Justin Lee

CMSI 402

Homework #2

02/12/18

1. What’s the difference between a component-based architecture and a service-oriented architecture?

**In a service-oriented architecture, the pieces themselves are implemented as services.**

1. Suppose you’re building a phone application that lets you play tic-tac-toe against a simple computer opponent. It will display high scores stored on the phone, not in an external database. Which architectures would be most appropriate and why?

**A client/server architecture may be appropriate for this application as this architecture can have the application directly store its own data. In other words, the database is directly integrated into the application. Also, even though this architecture is limited in the sense that it doesn’t have the capability for multiple users to access the data, this application wouldn’t need that feature, as there’s only one user. A monolithic architecture for this application may also work too as it doesn’t require an external database. The data is stored in one program along with all the other functionality of the application.**

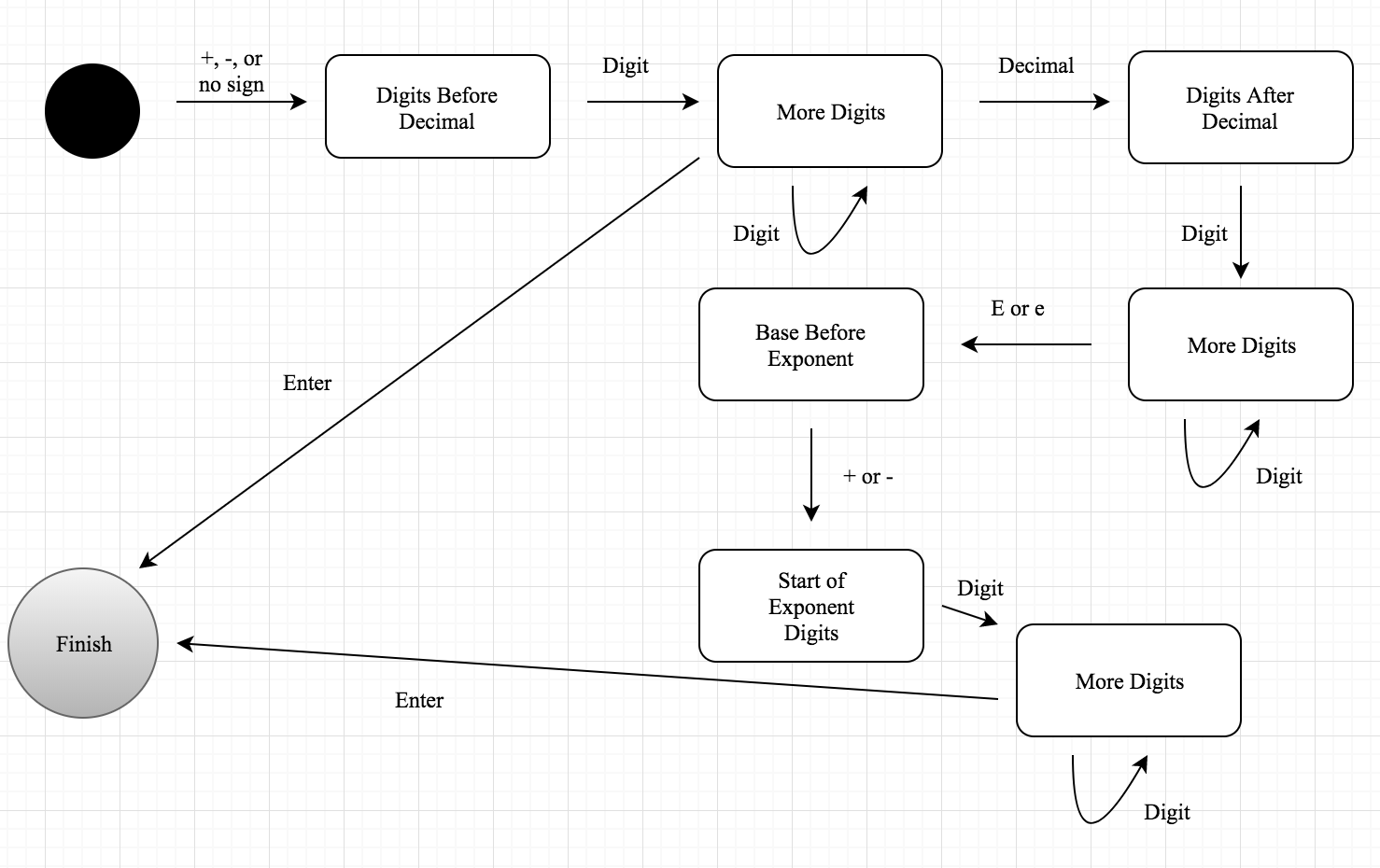
1. Repeat question 3 [after thinking about it; it repeats question 2 for a chess game] assuming the chess program lets two users play against each other over an Internet connection.

**It seems that a two-tier architecture would seem appropriate for this app. Now two users can use the same data, there are two instances of the user interface, and it uses a network, which enables the two players to connect to each other via an Internet connection and play against each other.**

1. What kind of database structure and maintenance should the ClassyDraw application use?

**It depends on the data that needs to be stored for this application. If the ClassyDraw application just stores text files that represent drawings, then some sort of text/document file database may work, assuming there are no relationships between text files (e.g. MongoDB). If there are relationships between files, then you may want to consider using a relational database (e.g. MySQL, Oracle, etc.) if those relationships are useful for the application. Assuming that there isn’t an emphasis on relationships, it may not be necessary for ClassyDraw. If the application involves many classes and object oriented programming, then an object database may be useful. An example is having multiple classes representing different objects such as rectangles, triangles, circles, etc. You may want to store information associated with those objects such as coloring, the manipulation of the object, and more. It’s very similar to an audit trail where you’re recording manipulations to data. Again it depends on the data that ClassyDraw needs to store in order to provide good functionality for the application. If there are a lot of manipulations with the data, it would be a good idea to reorganize the data for maintenance purposes and to have the application more efficiently. There’s a possibility of needing a data warehouse to store obsolete data. Another option is to simply delete the obsolete data if you really don’t have any use for the data at all.**

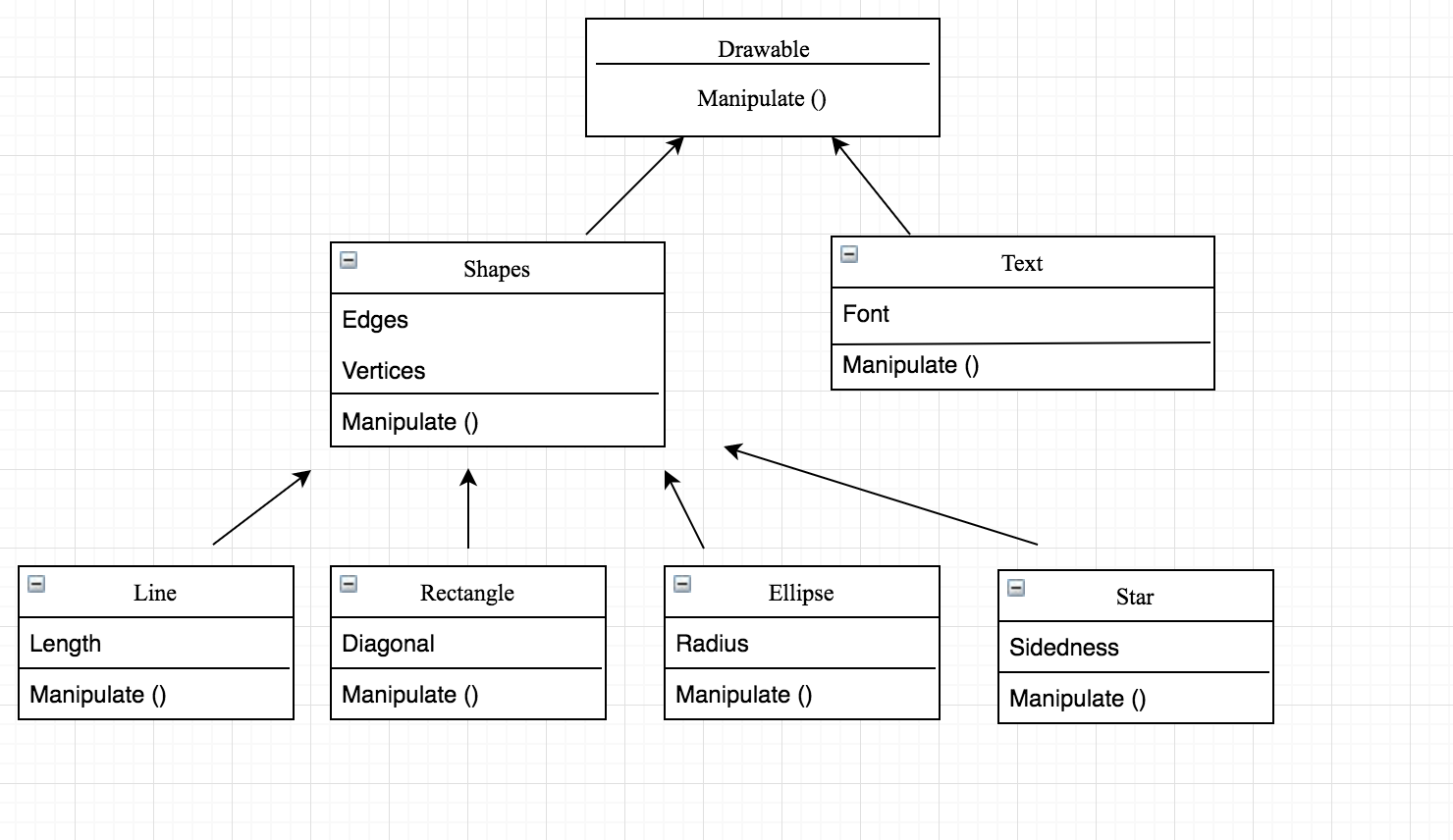
5.



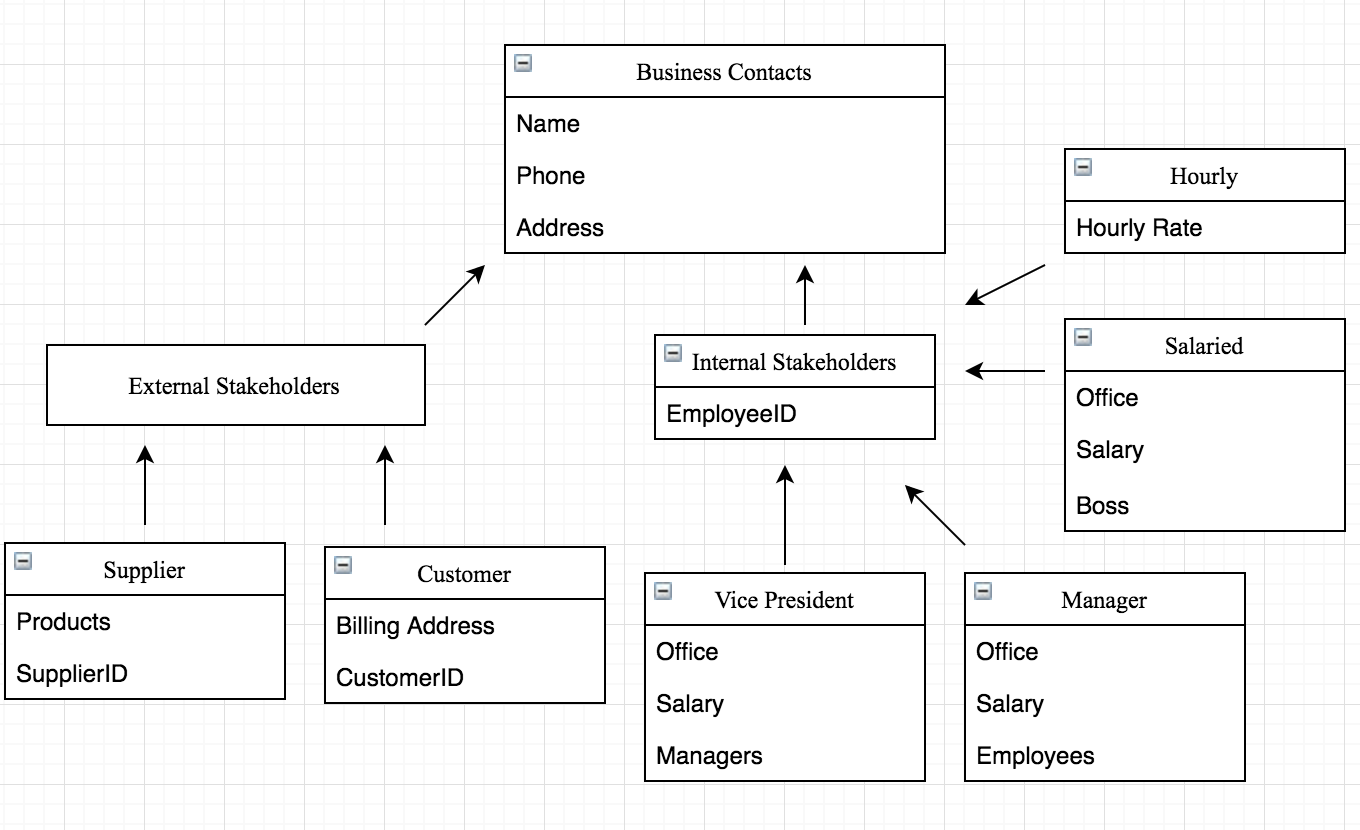
(Black Circle is the start. Couldn’t get an outlined circle to represent finished state)

1. Consider the ClassyDraw classes “Line”, “Rectangle”, “Ellipse”, “Star”, and “Text”. What properties do these classes all share? What properties do they not share? Are there any properties shared by some classes and not others? Where should the shared and nonshared properties be implemented.

**All of these classes are items that will be placed on the drawing board/canvas. All of these classes are also items that can be manipulated. At the same time, these classes all have properties that are specific to their classes. For example, the line property may have a length property that measures the length of the line. The rectangle class may have a diagonal property that gives the vertices and edge of a diagonal within the rectangle. The ellipse class may have a radius property that gives the length of the radius. The star property may have a property that records the number of points in the star. The text class may have a font property that returns the font text. The point is that all of these classes have potential properties that are specific to that class and differ from other classes. The line, rectangle, ellipse, and star class can all share a vertices property while the text class can’t have a vertices property. The classes with a shared property could each be connected to one parent class. This is known as generalizing the class. As for classes with different properties, it depends on whether the properties illustrate significant behavioral differences between the classes. If that’s the case, then branching off into different subclasses may be a good idea to distinguish class differences. This is also known as refining the class. However, it’s important to not go overboard with refining or generalizing because it can create complex and unnecessary hierarchies.**



8.



9.

