ECS160FloorPlanDesigner

By Wahad Latif, Kevin Cao

Github Link:https://github.com/Wahad10/ECS160FloorPlanDesigner
(Github Repo is private currently, will be made public after final submission deadline)

Table of Contents:

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 palette 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, design doc, and video.

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. We also switched to Visual Studio at some point, because Eclipse became a hassle. We regularly committed changes to git, updated this report document, and added ChatGPT Logs to document the process of development.

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 down objects, selecting and manipulating design elements
- Phase 4: Implement save and load capabilities for floor plan design
- Phase 5: Implement select, rotate, resize, and remove functionality
- Phase 6: Add more building blocks for users to use
- Phase 7: Refine user interaction with design elements
- Phase 8: Final testing, documentation, and preparation of demonstration materials
- Phase 9: Implement a user guide

Coding Challenges and Solution: The main issue is when implementing select functionality due to the object detection for different building objects in a floor plan designer.

Testing and Evaluation:

Testing has been done manually after each feature implementation. When we're doing the testing procedure, we keep track of the observer outputs while looking for any unusual behavior after each building block object is added. This includes looking for unusual behavior when drawing the object, to all the toolbar functions we can use on the object. At last, we test the save and load feature by checking if the object in the file still works after save and load.

Result and Discussion:

The implemented code provides a versatile floor plan designer application with various design elements such as furniture, fixtures, and architectural features. Here's a summary of the results and a discussion of key points:

Results:

- Design Elements: The application supports drawing various design elements, including sofas, chairs, tables, sinks, counters, baths, lamps, chimneys, stairs, and storage containers.
- Customization: Each design element can be resized, rotated, and selected, allowing users to customize the floor plan according to their preferences.
- User Interface: The UI includes a drawing panel, toolbox, and function box, providing an intuitive interface for users to interact with the application.
- Functionality: Users can perform actions such as saving, loading, and clearing the canvas, enhancing the usability of the application.

Discussion:

- Versatility: The application's ability to handle various design elements makes it suitable for designing floor plans for different types of spaces, including residential and commercial environments.
- Ease of Use: The intuitive UI design and simple controls make it easy for users to create and modify floor plans without needing advanced technical knowledge.
- Scalability: The modular architecture allows for easy extension with additional design elements or features in the future, making the application scalable and adaptable to evolving requirements.
- Limitations: While the application provides basic drawing and editing capabilities, it may lack advanced features such as precise measurement tools, advanced rendering options, or real-time collaboration.
- Future Improvements: Future updates could focus on adding advanced features, improving user experience, enhancing performance, and addressing any identified limitations or user feedback.

Overall, the implemented floor plan designer application offers a solid foundation for creating and customizing floor plans, catering to the needs of both casual users and professionals in the architecture and interior design fields. With further refinement and feature enhancements, it has the potential to become a valuable tool in the design process.

Conclusion:

In conclusion, the floor plan designer application provides a user-friendly and versatile platform for creating, customizing, and visualizing floor plans. By offering a range of design elements such as furniture, fixtures, and architectural features, the application empowers users to express their creativity and design spaces according to their preferences.

Throughout the development process, key considerations such as usability, functionality, and scalability were prioritized to ensure a seamless user experience and accommodate future expansion. The modular architecture of the application allows for easy integration of additional design elements and features, making it adaptable to evolving requirements and user needs. While the current version of the application offers basic drawing and editing capabilities, there is ample room for future improvements and enhancements. Future updates could focus on adding advanced features, improving performance, and addressing any identified limitations or user feedback to further enhance the utility and effectiveness of the application.

Overall, the floor plan designer application represents a valuable tool for architects, interior designers, and homeowners alike, providing a convenient platform for creating and visualizing floor plans with ease. With its intuitive interface and versatile functionality, the application has the potential to streamline the design process and facilitate the creation of beautiful and functional spaces.

References and Appendices:

https://floorplanner.com/
https://chat.openai.com/

User Manual

Installation:

Currently, the software can only be accessed or run through Eclipse or another IDE (run the *App.java* class under *src/floorplan* after compiling), and the code downloaded from GitHub

Update: There is now an executable jar file in the Git Repository under *ECS160FloorPlanDesigner\target* called *ECS160FloorPlanDesigner-1.0-SNAPSHOT.jar*. Double click this file after downloading the git repo to run the app

How to Use:

Basic Functionality

- Once the code starts running, the main app window will pop up.
- **Resize Window**: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.
- **Close**: Just exit the window or hit Alt+F4
- **Menu:** This contains options to load an existing floor plan, save your current floor plan, clear the canvas.
- **Right Elements Bar:** This contains various buttons for design elements. Just click on the button, then hover over the canvas to see a preview of the drawing. Walls are a special case (you must click and drag to see the preview wall, and releasing mouse will finalize the wall, Walls can only be draw orthogonally, but may be rotated later)
- **Top Function Bar:** This contains various buttons for manipulation functions. Select elements before you start manipulating them. See advanced functionality below

Advanced Functionality

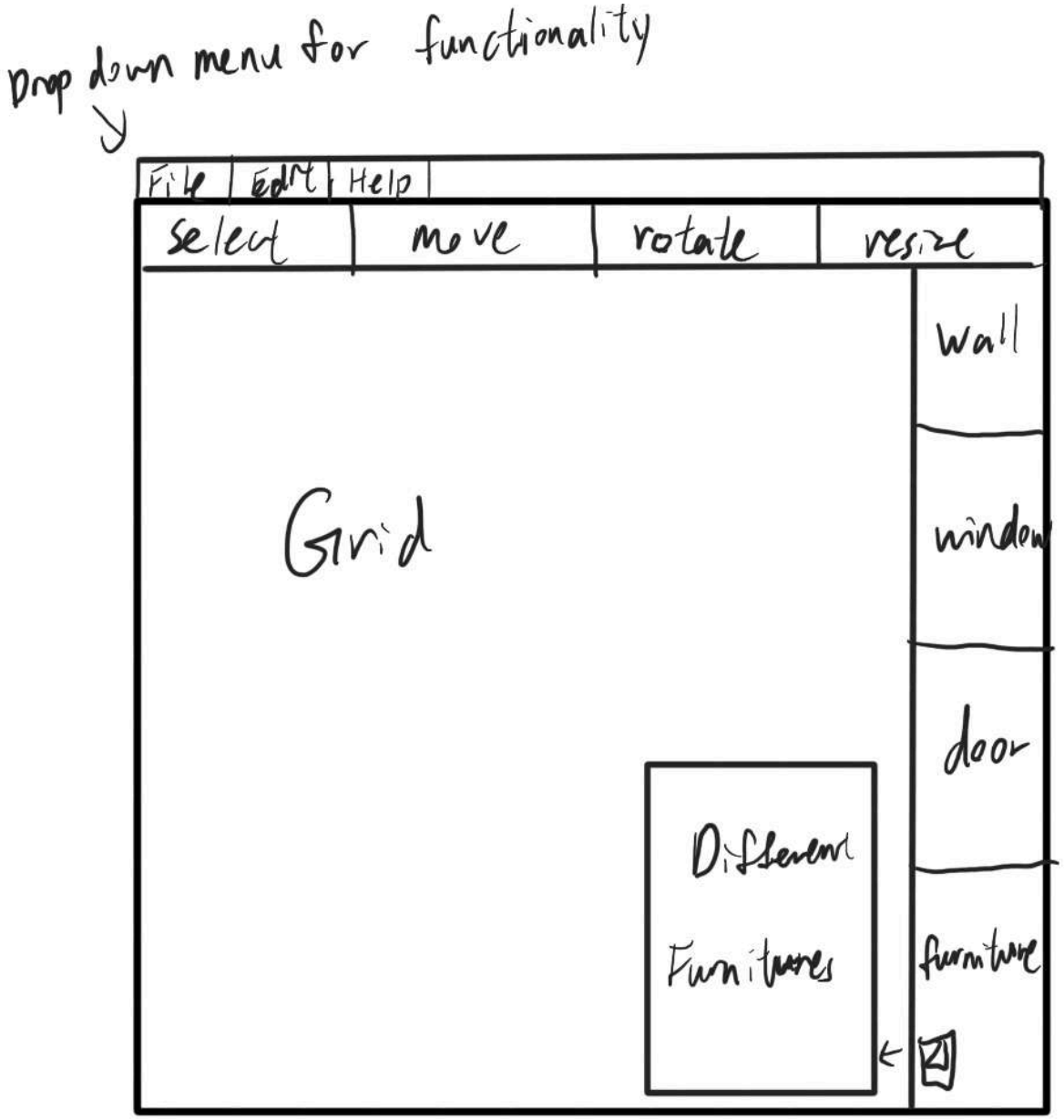
- **Save & Load Files**: Click File on the top left of the window, then click Save or Load
- **Software Helper**: Top left, click Help.
- **Select, move, remove, rotate, and resize** the manipulation toolbar will be under the helper bar.
- **Select :** you must click the select button (or S key), then click and drag over the components you want to select which will turn purple. Then, you can click the other functions on the manipulate toolbar to manipulate the selected element. Once you select, things stay selected until you remove them or hit SPACE, or start placing new design elements. This is on purpose so you can manipulate elements as much as you want once they are selected, and only unselect once you are satisfied. This way you can also make complicated selections, by dragging the mouse multiple times around the

canvas to select multiple objects that may not be right next to each other. If you select something you do not want, you have to hit SPACE to unselect and restart the selection.

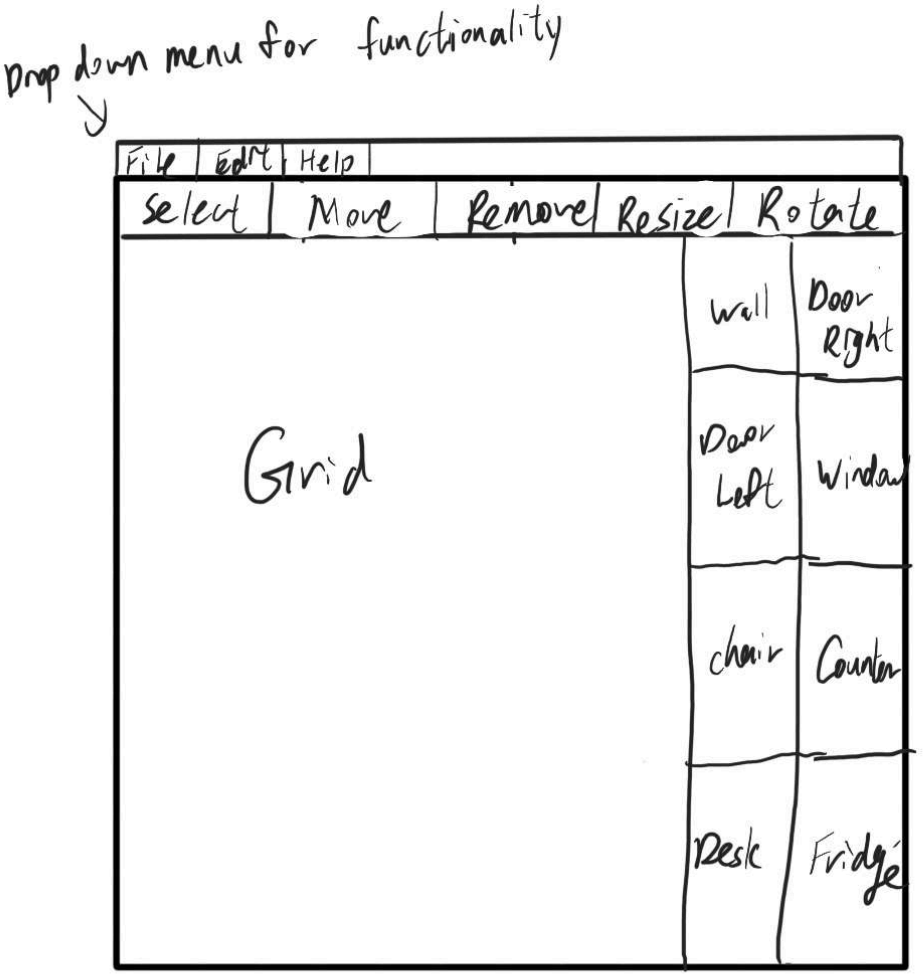
- **Move**: After selection, the user can use the move by dragging their mouse and the components will move accordingly.
- **Remove**: To remove the object, select the object first, once it turns purple, click remove on the toolbar and the object will be removed
- **Rotate and Resize:** will have a helper slider under the toolbar that can be interacted with by pulling from left to right to change the angle for rotation and change from small to large for resize
- Users can select and manipulate multiple objects in the drawing board at once, by holding down the left mouse button with select enabled at an empty space, pulling the mouse and including all the objects in the blue selection box, releasing the mouse button, and then selecting tools in the toolbar.
- Components toolbars like walls, windows, doors, etc will be on the vertical toolbar on the right-hand side.
- If the wall component is selected, press down the left mouse button 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.
- **KeyBoard shortcuts**- S (select), M(move), R(remove), T(rotate), Z(resize), Ctrl+S (save floor plan), Ctrl+O (load floor plan), SPACE (unselect), ESC (stop placing design element), Alt+F4 (exit)

Design Manual

Sketch of User Interface (OLD):



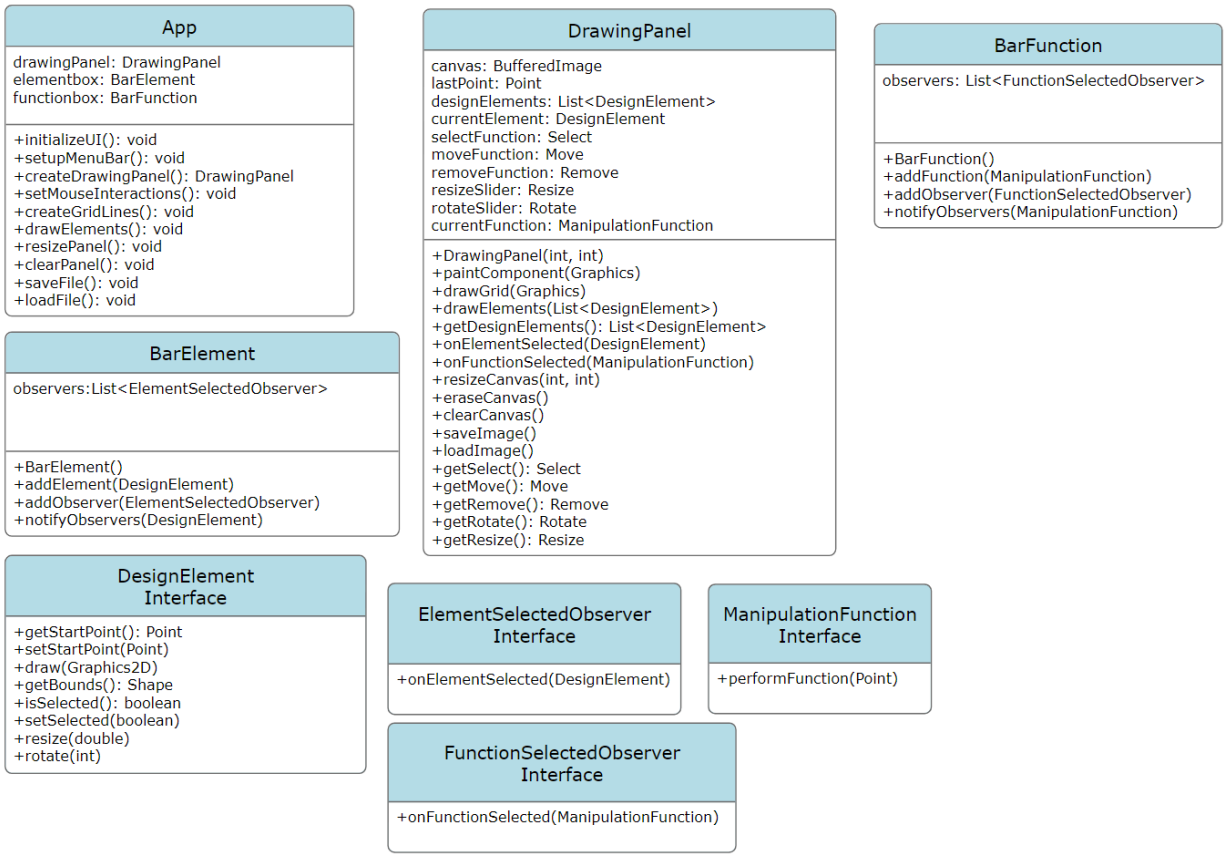
Sketch of User Interface (NEW):



Architecture Overview:

The architecture facilitates the creation and manipulation of design elements within a floor plan designer application. Users can interact with the UI to draw, resize, rotate, save, and load floor plans, following a structured MVC pattern for organization and maintainability.

UML Diagram:



Package Structure:

Floorplan

- **App**: main class, initializes UI including the drawing panel canvas, bar element and function panel, and menu bar
- **DrawingPanel:** canvas for drawing, implements observer interfaces, draws design elements, contains mouse/keyboard handlers to help manipulate design elements
- **ElementSelected Observer Interface**: Drawing Panel implements this so it is notified of what element was selected in BarElement panel
- **FunctionSelected Observer Interface**: Drawing Panel implements this so it is notified of what element was selected in BarFunction panel

Elements

- **BarElement** - JPanel with buttons for each design element, notifies DrawingPanel so the canvas knows what to draw on next mouse click
- **DesignElement** - Interface which all design elements inherit from, Defines methods for draw, bounds, and manipulation
- **Concrete Design Element Classes** - Consists of Wall, DoorRight, DoorLeft, Window, Bath, Bed, Chair, Counter, Desk, Fridge, Lamp, Plant, Sink, Sofa, Stairs, Stove, Table, Toilet Classes

Functions

- **BarFunction** - JPanel with buttons for each manipulation function, notifies DrawingPanel so the canvas knows what to do on next mouse click
- **ManipulationFunction**- Interface which all manipulation functions inherit from, Defines method to perform function using Strategy Pattern
- **Concrete ManipulationFunction Classes** - Consists of Select, Move, Remove, Rotate, Resize, Note that Rotate and Resize are sliders

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 ElementSelectedObserver
- 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

ElementSelectedObserver Interface
- An observer interface so drawingPanel can observe when elements are selected

FunctionSelectedObserver Interface
- An observer interface so drawingPanel can observe when functions are selected

BarFunction Class
- Setting the layout of the toolbar on the top side of the drawing panel
- The bar function includes all the interactable assets, like move, remove, rotate, and resize.

ManipulationFunction Interface
- A public interface for the perform function method

Select Class
- Determine the object the user selected
- Change the selection state of the object

Move Class
- Move the selected object into a location upon user input

Remove Class
- Remove the selected object

Resize Class
- Resize the selected object with an interactable resize bar under the function toolbar

Rotate Class
- Rotate the selected object with an interactable rotate bar under the function toolbar

BarElement Class
- Setting the layout of the toolbar on the right side of the drawing panel
- Has Wall, DoorLeft, DoorRight, Window and all the furniture item: Bath, Chair, Counter, Desk, Fridge, Lamp, Plant, Sink, Sofa, Stairs, Stove, Table, and Toilet

DesignElement Interface
- A public interface that gets used across every building block element that we use
- All building block was implemented based on this interface

- Implemented Wall, DoorLeft, DoorRight, Window
- Implemented Bath, Chair, Counter, Desk, Fridge, Lamp, Plant, Sink, Sofa, Stairs, Stove, Table, and Toilet
- Set and get the starting point and bound of an object
- Saves the selection state of an object
- Functional rotation and resizing for each object
- Sets the object's value in the drawing method

Wall, DoorRight, DoorLeft, Window, Bath, Bed, Chair, Counter, Desk, Fridge, Lamp, Plant, Sink, Sofa, Stairs, Stove, Table, Toilet Classes

- Various design elements that implement design element interface
- Objects that can be placed, drawn and manipulated on the canvas
- Have their own logic for custom drawing, bounds logic, and manipulation

Design Patterns:

Observer Design Pattern - The DrawingPanel observes the BarElement and BarFunction Panels. Anytime a button is clicked on the panels, we update the current DesignElement or Manipulation Function within the DrawingPanel. This DesignElement (wall, window, etc.) or function (select/move) will get drawn/performed at the next mouse action until another element is selected from the Panels.

Strategy Design Pattern - The ManipulationFunction interface has several subclasses (Select, Move, Rotate, Remove, Resize) which implement a method performFunction using the Strategy Pattern.

Iterator Design Pattern - Remove uses Iterator to iterate over design elements and remove the selected ones safely.

User Stories:

As a floor plan designer, I want to efficiently create detailed floor plans with various furniture items, fixtures, and architectural elements. Specifically, I want to:

- Easily add, select, rotate, and resize elements such as beds, sofas, stoves, toilets, plants, flowers, desks, and baths to design accurate floor plans.
- Visualize real-time changes as I edit the floor plan, enabling me to see how the design evolves.
- Save and load floor plans for future editing or sharing with clients and collaborators.
- Clear the canvas to start new floor plans quickly.
- Navigate through an intuitive user interface with tooltips, help menus, and tutorials for guidance.
- Access a wide range of design elements and features to create diverse and creative floor plans for different spaces and purposes.
- Collaborate with clients and team members by saving, loading, and sharing floor plans and working on them together in real-time.
- Ensure that the application is user-friendly, efficient, and reliable for seamless floor plan design experiences.

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.

Appendices

ChatGPT Logs for Wahad Latif:

See Full HTML of Chat Logs inside the Git Repository under folder ChatGPTLogs.

2/22/2024 -

Link in Git Repository:
ECS160FloorPlanDesigner\ChatGPTLogs\2_22_2024_LatifWahad_ChatGPTLog.html

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.

3/12/2024 -

Link in Git Repository:
ECS160FloorPlanDesigner\ChatGPTLogs\3_12_2024_LatifWahad_ChatGPTLog.html

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

1. **Initial Development:** You were working on a floor plan designer application using Java Swing. The application allows users to draw walls, doors, windows, and furniture on a canvas. You implemented various design elements (`Wall`, `Door`, `Window`, `Furniture`) and manipulation functions (`Move`, `Select`, `Remove`, `Resize`).

2. **Refactoring and Bugfixing:** You refactored your code to improve its structure and readability. You also fixed bugs related to drawing elements, selecting them, and saving/loading the canvas.

3. **Use of Design Patterns:** You used the Observer pattern to notify the drawing panel of selected elements and selected functions. You also used the Strategy pattern to implement different manipulation functions (`Move`, `Select`, `Remove`, `Resize`).

4. **Adjustments to the Code:** You made adjustments to the code based on my suggestions. For example, you implemented a `resize` method in the `DesignElement` interface and adjusted the `performFunction` method in the `Resize` class to resize elements correctly.

5. **Final Implementation:** The final implementation allows users to draw walls, doors, windows, and furniture on a grid-based canvas. They can select, move, resize, and remove elements. The application provides a user-friendly interface for designing floor plans.

Overall, you successfully developed a floor plan designer application with various features and implemented several design patterns to improve its design and functionality.

3/20/2024 -

Link in Git Repository:
ECS160FloorPlanDesigner\ChatGPTLogs\3_20_2024_LatifWahad_ChatGPTLog.html

Summary:
Certainly! Here's a summary of our conversation from the beginning of this specific chat up to where you said "Start Log - 3/13/2024":

1. **Move Event Handlers**:
   - You wanted to move the mouse and keyboard event handlers out of the `DrawingPanel` class to clean it up.

2. **Separate Class for Event Handling**:
   - We discussed creating a separate class, such as `MouseHandler`, to handle mouse events and updating the necessary variables.

3. **Custom Adapter for Event Handling**:
   - You wanted to implement a custom adapter class to handle only the required event methods (`mousePressed`, `mouseReleased`, etc.).

4. **Update Method for Event Handling**:
   - You wanted to implement an `update` method in the `MouseHandler` class to update the necessary variables in response to events.

5. **Observer Pattern**:
   - We discussed using the Observer pattern to notify the `DrawingPanel` class of changes in the `MouseHandler` class.

6. **Adjustments to Code**:
   - You adjusted the code to use the custom adapter class for mouse event handling and implemented the Observer pattern for element and function selection notifications.

Overall, the focus of this conversation was on refactoring the code to separate event handling logic from the `DrawingPanel` class and implementing the necessary changes to ensure the `DrawingPanel` class remained up to date with the latest variables and events from the `MouseHandler` class.

ChatGPT Logs for Kevin Cao:

See Full HTML of Chat Logs inside the Git Repository under folder ChatGPTLogs.

2/22/2024 -

Link in Git Repository:
ECS160FloorPlanDesigner\ChatGPTLogs\2_22_2024_CaoKevin_ChatGPTLog.html

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.

3/20/2024 -

Link in Git Repository:
ECS160FloorPlanDesigner\ChatGPTLogs\3_20_2024_CaoKevin_ChatGPTLog.html

Summary:
Throughout this conversation, we discussed the development of a floor plan designer application. The application allows users to create, customize, and visualize floor plans by providing various design elements such as furniture, fixtures, and architectural features.
The conversation began with requests to draw specific design elements such as a bed, sofa, stove, toilet, plant, flower, desk, bath, chimney, and storage container, among others. These requests involved refining the drawings to meet specific requirements, such as adjusting sizes, adding detail, and rotating elements.
As the conversation progressed, new design elements were introduced, including a round table, sink, counter, lamp, and storage. Each element was implemented as a class that implemented the DesignElement interface, allowing for consistent behavior and easy integration into the application.
Key architectural principles were discussed, such as modular design, encapsulation, and scalability, to ensure that the application could accommodate future enhancements and additions. The conversation also touched on usability considerations, user interface design, and potential areas for future improvement.
In conclusion, the conversation provided insights into the development process of the floor plan designer application, highlighting the importance of user feedback, iterative design, and continuous improvement to create a valuable and effective tool for designing floor plans.