Thomas O'Brien

CMSI 402

2/12/18

Homework Assignment #2

5.1. A component-based architecture is where the system is regarded as a collection of loosely coupled components that provide services for each other. These components serve specific tasks that are only ran when called upon. Note that in this kind of architecture, all of the code is contained within the same executable program and do not communicate over a network. A service-oriented architecture is similar to a component-based architecture, but instead utilizes services. Services are self-contained programs that run independently. Generally, services live on other machines are require communicating over a network to access them.

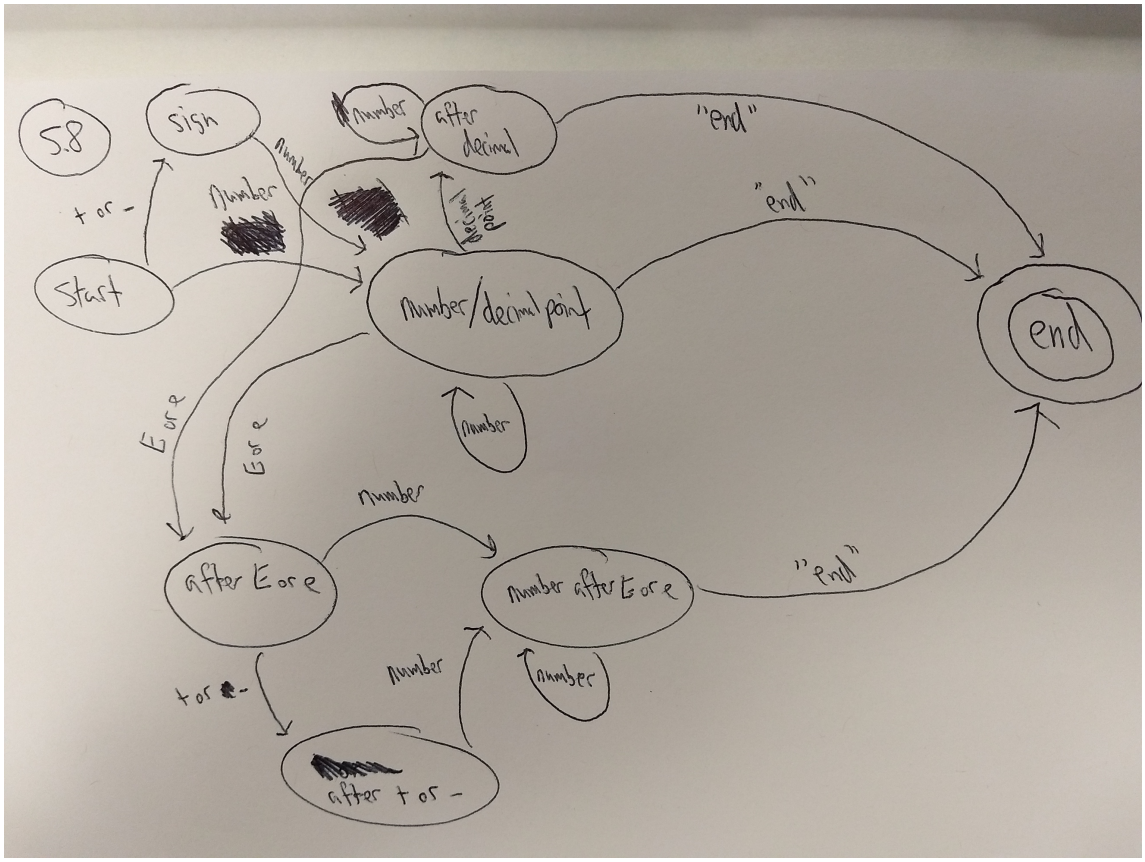
5.2. Given that this tic-tac-toe game only lets the user play against a simple computer and display their high scores that are stored on the phone in an internal database, a monolithic architecture appears to be a strong choice. A monolithic architecture is where a single program does everything that the application is capable of doing. The reason why this is a good choice is due to the scale of the application. Tasks like creating client-server connections and requesting data from services are unnecessary given the scope of the application. Developing an intractable tic-tac-toe board and a simple AI to play again the user can be contained in one application without risking multiple function calls overloading the system.

5.4. While implementing a chess game is not too much more complex than tic-tac-toe (aside from extra rules, unique pieces, etc), the additional feature of enables two users to play against one another over an Internet connection demands new requirements. While the application can still generally be a monolithic application, it will need to have a service-oriented architecture as

well. There will need to be a web service that allows two machines to send data from one another. A TCP connection is likely the best choice since the loss of packets could lead to the two separate machine's games becoming out of sync. The user interface and ruleset will not be influenced by this.

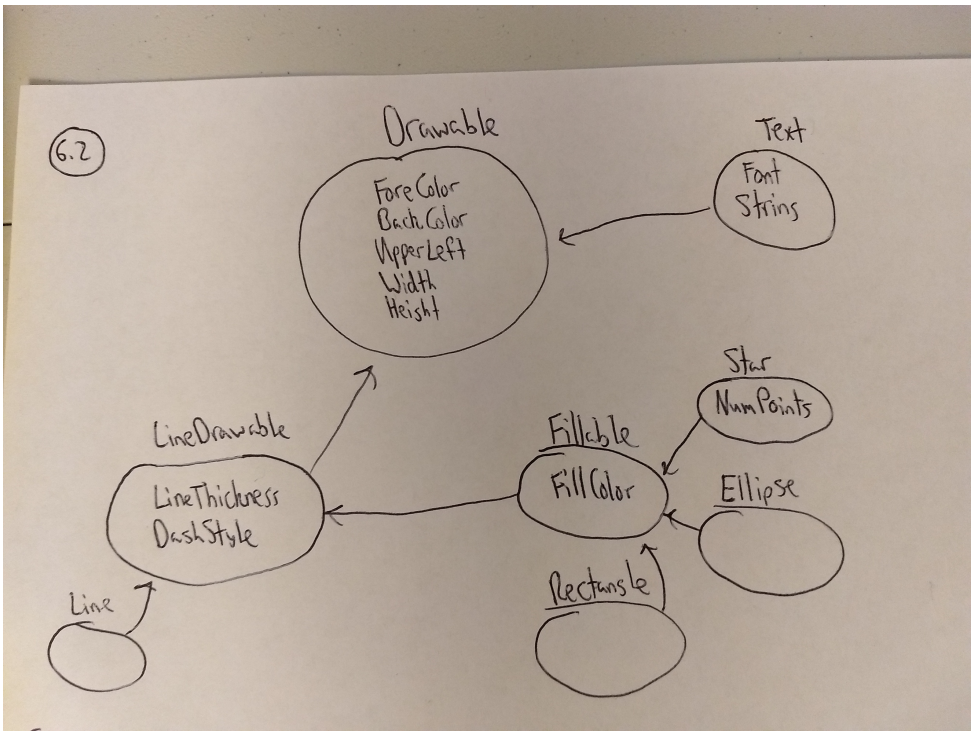
5.6. There is no need to set up a database for the ClassyDraw application. Since the application creates files for each drawing that the user chooses to save, it is best to use the operating system's file system instead. This allows the user to choose where files are stored and will not waste memory setting up a database for simple, unrelated files. In order to deal with random crashes, ClassyDraw could be programmed to create a hidden temporary file that stores a file that has not been "saved" by the user. If the user chooses to close the application and not save an unsaved file, then the hidden file will be deleted. Since this is only one file, it will not use a significant amount of memory.

5.8.

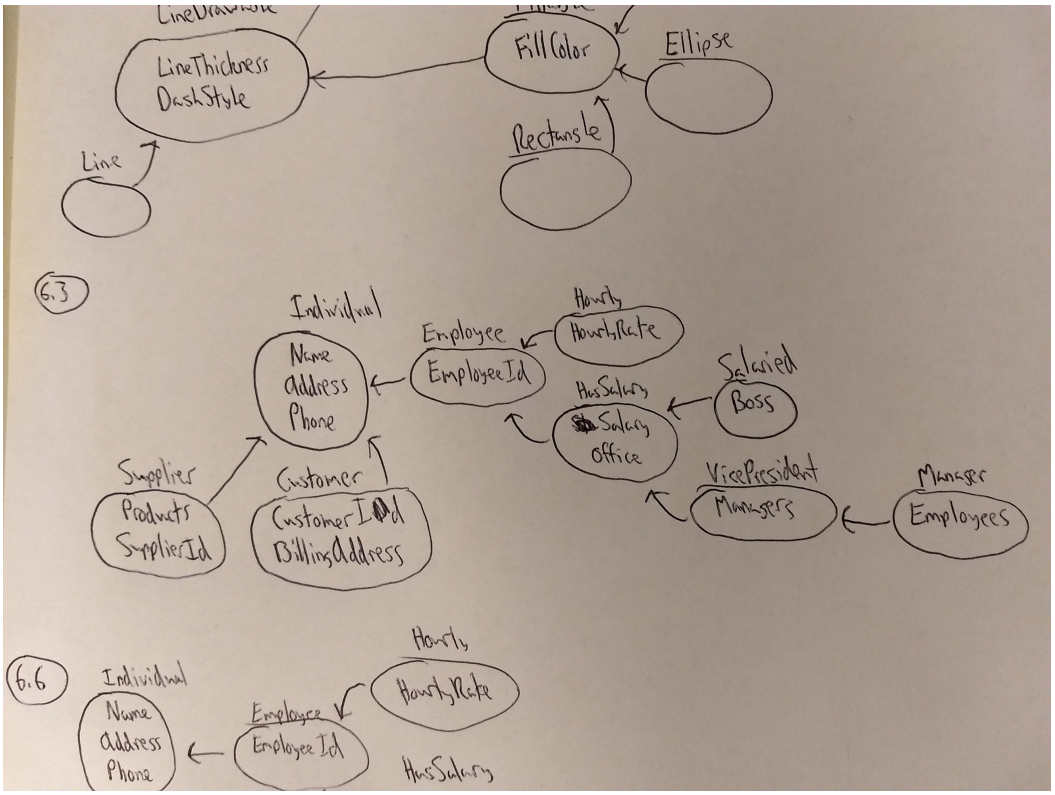


6.1. Some of the properties that all of these classes share are ForeColor, UpperLeft, BackColor, Width, and height. The properties LineThickness and DashStyle are only in Rectangle, Ellipse, Star, and Line. FillColor is in Rectangle, Ellipse, and Star. NumPoints is only in Star.Lastly, Font and String are in Text. It makes sense why some of these classes do not contain certain properties. For example, Rectangle would have no reason to have the properties Font and String. Also, it makes sense that Font and String are together since they directly relate to displaying words.

6.2. Note: the title of a class is above a circle. Within the circles are the properties of the class.



6.3. Note: the title of a class is above a circle. Within the circles are the properties of the class.



6.6. One could modify the HasSalary class to have 3 properties. These properties are Manager, Employees, and Salary. Manager is whoever the individual reports to. Employees is whoever the individual reports to. Salary is how much the person makes.