall, so to speak, so that the designer can ensure that objects are always in a meaningful state.

Since DELAY is final, it would make more sense to declare it static.

Aside from the considerations mentioned above, the code and documentation are excellent. The application works perfectly.