Design Manual

Architecture Overview

The application is structured around the main `App` class, which extends `JFrame`. It orchestrates the overall layout and interactions between the `PalettePanel`, `GridPanel`, and `TopMenuBarPanel`.

- `App`: Serves as the main entry point and container for the application. It initializes and arranges the primary panels.
- `MenuBar`: Contains menu items for `Save` and `Open` operations
- `MainPanel`: Handles mouse events and interactions between the PalettePanel and GridPanel
- `PalettePanel`: Displays useable LogicGates and information.
- `PaletteComponent`: Represents a draggable component in the `PalettePanel` with properties such as `logicGate` and `bounds`. It includes a `draw()` method for rendering.
- `GridPanel`: Handles grid components and wires using relative coordinate system
- `GridLogicHandler`: Handles the logic related to the grid, including interactions with logic gates and wires.
- `LogicGate`: An abstract class extended by specific gate types like `ANDGate`. It provides essential methods like `draw()`, `contains()`, and `output()` to handle logic gate functionalities.

Each of these components works together to provide a comprehensive digital logic simulation environment.

Design Patterns

State

The `GridLogicHandler` will potentially implement the State pattern to step through the circuit states, much like a debugger stepping through code.

Component Descriptions

App

At the top of the architectural hierarchy is the App class. This initializes the window with a MainPanel and MenuBar object.

MenuBar

The MenuBar contains commands for saving and loading circuits.

MainPanel

The MainPanel contains the logic for handling mouse operations (may be refactored out to a MouseHandler class). This class manages the interaction between the PalettePanel and GridPanel, such as drag and dropping new logic gate items from the PalettePanel into the GridPanel.

PalettePanel

The PalettePanel displays PaletteComponents which each house a LogicGate component.

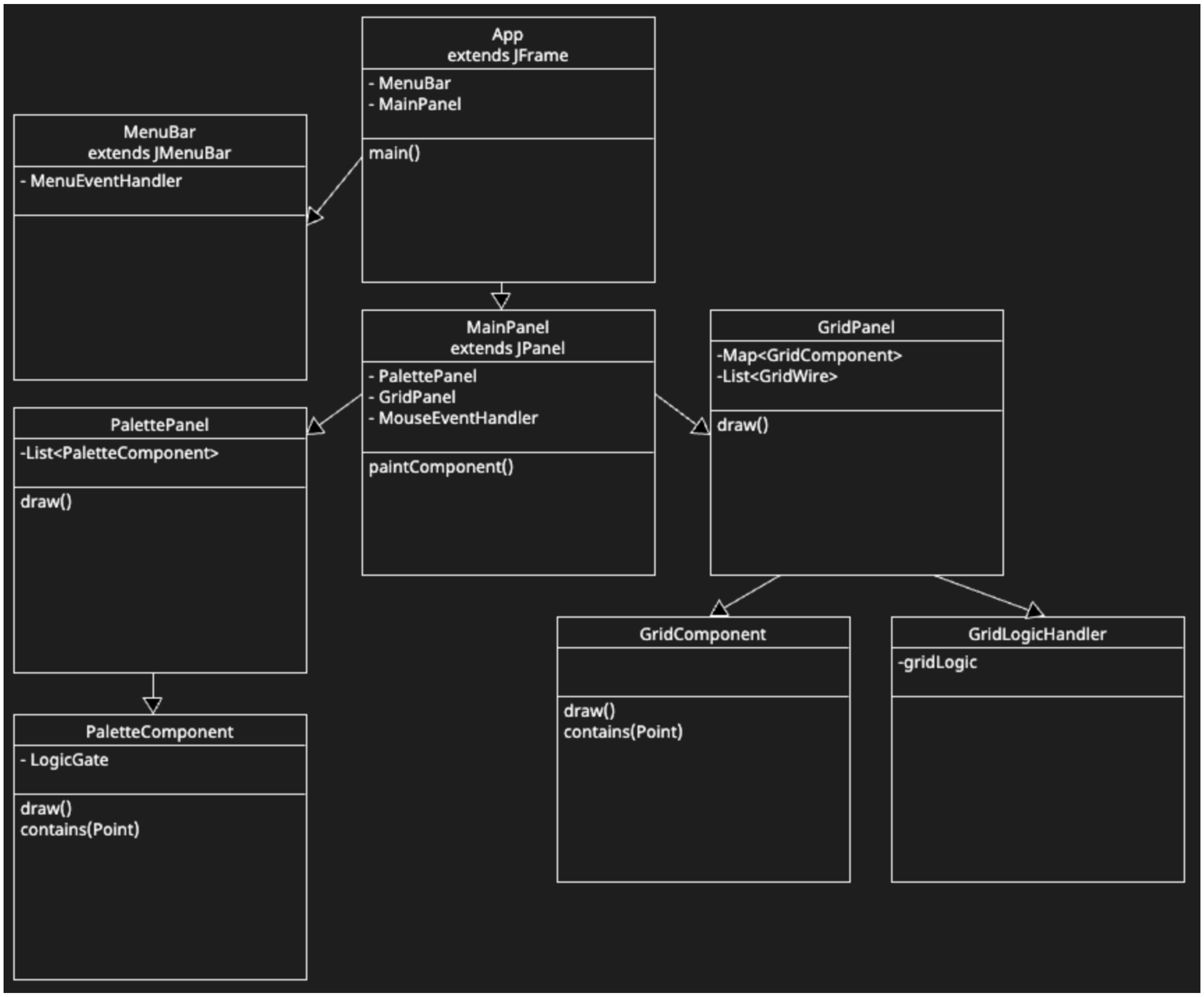
GridPanel

The GridPanel displays a grid and the current view of the logic gates and wires. In future updates, a CircuitLogic class will be added that calculates outcomes based on the circuit on the grid.

LogicGate

Logic Gates are currently implemented as an abstract class which is extended by each gate such as ANDGate.

Diagrams



Standards and Conventions

The project adheres to Java's camelCase naming convention, with classes beginning with uppercase letters, methods as verbs, and objects as nouns.

Project Report

Introduction

The digital logic simulator project aims to create a user-friendly, interactive environment for building and testing digital circuits. The primary goal is to address usability issues encountered in existing simulators, like Logisim, and incorporate their successful features such as a grid layout, panning, and zooming.

Background

Logisim, a key reference for this project, offers valuable lessons in user experience and functionality. However, its wiring system presented significant usability challenges, prompting the development of this simulator. By improving the user interface and streamlining the wiring process, this project seeks to enhance the digital circuit design experience.

Methodology

The project adopted an agile development methodology, with iterative testing and refinement. Java Swing was used for the graphical user interface, ensuring cross-platform compatibility and ease of use. A focus on modularity allowed for flexible addition or alteration of components during the development process.

Implementation Details

The simulator is being implemented in phases, starting with a large-scale skeleton overview of the project. Each item is being added in incremental steps. For example, the grid is first being created statically without zooming and panning, and will be expanded to include these features.

Testing and Evaluation

Testing was conducted informally through debugging sessions.

Results and Discussion

The first week took a lot of trial and error learning about Java Swing, brainstorming the architecture of the project, and working with ChatGPT effectively.
In the second week, much of the project was rewritten to manually handle the transition of drag and drop objects from the PalettePanel into the GridPanel, instead of using the TransferHandler interface and multiple JPanels in a Border style layout.

Conclusion

Now that a larger framework has become clearer in the design process, the next weeks of adding additional features should be easier.

References and Appendices

- Logisim software
- Java Swing documentation
- ChatGPT

User Manual

Setup

Launch the Application by running the App.java file `/src/main/java/logicsim/App.java`.

Functionality

Clicking and dragging a component will allow you to add the Logic Gate to the Grid on the right-hand side. Each logic gate will be able to be rotated, as well as have its inputs or outputs negated. Each component will snap to the grid when dragging, and a visual shows the current position of the dragged object.
Objects placed on the grid are able to be moved with another drag and drop operation.

Future Functionality

Modifying Components

Rotation and editing the `not` state of each of the inputs and outputs for the object will be added.

Wiring

Click and drag from the input-end or output-end of any component to connect wires between components.