Design Manual

Architecture Overview:

The architecture of the Interactive Floor Plan Designer can be broken down into two different categories: the front end, which implements the GUI and elements which users will interact with; and the back end, which implements the business logic, features, and tools required by the front end. The front end was designed in such a manner as to achieve the desired functionality with a clean interface, and without incorporating unnecessary complications or forcing any unwanted aesthetic elements upon the user. As such, the GUI was designed to be rather simple and unobtrusive- getting out of the way of the user and allowing them to most efficiently create their floor plan.

The GUI is a series of JPanels placed within a BufferedImage, along with a JmenuBar. The BufferedImage is used to establish a window for the application, and is thus the canvas upon which all of the other components are drawn (indeed, this BufferedImage is named `canvas`). On top of the BufferedImage, three JPanels are placed using a BorderLayout, two of which are visible when the application launches. On the left/west side of the layout is the panel containing the “Add Item” button. Clicking on this button displays the JPopupMenu with each of the furniture categories. Selecting one of these categories then makes the JPanel on the right/east side of the BorderLayout visible. This panel displays buttons for users to be able to select from the furniture options within the category selected from the popup menu.

Design Patterns:

One of the Design Patterns used repeatedly in the Interactive Floor Plan Designer is the Composite Design Pattern. The Composite Pattern was made use of in both the front end and back end, and was found to be an essential and valuable tool in the design of the application. On the front end, things like JPanels, JButtons, and different layouts are used as individual components, as well as in components consisting of multiple of these different objects. For example, the left/west panel of the GUI is a button within its own panel, and the panel is a component within the BorderLayout of the main BufferedImage window. Similarly on the back end, the drawings for objects like the Bed are a composite of many different individual component objects. The different rectangles for the outline of the bed, each pillow, and the sheets are each their own component, but the Bed object combines them into a single composite object, containing each of the individual components within itself.

Component Descriptions:

Diagrams:

Standards and Conventions:

Project Report

Introduction:

The primary objective of this project is to design and implement an Interactive Floor Plan Designer using Java Swing, a versatile GUI toolkit for Java applications. The purpose of this software tool is to empower users to create detailed floor plans for various types of structures, ranging from residential homes to commercial office spaces. The project aims to provide a user-friendly interface reminiscent of 2D architectural CAD software, facilitating intuitive interaction and efficient design creation.

The scope of the project encompasses the development of a top-down view interface, allowing users to visualize floor plans from an aerial perspective. Key elements such as doors, windows, walls, and furniture are represented using familiar symbols and icons to ensure easy comprehension and manipulation. Additionally, the inclusion of specialized furniture items such as refrigerators, sinks, and beds caters to the specific needs of residential floor planning.

Significance-wise, this project holds importance in bridging the gap between novice developers and complex GUI applications. By leveraging the assistance of ChatGPT, an AI language model, developers can overcome knowledge gaps and accelerate productivity. While the Interactive Floor Plan Designer may not aspire to become an industry standard tool, its creation demonstrates the potential of AI-driven development in simplifying complex tasks and democratizing software engineering.

Methodology:

The development methodology employed in this project revolves around leveraging Java Swing for GUI development and ChatGPT for knowledge augmentation. Java Swing's comprehensive set of components and robust event-driven programming model make it well-suited for creating interactive and visually appealing user interfaces. Additionally, the choice of IntelliJ IDEA as the IDE enhances developer productivity with its advanced features and seamless integration with other development tools. In particular, IntelliJ’s tools for finding usages, inserting imports and unimplemented methods, and displaying parameter names in method calls were invaluable in the development of the Interactive Floor Plan Designer.

The iterative development process involves submitting prompts to ChatGPT, analyzing the generated outputs, and iteratively refining the codebase. While ChatGPT serves as a valuable resource for generating initial solutions and providing insights, manual intervention and refactoring are crucial for maintaining code quality and adhering to best practices.

Implementation Details:

The implementation of the Interactive Floor Plan Designer commenced with the setup of the GUI layout, encompassing elements such as buttons, selection panels, menus, and the central drawing canvas. These components were meticulously crafted to ensure a cohesive and intuitive user experience, with emphasis placed on ease of navigation and functionality.

A primary goal of the GUI design was simplicity and allowing the user to focus on their design, rather than becoming distracted by all of the menu and tool options on-screen. The architecture implementation supported this design goal by putting the furniture categories in a popup menu activated by a button, rather than leaving them constantly in view. One of the key

architectural/design decisions made in the process of developing the application was whether the button and furniture panel should be part of the same composite panel, or belong to different panels. It would be very intuitive to the user to have a single panel where they could access their menus, tools, and objects, and thus an early implementation considered putting the button above the furniture panel in a BoxLayout. However, it quickly became apparent that this architectural decision, though excellent on application launch, breached the design requirements during use. The usage of a popup menu had already been finalized, and having the button placed above the furniture panel resulted in the menu dropping over the furniture panel, obscuring the user's view. This was deemed an unacceptable breach of the simple design ethos, and thus two separate panels were employed to handle the button/popup menu and furniture panel separately.

The back end was notably more complicated to build. One of the particularly challenging aspects of the architecture was implementing different functionalities for different furniture objects, while still relying on the base FurnitureObject interface to provide a uniform interface for the fundamental features and methods required by all of the furniture objects. To address this problem, one of the main goals of the development team was the separation of concerns.

Some of the furniture objects needed to have the ability to be rotated, resized, or moved but one did not imply the others. Additionally, for some of the objects, it wouldn't sense for any of these functionalities to be implemented. For example, when selecting a region of the design to resize the objects within it, the user should expect all of the objects within that area to be selected. However, the user would probably be surprised if a wall within this region was selected and also became bigger. Furthermore, there was no point in implementing rotation for a circular lamp, but it was decided to give users the ability to create lamps of various sizes in their floor plan designs.

The ability to implement multiple interfaces in Java was the backbone of the architectural implementation of separation of concerns. Interfaces were created to allow objects to implement different methods for resizing, rotation, and movement. In the aforementioned example of trying to resize objects but leave the walls alone, the resizing method can be applied to all selected instances of the Resizable interface.

/// selection implementation

Without a doubt, the most frustrating and difficult part of the project was trying to get ChatGPT to do more than the simplest of tasks. Many, many hours of work were wasted trying to craft prompts which ChatGPT might actually respond to by providing even moderately advanced functionality. Almost without fail, asking ChatGPT for more than a single method implementation resulted in it providing nothing but function signatures without any implementation. After spending far too much time carefully and explicitly describing the desired functionality, layout, and requirements of the entire project, ChatGPT gave me a nearly empty code section, full of unimplemented methods- just enough code to open a blank Swing window and close the program when the window was closed.

And code wasn't the only area in which ChatGPT fell far short of expectations. After providing details for it to use to write a section of the Project Report, ChatGPT failed to include a single one of the specific details provided, and ended up outputting less text than what was entered into the prompt.

User Guide: Interactive Floor Plan Creator

Welcome to the Interactive Floor Plan Creator! This user guide will walk you through the various features and functionalities of the application to help you create your own floor plans efficiently.

Table of Contents

1. Getting Started

Upon launching the application, you will be presented with a user-friendly interface consisting of a grid canvas in the center and a sidebar on the left. The canvas is where you will create your floor plan, while the sidebar provides options to add furniture objects.

2. Adding Furniture Objects

To add furniture objects to your floor plan, follow these steps:

- Click on the "Add Item" button located on the left sidebar.
- A menu will appear with different categories of furniture objects such as "Essentials," "Kitchen," "Bathroom," "Bedroom," and "Living Room/Office."
- Select a category to view the available furniture options within that category.

3. Placing Objects on the Canvas

Once you have selected a furniture object from the sidebar, you can place it on the canvas by following these steps:

- Click on the box corresponding to the desired furniture object in the right sidebar.
- Next, click again on the canvas at the location where you want to place the object.
- The selected furniture object will be placed under the cursor on the canvas.

4. Modifying Objects

You can modify placed objects on the canvas using the following options:

- **Moving Objects:** Right-click on the object you want to move to select it. This will open a menu where you can choose the "Move" option. After selecting "Move," click on the canvas to specify the new location for the object.
- **Resizing Objects:** Right-click on the object and choose the "Resize" option from the menu to resize the object as desired.
- **Rotating Objects:** Similarly, right-click on the object and choose the "Rotate" option from the menu to rotate the object.

- **Deleting Objects:** To delete an object, right-click on it and select the "Delete" option from the menu.

5. Saving and Loading Drawings

You can save your floor plan drawings to a file and load them later using the file menu. Here's how:

- Click on the "File" menu located at the top of the application window.
- From the dropdown menu, select "Save" to save your current drawing to a file. You will be prompted to choose a location and filename for the saved file.
- To load a previously saved drawing, select "Load" from the "File" menu. Choose the file you want to load, and your drawing will be loaded into the canvas.

Congratulations! You are now ready to create your own floor plans using the Interactive Floor Plan Creator. Enjoy designing!