# CS 1181 Project 2: GUI-based Game

**PURPOSE:** The primary goal of this project is to develop a more in depth understanding of object-oriented programming by developing a project in an object-rich environment. Java Swing serves as an ideal platform where you can develop and demonstrate mastery of object inheritance, class contracts, multiple inheritance (via interfaces), event processing, and (eventually) concurrency. This project will require the development of a simple game using Java.

**PROBLEM:** The project will be a simple game of your choice. It can be a novel game or your own take on a classic. You have a great deal of freedom in the design of this game. You can do something as simple as "concentration". You can do something more complex including multiple levels, sophisticated images, simple AI, and/or sound. Your only constraints are your knowledge and available time. A (small) portion of your overall grade will be based on a subjective measure of how your game stacks up to the other students'. Your game MUST meet, as a minimum, the specifications listed below.

## DESIGN REQUIREMENTS

0. If you have done this project previously (for example because you are re-taking this course) you **must** implement a different game. Duplicate submissions will not receive any credit. Note that the rubric has been revised since last time.

1. There will need to be a game window implemented as a JFrame.

2. Your game must **provide instructions** to the player.

3. You must have at least two different visual elements (buttons, labels, text fields, etc.) – at a minimum!

4. There must be a way for the user to interact with the game that causes some visual elements to change their appearance. For example, clicking on a button that represents a playing card could flip the card over to show its value, or correctly typing in the answer to a riddle could cause a smiley face to appear.

5. Your game should have some sort of score or goal. The level of success/total points should be displayed.

6. Be sure to design your game using good object-oriented coding principles. Make appropriate use of collections like arrays or ArrayLists and loop over them to avoid unnecessary code repetition. If relevant, consider storing data needed by your game in a file so that it can be modified easily.

The specifications above are fixed. The example below is just an *example*. You can feel to use the example requirements below or develop your own… so long as they meet the requirements above.

**ADVICE:** Be careful not to bite off more than you can chew on this assignment — start by getting an extremely basic game working and then work on making it more advanced if you have time. Focus on good software design principles — have a logical division of program functionality into classes and methods.

**ACADEMIC INTEGRITY:** You are welcome to search online or ask others for help in implementing small pieces of your game, but the overall product must be your original work. For example, maybe you would like to play a music clip if the user wins your game. You could look up "how to play a music file in Java" and use the four or five lines of code you find in your program. Be sure to include a comment indicating where you got that from. What is not ok is to google something like "how to write tic-tac-toe in Java" and use the 100+ lines of code you find in your program. Think of it like writing a paper — it's fine to use a dictionary or a thesaurus to find a word you need, but it's not ok to take most of your paper from something you find online. If you have any confusion about whether or not something is allowed, ask me first.

**EXAMPLE**

Use the sample screenshots below as a guide for understanding the project. These screenshots are from a barebones implementation of the project requirements. If you have a hard time thinking of your own game idea in the time provided, feel free to base your project off of this example – however, you are not required to build the game below. This is just an example of a game could meet the project requirements if correctly implemented.

This example game is called Concentration (or sometimes Memory). The player flips over two cards at a time in an effort to find matched pairs. Once a matched pair is found, those two cards are removed from the game board. The goal is to clear all the cards from the game board in the smallest number of moves possible. The images above show, respectively, the game instructions, the beginning state of the game board, the state of the board in the middle of a turn, and the state of the board when the game is over.
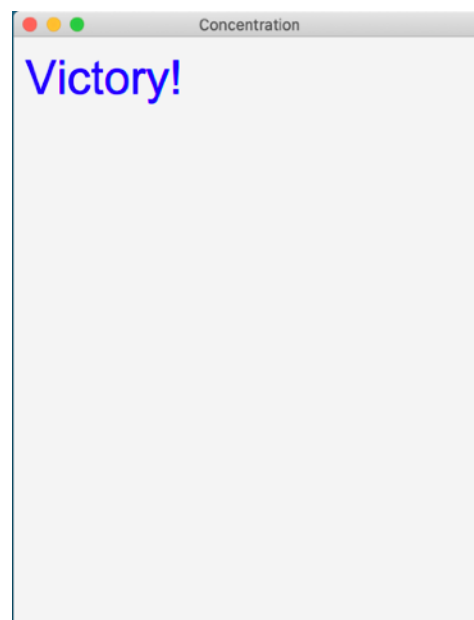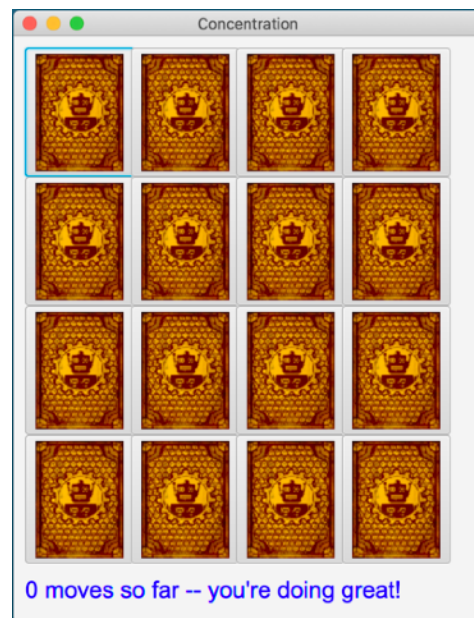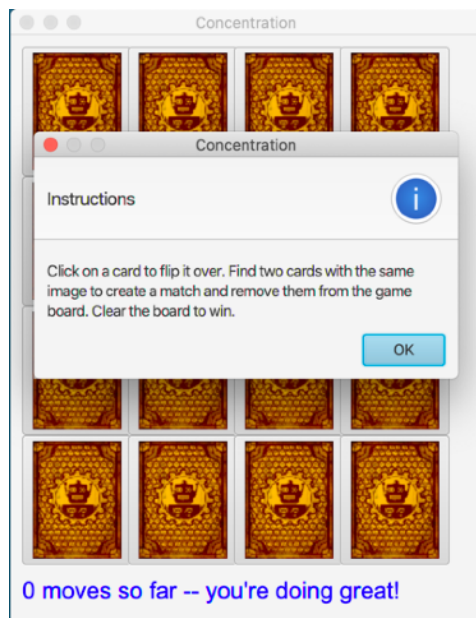
## GRADING

(60 pts) **Functionality**
>[5] Game 'Instructions' clearly communicated
>[20] Game window includes visual elements of at least two different types
>[10] At least one visual element is capable of movement/change
>[10]  User input updates the game in a meaningful way.
>[10]  The game calculates and displays some 'score' or other measure of success
>[5] Subjective coolness. This rating is subjective (based on a comparison with other projects).

(40 pts) **Style**
>[10] Good use of classes and inheritance. Classes are coherent and cohesive.
>[10] There is no unnecessary repetition of code.
>[10] Standard coding conventions are followed, including variable names and indentation.
>[10] The coding is ***meaningfully*** commented, and there is not commented out junk code

**NOTE:**  There is no valid excuse for turning in a program which does not compile.  A program will receive no credit if it does not compile and execute.  Partial credit *can* be earned on a functioning program that meets some, but not all, of the design requirements or partially solves the problem, but it must successfully compile and execute!  Test your final changes *before* you archive and submit your work.