Project Report

Introduction:

The goal of this project is to create an interactive floor planner in 2D that allows users to interact with the design element pallet inside the window, with functionalities for placing, selecting, and manipulating different design elements. When users are done or trying to get started with a floor plan, they can save or load their floor plan design. At the end of the project, we will be able to have a simple yet powerful interface allowing straightforward manipulation of space planning with a user guide.

Literature Review/Background Study:

We use existing software like Floor Planner as our background study in terms of UI design. We studied their UI design of putting a manipulation toolbar on top while having the component UI on the side.

Methodology:

We will use the Java Swing programming language in Eclipse IDE for coding, and we will use GitHub as our version control software to keep track of our files.

Implementation Detail:

- Phase 1: Set up Java Swing in Eclipse and Github repository
- Phase 2: Use the existing App.java code as a base and visualize the UI window
- Phase 3: Implement the functionality for placing, selecting, and manipulating design elements
- Phase 4: Implement save and load capabilities for floor plan design
- Phase 5: Refine user interaction with design elements
- Phase 6: Final testing, documentation, and preparation of demonstration materials
- Phase 7: Implement a user guide

Coding Challenges and Solution:

Testing and Evaluation:

Result and Discussion:

Conclusion:

References and Appendices:

User Manual

Installation:

Currently, the software can be only accessed or run through Eclipse, and the code downloaded from GitHub

How to Use:

Basic Functionality

- Once the code starts running, the main design window will pop up.
- The window size or the drawing board can be resized by resizing the window by using the default windows resize option, this will not resize the drawing that's already on the drawing board.

Advanced Functionality

- Save & Load Files: Click File on the top left of the window, then click Save or Load
- Software Helper: Top left, click Help.
- Select, move, rotate, and resize the manipulation toolbar will be under the helper bar.
- Components toolbars like walls, windows, doors, and furniture will be on the vertical toolbar on the right-hand side, to find the furniture, click on the furniture tab to expand all the furniture we had in the program.
- If the wall component is selected, hold the mouse button down at your starting point and release it at the endpoint, this will create a straight wall
- If any other components were selected, just click on the panel and it will create the component where you want it to be.
- In order to manipulate the component that is already on the drawing board, the user has to select the component first before manipulation.

Design Manual

Sketch of User Interface:

Drop down menu for functionality

File | Edit | Help

| Select | move | rotate | resize |

Grid

wall

window

door

Different Furnitures
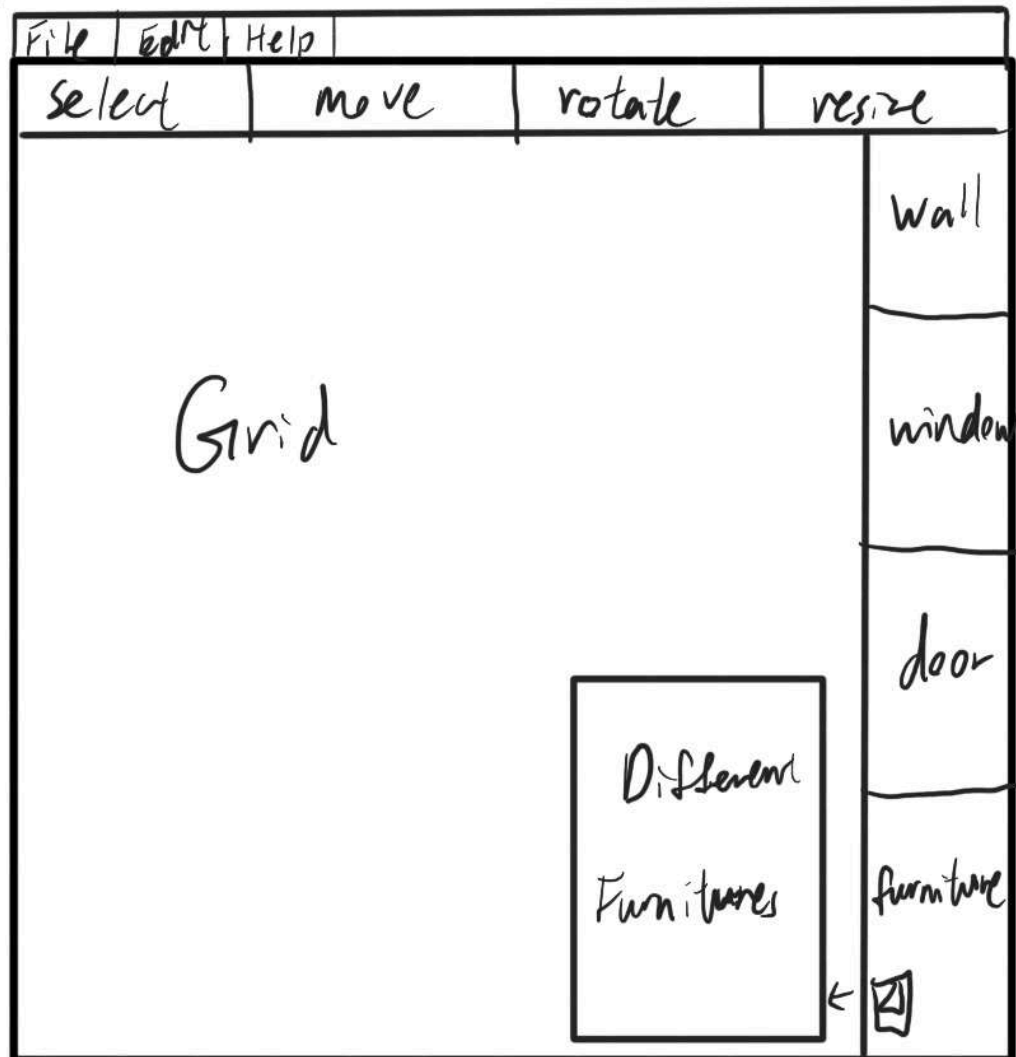
furniture

Architecture Overview:

"As

User Stories:

"As a floor plan designer, I want to add walls, doors, windows, and furniture to my buildings so that I can plan the exterior and interior layout of my space." This suggests we need a feature that can possibly add preset furniture images/sketches to the floor plan.

Design Pattern:

Observer Design Pattern - The DrawingPanel observes the Toolbox. Anytime a button is clicked on the toolbox sidebar, we update the current DesignElement within the DrawingPanel. This DesignElement (wall, window, etc.) will get drawn at the next mouse action until another element is selected from the Toolbox.

Coding Standards and Conventions:

For coding standards, we will have consistent indentation and formatting like braces and parentheses placement. And consistent naming conventions and capitalization across the project. For file structure and classes, we will have one class per tool or function and make sure it's modular.

Component Descriptions:

App Class:
- Initialize the UI
- Set up menu bar
- Creates the file, edit, and help menu with drop-down menu

DrawingPanel Class:
- Implements ElementSlectedObserver
- Draws appropriate elements based on Toolbox selections
- Creates a new drawing panel
- Set all the mouse interactions in the panel by using the mouse listener
- Creates a set of gray grid lines in the drawing panel
- Allows users to draw things in the panel
- Allows resize of the drawing panel
- Allows clearing the panel controlled by the app class.
- Allows save and load files from the app class

DesignElement Interface:
- Abstract class representing a design element.

Door Class:
- Implements DesignElement
- Draws out an arc door

ElementSelectedObserver Class
- An observer class, DrawingPanel is observing the Toolbox

FreeWall Class
- Allows users to draw walls without restriction on the panel

Furniture Class
- Allows users to draw specific furniture on the panel

Wall Class

- Allow users to draw a straight line horizontally or vertically that represents a wall from the start point to the endpoint
- Calculate the starting point and end point of the mouse interaction

Window Class
- Allow users to create a limited-size line that represents the window in the panel

Toolbox Class
- Creates another interactive UI toolbar on the right side of the panel

Appendices

ChatGPT Logs for Wahad Latif:
See Full Pdf/HTML of Chat Logs inside the Git Repository

2/22/2024 - Link: https://chat.openai.com/share/52b1fb64-d3f1-43e7-8fb3-b6eece6600d9

Summary
Certainly! Here's a summary of our conversation:

Initial Development:
- You started developing a floor plan designer application using Java Swing.
- You created a `DesignElement` interface and several classes (`Wall`, `Door`, `Window`, `Furniture`) implementing this interface to represent different design elements.
- You implemented a `DrawingPanel` class extending `JPanel` for drawing on a canvas. It included logic for drawing lines and resizing the canvas.

Refactoring:
- We refactored the code to use a factory method for creating design elements based on the selected tool in the toolbox, avoiding instanceof checks.
- We separated the drawing logic for different elements into separate methods to improve readability and maintainability.
- We added a grid of light gray lines to the canvas to make it look like a grid.

Bugfixing:
- We fixed issues related to drawing walls straight vertically or horizontally, drawing doors as quarter circles, and drawing windows as blue rectangles.

Design Patterns:
- We used the Factory Method pattern to create design elements based on the selected tool in the toolbox.
- We used the Observer pattern to notify the `DrawingPanel` when a new design element is selected in the toolbox.

Final Implementation:
- The application allows users to select design elements (walls, doors, windows, furniture) from the toolbox and place them on the canvas by clicking.
- Walls are drawn straight vertically or horizontally.
- Doors are drawn as quarter circles with one click.
- Windows are drawn as blue rectangles with one click.
- Furniture can be placed on the canvas with one click.

Overall, the application provides a user-friendly interface for designing floor plans and demonstrates the use of various design patterns and best practices in Java Swing development.

2/29/2024 -
Link in Git Repository:
ECS160FloorPlanDesigner\ChatGPTLogs\2_29_2024_LatifWahad_ChatGPTLog.html

Summary:
Certainly! Here's a summary of our conversation:

1. **Initial Development:**
   - You were developing a floor plan designer application using Java Swing.
   - You had a `Toolbox` class for the sidebar with buttons to select different design elements.
   - Each design element was represented by a class implementing the `DesignElement` interface.

2. **Refactoring:**
   - You wanted to add a dropdown menu for the "Furniture" button in the `Toolbox` class.
   - The dropdown menu would contain options for different types of furniture, each represented by its own design element class.

3. **Bugfixing:**
   - You encountered issues with the popup menu opening below the button instead of to the side.
   - We adjusted the code to use the button's location relative to its parent container to position the popup menu correctly.

4. **Use of Design Patterns:**
   - We used the Observer pattern to notify observers (e.g., the `DrawingPanel`) when a design element was selected.
   - The Factory Method pattern was implicitly used when creating different types of furniture objects based on user selection.

5. **Adjustments to the Code:**
   - Added a dropdown menu to the "Furniture" button in the `Toolbox` class.
   - Implemented logic to create different types of furniture objects (e.g., `Chair`, `Table`) based on user selection.
   - Adjusted the popup menu's position to open to the side of the button instead of below it.

6. **Bugfixes and Finalization:**
   - Addressed issues with the popup menu opening outside the application window.
   - Ensured that the popup menu opened correctly relative to the button within the application window.

Throughout the conversation, we focused on improving the user experience and the maintainability of the code by using design patterns and refining the UI behavior based on your requirements.

ChatGPT Logs for Kevin Cao:

See Full Pdf of Chat Logs inside the Git Repository

2/22/2024 - Link: https://chat.openai.com/share/431fce42-0e05-48c2-b8c2-af7cb21309ba

Summary

The initial code drew a quarter circle with a connecting line. Following your request, the code was refactored to draw a "D" shape by using a semicircle and connecting its endpoints with a line. Subsequently, the line was rotated by 90 degrees to achieve the desired visual effect. Throughout the conversation, we iteratively refined the code based on your requirements, focusing on achieving the desired shape and orientation.