Senior Project Group EVE

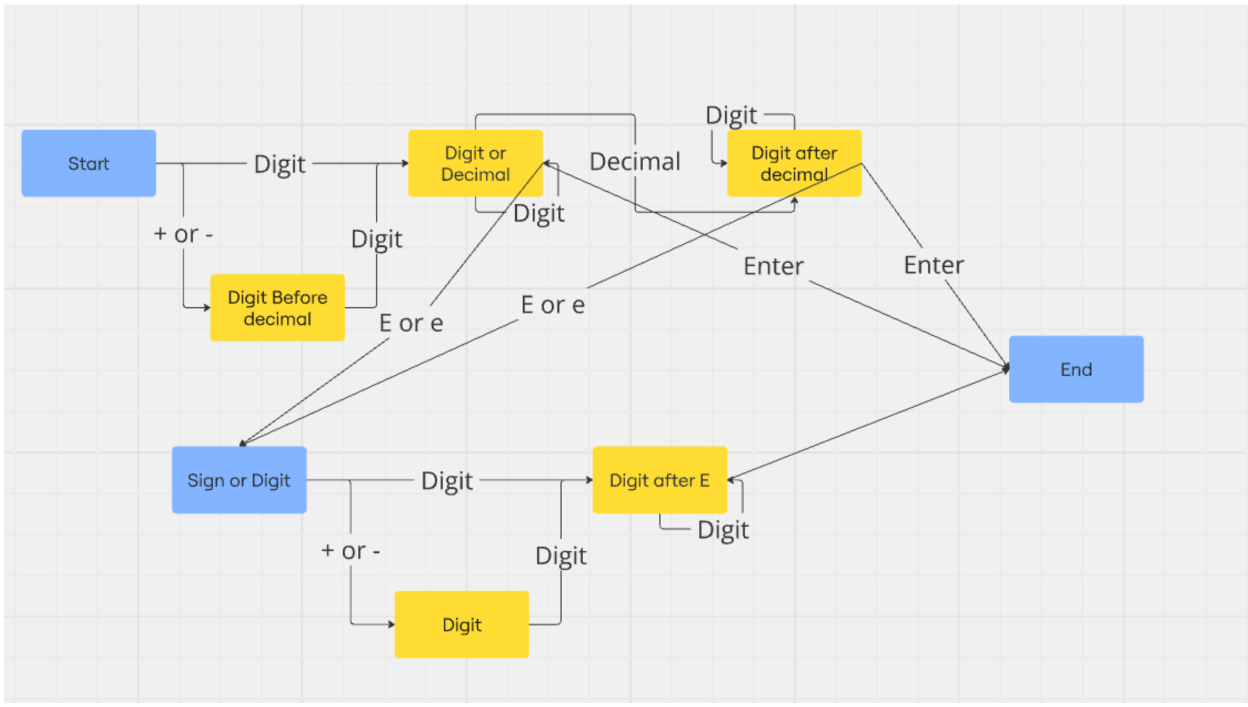
Professor Johnson

2-19-25

Assignment #2

- Problem 5.1:
    - A component-based architecture views parts of the system as loosely connected components that offer services to one another. A service-oriented architecture closely resembles this but implements the components as services, typically operating on different machines that communicate over a network. While the two are alike, the components in a service-oriented architecture are generally more distinct from one another.
- Problem 5.2:
    - This application is straightforward and self-sufficient, requiring no remote services or database. As such, client-server and similar architectures may be excessive. A monolithic architecture is likely suitable due to its simplicity and compact size. Additionally, a data-centric approach works well, as tic-tac-toe allows for the easy creation of move tables and responses, utilizing rule-based methods. The user interface will be event-driven, responding to user actions, and the computer opponent can trigger events when it makes moves, though this may not be necessary. While distributed components could evaluate different move sequences simultaneously, tic-tac-toe is too simple for that. In summary, the application would be easiest to develop as a monolithic, rule-based (data-centric) design.
- Problem 5.4:
    - This situation may appear more complex than the former one, but it's quite manageable. The user interface remains largely unchanged. The only modifications are that the application must share information with another instance of itself over the Internet and that there is no computer opponent. Eliminating the computer opponent indicates that the program no longer requires distributed components. You could utilize web services to enable two programs to interact over the Internet. This would transform the application into a monolithic rule-based (data-centric), service-oriented application.
- Problem 5.6:
    - The ClassyDraw application is capable of saving each drawing in its own individual file, which means it doesn't require a large database. The tools provided by the operating system allow users to manage their files effectively. For instance, these tools enable users to remove outdated files and create backup copies of their documents. The program has the ability to generate a temporary file while a

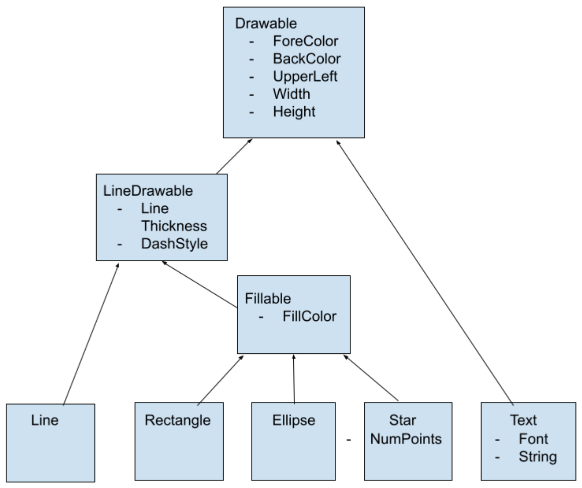
drawing is being edited. If the program crashes or is closed unexpectedly, it can prompt the user to inquire if they would like to restore the temporary file during the next launch.

- Problem 5.8:
  - 

- Problem 6.1:
  - a.) ForeColor, BackColor, UpperLeft, Width, and Height
  - b.) The Text class needs the Font and the String, and the Star class needs to know NumPoints.
  - c.) Certain properties may be common to some classes but not to others. The Rectangle, Ellipse, and Star classes can be filled, which requires a fill color. The classes that are responsible for drawing lines (Line, Rectangle, Ellipse, and Star) also require line attributes (like line thickness and dash style).
  - d.) The shared properties, including foreground color, background color, position, width, and height, should be implemented in a common base class (Drawable) to ensure all drawable objects have these fundamental attributes. Nonshared properties, like font and string for the Text class or the number of points for the Star class, should be implemented directly in their respective subclasses since they are only relevant to those specific shapes. Properties shared by some but not all classes should be handled in intermediate classes. For example, a Fillable class should include the FillColor property and be inherited by Rectangle, Ellipse, and Star, as these shapes can be filled. Similarly, a LineDrawable class should manage properties like LineThickness and DashStyle, which apply to Line, Rectangle, Ellipse, and Star. This structure ensures that common properties are centralized in

the base class while unique and partially shared properties are organized efficiently in the appropriate subclasses.

- Problem 6.2:



    -