Aryaman Ramchandran

Jerahmeel Mendoza

Homework 02

5.1 What's the difference between a component-based architecture and a service-oriented architecture?

Component-based architecture involves creating different parts of the application as components that provide services to each other. Each component is a self-contained unit that can be used and reused in various parts of the system. On the other hand, service-oriented architecture (SOA) often involves distributed systems where different services run on different processors, potentially across different computers. SOA emphasizes the provision of services across networked computers, facilitating interoperability and communication between different systems or software applications.

5.2 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?

For a tic-tac-toe phone application that plays against a computer opponent and stores high scores locally, a monolithic architecture would be most suitable. This is because the application is simple and doesn't require complex interactions between different systems or distributed components. A monolithic architecture is straightforward, easier to develop, and maintain for such a small-scale application.

5.4 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.

A chess program allowing two users to play over the Internet would benefit from a client/server architecture. This architecture would enable each player's device to act as a client, while a central server manages the game state, player interactions, and ensures synchronization between the players' moves. The client/server model is well-suited for online multiplayer games due to its ability to handle requests from multiple clients and maintain a consistent game state.

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

For the ClassyDraw application, a relational database system like Access, SQL Server, Oracle, or MySQL would be appropriate for storing drawing data. The high-level design can start with sketching out tables and their relationships, which can be detailed later in the development process. The database should be designed to store and retrieve drawing objects and their properties efficiently.

5.8 Draw a state machine diagram to let a program read floating point numbers in scientific notation as in +37 or -12.3e+17 (which means -12.3 x 1017). Allow both E and e for the exponent symbol. [Jeez, is this like Dr. Dorin's DFAs, or *what*???]

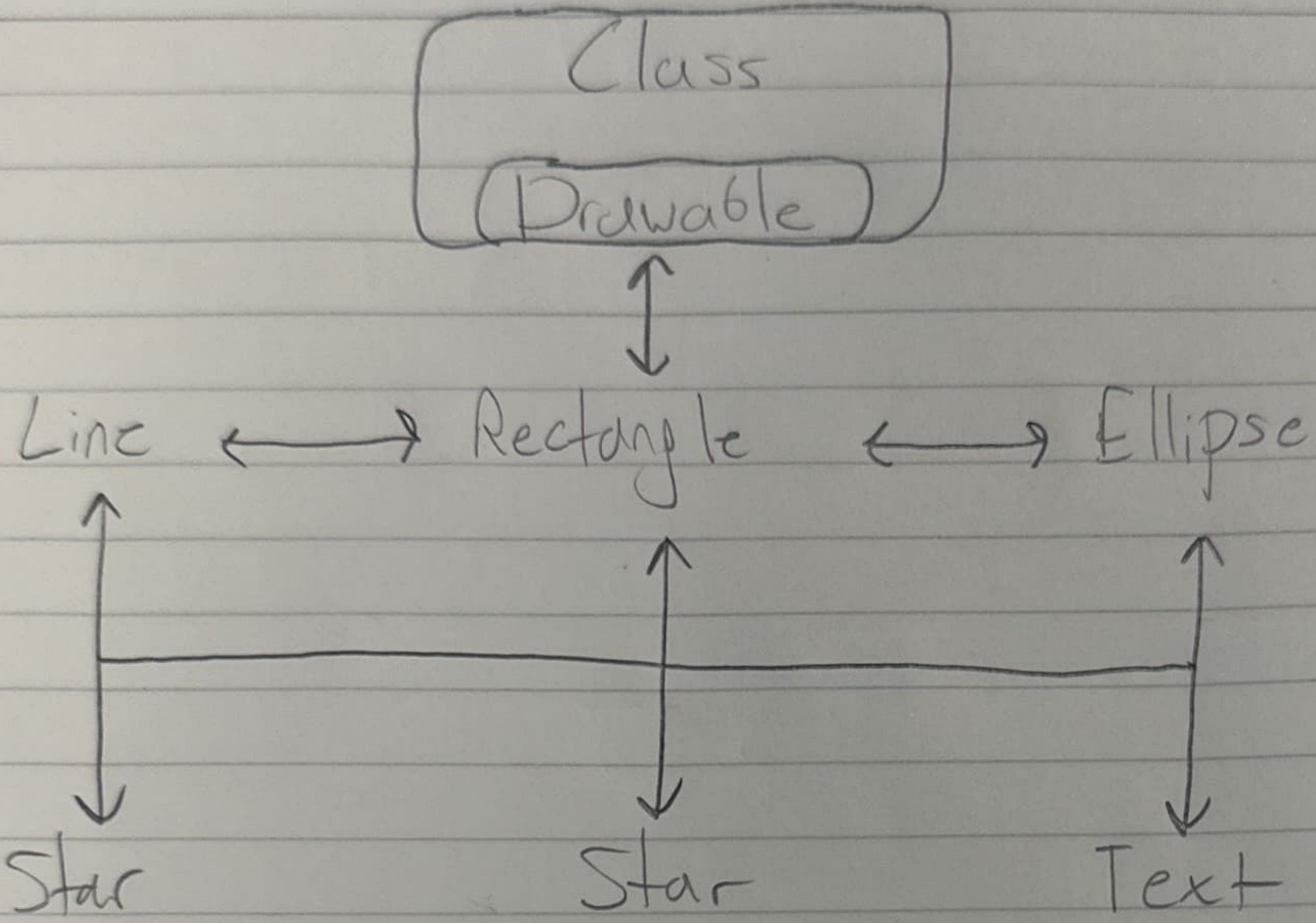
To create a state machine diagram for reading floating-point numbers in scientific notation, you would represent different states an object passes through in response to various events. The diagram should include transitions for handling digits, signs (+/-), the decimal point, and the exponent part. The starting state would involve deciding whether the input is a digit or a sign, and then transitions would occur based on further input like additional digits, a decimal point, and the exponent part. The diagram should clearly indicate the flow from one state to another based on the type of character inputted by the user.

6.1 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?

The shared properties among the ClassyDraw classes like Line, Rectangle, Ellipse, Star, and Text would likely include attributes related to drawing such as coordinates, color, and line thickness. Non-shared properties would be specific to each shape, like the radius for an Ellipse or the number of points for a Star. The shared properties should be implemented in a base class (such as a Drawable class), from which all specific shape classes inherit. Non-shared properties would be defined in their respective subclasses.

6.2 Draw an inheritance diagram showing the properties you identified for Exercise 1. (Create parent classes as needed, and don't forget the Drawable class at the top.)

The inheritance diagram for ClassyDraw classes shows a base class, named Drawable, from which specific classes like Line, Rectangle, Ellipse, Star, and Text inherit. This base class would contain shared properties and methods, while each subclass would have its unique attributes and methods. The diagram would visually represent this inheritance relationship, showing how the specific classes derive from the base Drawable class.


Class
Drawable
Line
Rectangle
Ellipse
Star
Star
Text