Mathew Sanders

CMSI402 Hw 2

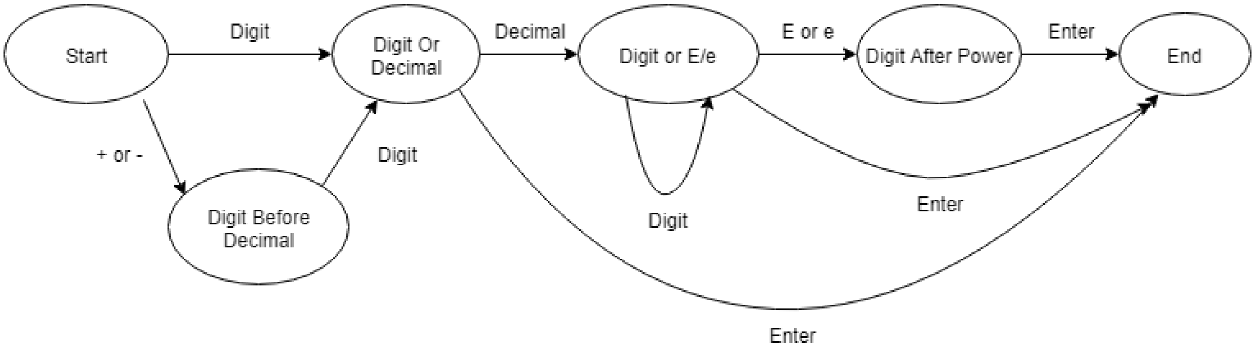
2/12/2018

5.2: A monolithic architecture would be the best architecture to use for this tic-tac-toe application. Since the data for the app is stored on the phone, there is no need for any outside communication by the app which means a client-server application would be wasteful. The app doesn't require much flexibility, only needing to update high-scores whenever broken, and otherwise, the app will follow simple rules that won't change. Since these rules of the game are understood beforehand, the app can be written tightly together from the beginning without need for much maintenance which could cause problems later in a monolithic architecture

5.4: A chess application would be run best on a distributed, client-server, rule-based architecture. Both players of chess would need to have access to the online game on their separate machines but neither individual machine should handle recording moves and sending it to the other machine. Instead, both machines would send moves to the centralized server which uses the chess rules to determine whether or not a move is legal. If legal, the move is sent to both computers and the turn changes to the other player.

5.6: ClassyDraw should use an Object based database. Since you will be manipulating and creating drawing objects, it makes more sense to store each object separately in the database to have easy access and use. These files should be stored in the class files and regularly indexed to ensure fast access when a user is trying to draw something. User creation files should be stored wherever the user specifies, but a temporary file can be saved in cases of needed recovery, similar to Microsoft Word.

5.8:

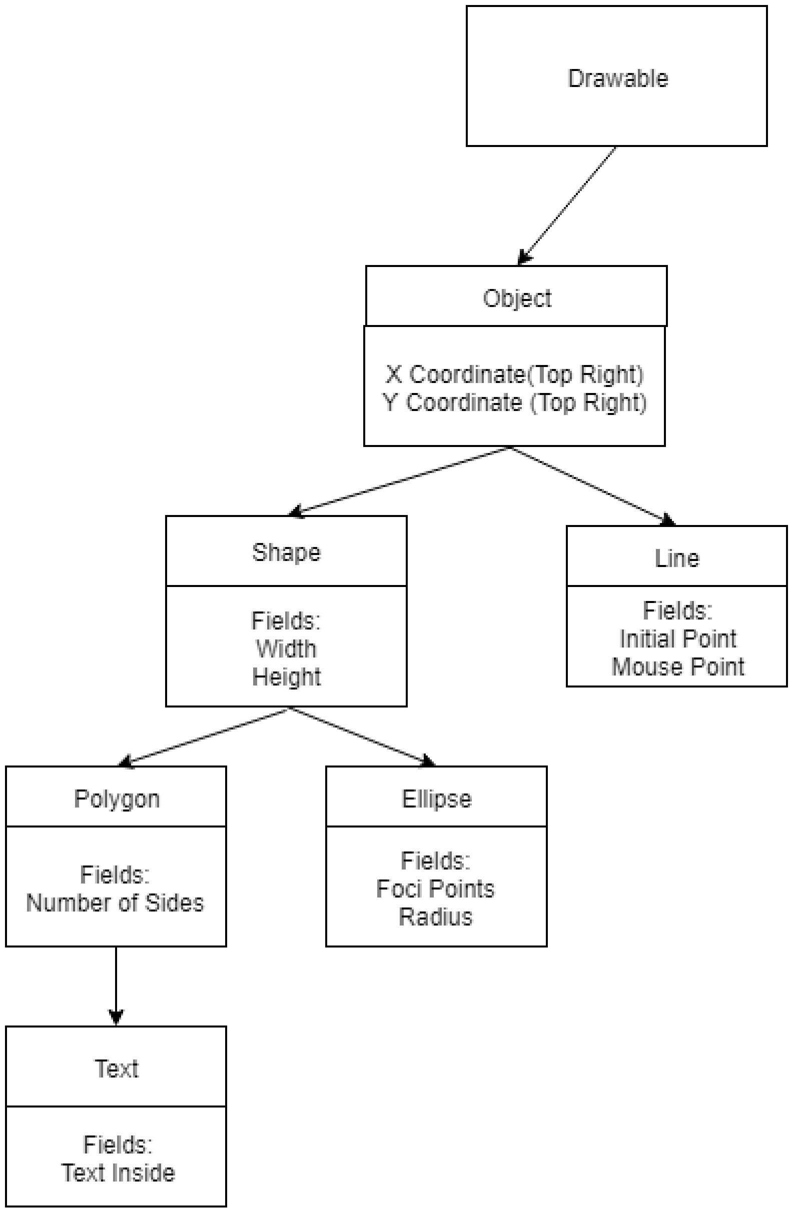


6.1: Some properties these all share, coordinates on the canvas, size, shape (if you view text as an invisible box with text inside), color

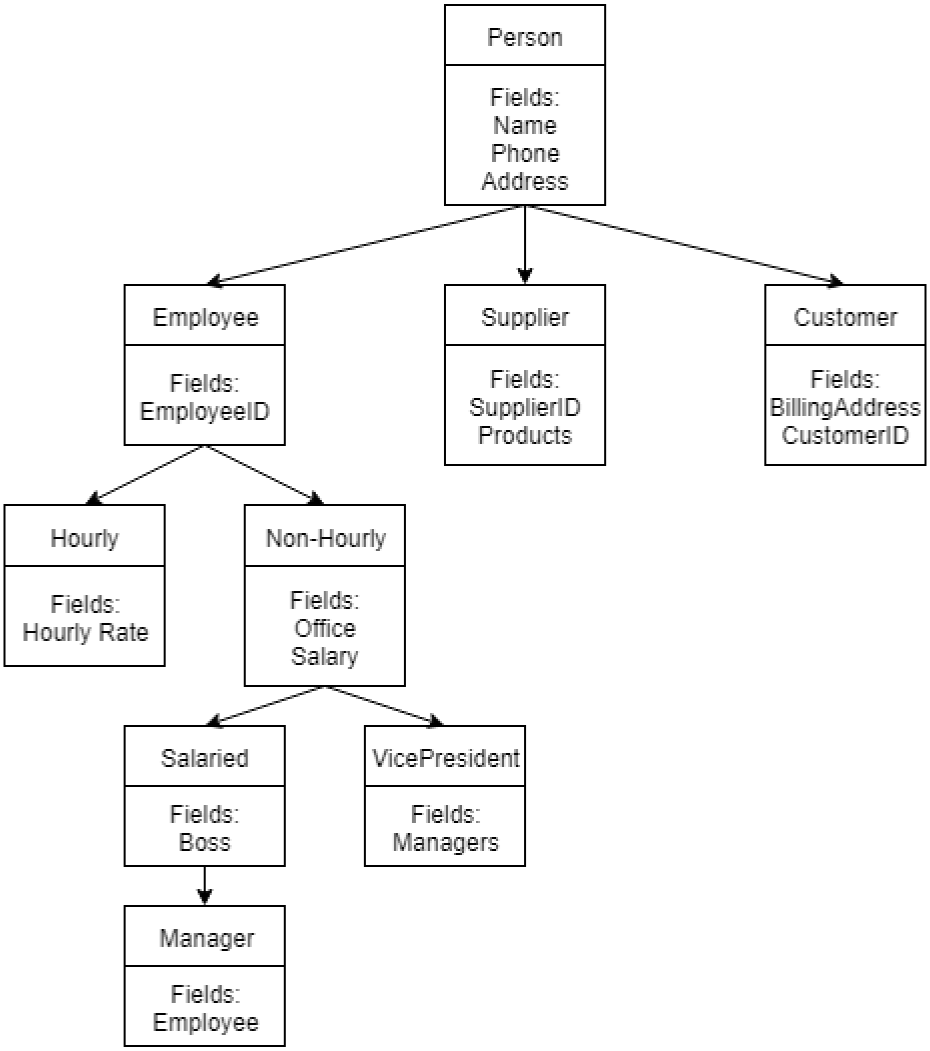
Don't share: number of sides, ellipse has two foci points, line has two points, text has a text field

Some of these properties can be shared in a parent class such as rectangle, star, ellipse, text being in a shape box, each then having it's own subclass, with text being a subclass of rectangle. Line is simply two points and can be its own class

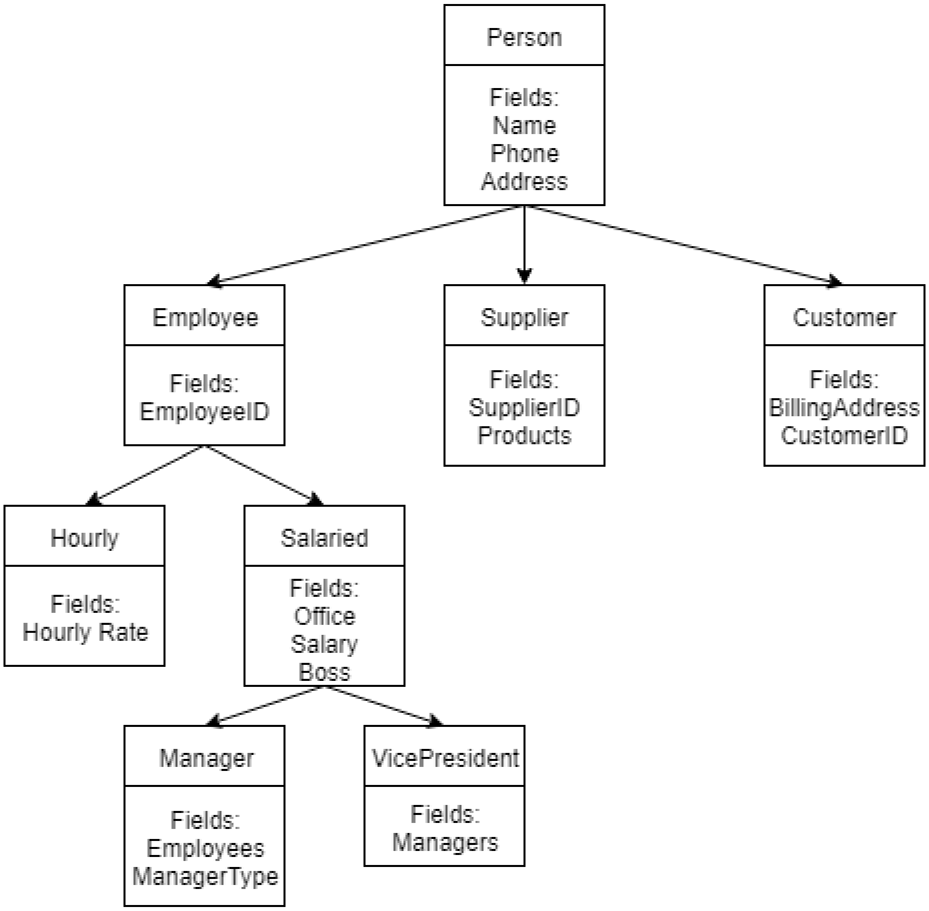
6.2:


Drawable
Object
X Coordinate(Top Right)
Y Coordinate (Top Right)
Shape
Fields:
Width
Height
Line
Fields:
Initial Point
Mouse Point
Polygon
Fields:
Number of Sides
Ellipse
Fields:
Foci Points
Radius
Text
Fields:
Text Inside

6.3:


Person
Fields:
Name
Phone
Address
Employee
Fields:
EmployeeID
Supplier
Fields:
SupplierID
Products
Customer
Fields:
BillingAddress
CustomerID
Hourly
Fields:
Hourly Rate
Non-Hourly
Fields:
Office
Salary
Salaried
Fields:
Boss
VicePresident
Fields:
Managers
Manager
Fields:
Employee

6.6:



Now since VicePresidents also have a boss, they can be grouped under the Salaried category. An additional field must be added to Manager to denote what type of manager they are.