Carter Esparza

Assignment #2

5.1.

A component-based architecture organizes a software system into discrete, reusable modules with well-defined interfaces, which can be assembled or replaced easily. In contrast, a service-oriented architecture structures the system around loosely coupled, network-accessible services that communicate through standardized protocols and are designed to interact across diverse platforms and languages.

5.2.

For a mobile tic-tac-toe application that runs entirely on the device and stores high scores locally, a component-based architecture is most appropriate. This approach enables you to break the application into self-contained modules without the overhead of managing network-based services.

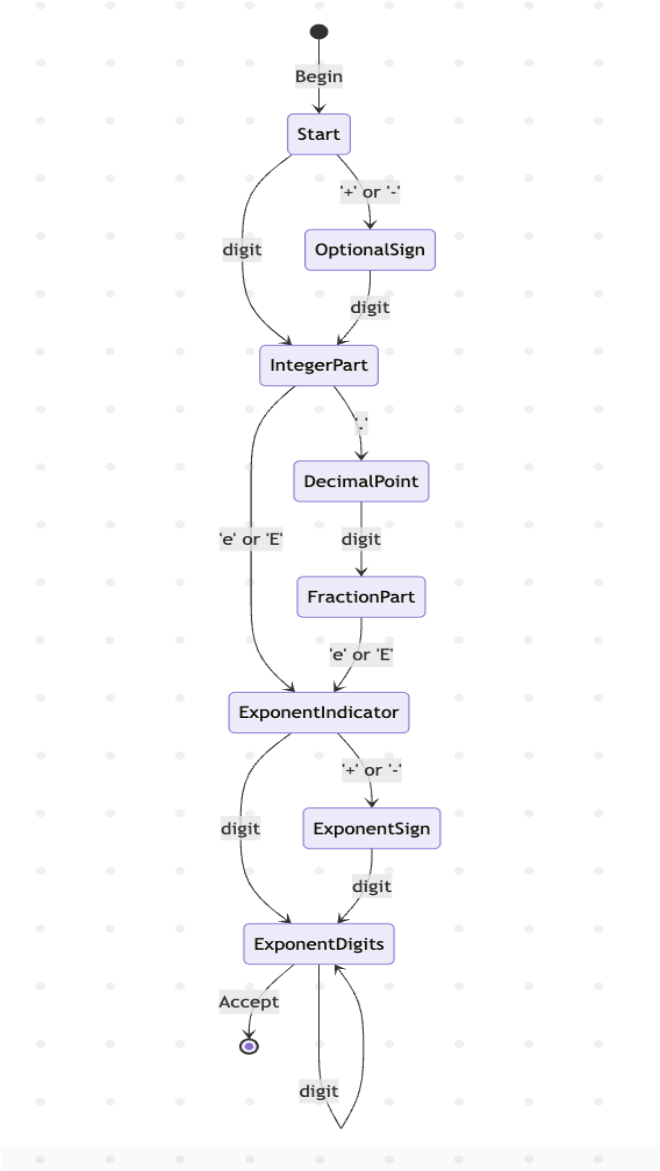
5.4.

When building a chess program that supports two-player gameplay over the Internet, a service-oriented architecture is more suitable. This architecture allows for distributed components that handle networking, session management, and real-time data exchange between two users, thereby supporting remote interactions and the integration of communication services. While internal modules could still be implemented as components, the overall design benefits from the flexible, loosely coupled services typical of SOA.

5.6.

The ClassyDraw application should use a structured database, such as a relational database. This will allow for all files and drawings to be stored with their relevant user / app session data if applicable.

5.8

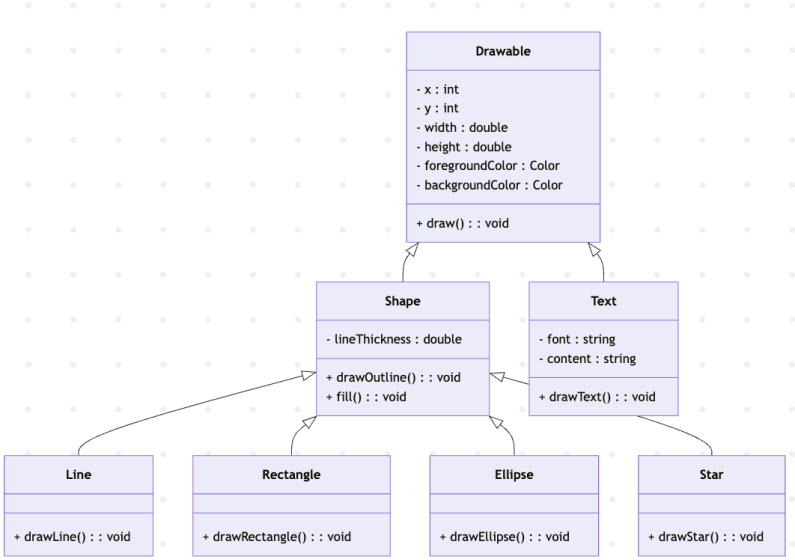


6.1

- Shared Properties:

    - All of these classes share basic drawable properties such as position (coordinates), line color, and line thickness. They typically have methods to render themselves and may inherit from a common base class responsible for graphical representation.
- Non-Shared Properties:
    - Some properties are unique to specific classes. For example, the Text class includes attributes like font style, size, and text content, whereas shape classes such as Rectangle, Ellipse, or Star may have properties like fill color or additional parameters (e.g., number of points for a Star).
- Properties Shared by Some but Not Others:
    - Certain properties such as a fill color or border style might be shared by closed shapes (Rectangle, Ellipse, Star) but not by a Line, which typically does not have a fill.
- Implementation Considerations:
    - Shared properties should be implemented in a common parent class (such as a Drawable class) to avoid redundancy. Properties that apply only to specific types of objects should either be implemented in sub-classes or in intermediate abstract classes (e.g., a Shape class for all fillable objects) to ensure that each class only contains attributes relevant to its functionality.

6.2


Drawable
- x : int
- y : int
- width : double
- height : double
- foregroundColor : Color
- backgroundColor : Color
+ draw() : : void
Shape
- lineThickness : double
+ drawOutline() : : void
+ fill() : : void
Text
- font : string
- content : string
+ drawText() : : void
Line
+ drawLine() : : void
Rectangle
+ drawRectangle() : : void
Ellipse
+ drawEllipse() : : void
Star
+ drawStar() : : void