Problem 5.1, Stephens page 116

A component-based architecture regards a system as a collection of loosely coupled components that provide services for each other. A service-oriented architecture is similar, except that the pieces are implemented as services, often running on separate computers and communicating across a network.

Problem 5.2, Stephens page 116

A monolithic architecture would be most appropriate because it is self-contained and does not require external services of databases. A data-centric approach could be useful for managing high scores locally. More complex architectures like client-server or service-oriented architecture are unnecessary for such a simple application.

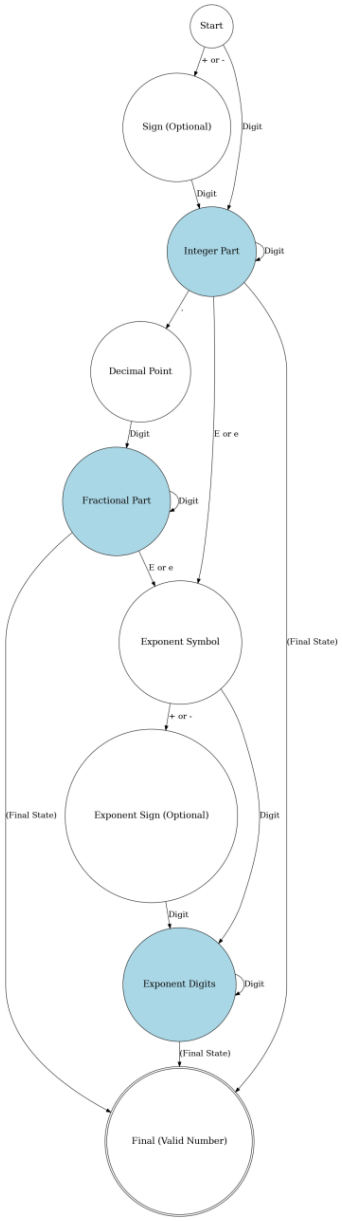
Problem 5.4, Stephens page 116

For a chess game with online multiplayer, a client-server architecture would be most appropriate, as it allows two users to play chess over the internet by handling game state synchronization and player interactions on a central server.

Problem 5.6, Stephens page 116

The ClassyDraw application does not require a complex database structure because it stores each drawing as a separate file rather than in a database. Users can rely on operating system tools to manage files, such as deleting old files and creating backups. For maintenance, the application can generate temporary files while editing, allowing users to recover unsaved changes if the program crashes.

Problem 5.8, Stephens page 116



Problem 6.1, Stephens page 138

Shared Properties :

All ClassyDraw classes, including Line, Rectangle, Ellipse, Star, and Text, share common properties that define their graphical representation. These include position (x, y) to determine their location on the canvas, color for their fill or stroke, stroke width to define the outline thickness, visibility to control whether they are displayed or hidden, and layer order to determine the rendering priority. Additionally, all shapes support transformations such as rotation and scaling.

Properties not shared :

Some properties are unique to specific classes. The Text class has a textContent property that stores the displayed string, along with fontSize and fontFamily. The Star class has a numPoints property to define the number of points, along with a radius. Rectangle, Ellipse, and Star require width and height properties to define their dimensions, whereas the Ellipse class specifically has majorAxis and minorAxis instead of width and height. The Line class is unique in that it does not have width or height, but instead requires startPoint and endPoint coordinates to define its shape.

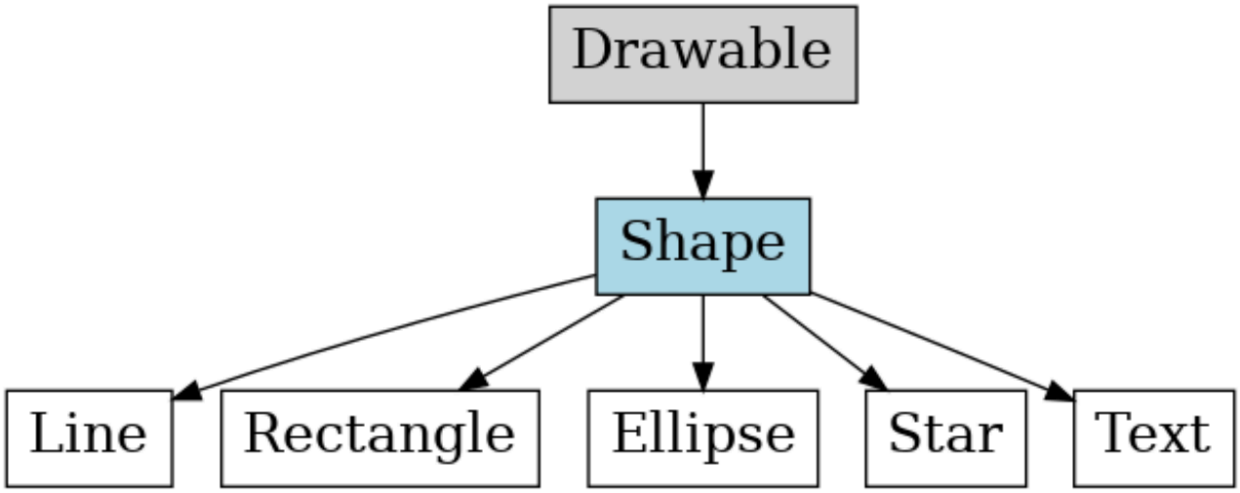
Are there any properties shared by some classes and not others?

Certain properties are shared among only some of the classes. The fillColor property is present in Rectangle, Ellipse, and Star, but it does not apply to Line or Text. The borderColor and strokeStyle properties apply to all classes except Text, which does not have an outline. The concept of a bounding box is relevant to Rectangle, Ellipse, Star, and Text, but it is not directly applicable to Line.

Where should the shared and nonshared properties be implemented?

The shared properties should be implemented in a base class named Shape, which all other shape classes inherit from. This base class should include properties like position, color, stroke width, visibility, layer order, and transformations. Unique properties should be implemented in their respective subclasses. For instance, the Text class should contain textContent, fontSize, and fontFamily, while the Line class should have startPoint and endPoint. Properties that are shared among a subset of classes, such as fillColor for Rectangle, Ellipse, and Star, should be implemented at an appropriate intermediate level of inheritance or within each relevant subclass.

Problem 6.2, Stephens page 138


Drawable
Shape
Line
Rectangle
Ellipse
Star
Text