Fix TopMenuBar Clear Issue

March 21, 2024

Anonymous

```
package com.ECS160.UI;

import javax.swing.*;
import java.awt.event.ActionListener;

public class TopMenuBar extends JMenuBar {

    private ActionListener gridViewToggleListener;
    private DrawingPanel drawingPanel;

    public TopMenuBar(DrawingPanel drawingPanel) {
        this.drawingPanel = drawingPanel;

        // File Menu
        JMenu fileMenu = new JMenu("File");
        JMenuItem saveItem = new JMenuItem("Save");
        JMenuItem loadItem = new JMenuItem("Load");
        JMenuItem clearItem = new JMenuItem("Clear");
        JMenuItem exitItem = new JMenuItem("Exit");

        saveItem.addActionListener(e -> saveAction());
        loadItem.addActionListener(e -> System.out.println("Load action"));
        exitItem.addActionListener(e -> exitAction());
        clearItem.addActionListener(e -> clearAction()); // Set action for clearItem

        fileMenu.add(saveItem);
        fileMenu.add(loadItem);
        fileMenu.addSeparator();
        fileMenu.add(clearItem);
        fileMenu.addSeparator();
        fileMenu.add(exitItem);

        add(fileMenu);

        // Edit Menu
        JMenu editMenu = new JMenu("Edit");
        JMenuItem undoItem = new JMenuItem("Undo");
        JMenuItem redoItem = new JMenuItem("Redo");

        undoItem.addActionListener(e -> undoAction());
        redoItem.addActionListener(e -> redoAction());

        editMenu.add(undoItem);
```

```
        editMenu.add(redoItem);

        add(editMenu);

        // View Menu
        JMenu viewMenu = new JMenu("View");
        JMenuItem gridViewItem = new JMenuItem("Grid View");

        gridViewItem.addActionListener(e -> drawingPanel.toggleGridView()); // Directly call the
DrawingPanel's method

        viewMenu.add(gridViewItem);

        add(viewMenu);

        // Help Menu
        JMenu helpMenu = new JMenu("Help");
        JMenuItem aboutItem = new JMenuItem("About");

        aboutItem.addActionListener(e -> System.out.println("About action"));

        helpMenu.add(aboutItem);

        add(helpMenu);
    }

    // Setter method to set the DrawingPanel instance
    public void setDrawingPanel(DrawingPanel drawingPanel) {
        this.drawingPanel = drawingPanel;
    }

    // Method to trigger grid view toggle action
    public void toggleGridView() {
        if (drawingPanel != null) {
            drawingPanel.toggleGridView();
        }
    }


    // Method to clear the drawing panel
    private void clearAction() {
        if (drawingPanel != null) {
            drawingPanel.clear();
        }
    }

    private void saveAction() {
        System.out.println("Save action");
    }

    private void exitAction() {
```

```
        System.out.println("Exit action");
        System.exit(0);
    }

    private void undoAction() {
        System.out.println("Undo action");
    }

    private void redoAction() {
        System.out.println("Redo action");
    }
}
```

Why does clear not work.

Here isthe main java file for the app:

```
package com.ECS160.Apps;

import com.ECS160.UI.*;

import javax.swing.*;
import java.awt.*;

public class FloorDesignApp extends JFrame {

    private DrawingPanel drawingPanel;
    private TopMenuBar menuBar;
    private FurnitureManager furnitureManager; // Add FurnitureManager

    public FloorDesignApp() {
        super("Interactive Floor Design");
        initUI();
    }

    private void initUI() {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (UnsupportedLookAndFeelException | ClassNotFoundException |
InstantiationException | IllegalAccessException e) {
            e.printStackTrace();
        }

        // Initialize the FurnitureManager and load furniture images
        furnitureManager = new FurnitureManager();

        furnitureManager.loadFurnitureImages("src/main/java/com/ECS160/Icons"); // Provide the
correct path

        PageSwapper pageSwapper = new PageSwapper(furnitureManager);
        // Main panel that will hold both the sidebar and the tabbed pane
```

```
        JPanel mainPanel = new JPanel(new BorderLayout());

        // Add the tabbed pane to the main panel
        mainPanel.add(pageSwapper, BorderLayout.CENTER);


        // Optionally, you can add the button to the top or bottom of the mainPanel or another panel,
as preferred
        mainPanel.add(pageSwapper.getControlPanel(), BorderLayout.NORTH);

        // Center panel containing canvas and sidebar
        JPanel centerPanel = new JPanel(new BorderLayout());


        // Create a scroll pane to contain the drawing panel
        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setPreferredSize(new Dimension(1100, 800)); // Set preferred size for the visible area

        // Create the drawing panel
        drawingPanel = new DrawingPanel(false);
        drawingPanel.setPreferredSize(new Dimension(1000, 1000)); // Set size of the drawing canvas
        scrollPane.setViewportView(drawingPanel); // Set the drawing panel as the viewport of the scroll
pane

        centerPanel.add(scrollPane, BorderLayout.CENTER);

        // Add the sidebar with furniture

        Sidebar sidebar = new Sidebar(furnitureManager); // Pass the furnitureManager to Sidebar
        mainPanel.add(sidebar, BorderLayout.WEST);

        // Menu bar
        // Assuming you want a single menu bar for the application window and not individual ones for
each tab
        menuBar = new TopMenuBar(null); // Modify TopMenuBar to handle a possibly null
DrawingPanel or update logic accordingly
        setJMenuBar(menuBar);

        setContentPane(mainPanel); // Set the main panel as the content pane of the frame
        pack();

        // Set a minimum size for the frame and other properties
        setMinimumSize(new Dimension(800, 600));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }



    public static void main(String[] args) {
```

```
        SwingUtilities.invokeLater(() -> {
            FloorDesignApp app = new FloorDesignApp();
            app.setVisible(true);
        });
    }
}
```

ChatGPT

The issue with the "Clear" functionality not working as expected in your application likely stems from the way the `DrawingPanel` is being managed and utilized across your `TopMenuBar` and `FloorDesignApp` classes. Here's a detailed breakdown of the potential problem and how you can resolve it:

1. **TopMenuBar Initialization with Null DrawingPanel**: In your `FloorDesignApp` class, you're initializing `TopMenuBar` with `null` instead of passing the actual `drawingPanel` instance. This means when you invoke the `clearAction()` in `TopMenuBar`, it checks if `drawingPanel` is not null (which it is, because you've passed `null`) and does nothing, hence no action is performed.

   To resolve this, you should pass the actual `drawingPanel` reference when you initialize `TopMenuBar`. Change the relevant line in `FloorDesignApp` from:

   ```java
   menuBar = new TopMenuBar(null);
   ```

   to:

   ```java
   menuBar = new TopMenuBar(drawingPanel);
   ```

2. **Setting DrawingPanel Instance**: The method `setDrawingPanel(DrawingPanel drawingPanel)` in `TopMenuBar` is currently never used. If there are scenarios where the `drawingPanel` might change or be set later, you should ensure this method is called appropriately. However, given your current code structure, it seems you can just pass the `drawingPanel` at the initialization as mentioned above and avoid potential complexities.

3. **Ensure DrawingPanel's clear() Method Works**: You haven't shown the implementation of `DrawingPanel.clear()`. Ensure that this method correctly clears the panel's content. It should ideally reset the state or content of the drawing panel to a blank state.

4. **Synchronize Across Components**: Ensure that any action on the drawing panel (like clear) is properly reflected across all components that might depend on its state. For instance, if there's any component showing properties or details of the selected objects in the `DrawingPanel`, it should be updated/refreshed when the panel is cleared.

After making the above modification, your clear functionality should work as intended. If it still doesn't work, you would need to debug further by checking:

- If the `clearAction` is being called when you expect it to be (you can add more print statements to verify this).
- If the `clear()` method in `DrawingPanel` effectively clears its content (ensure the graphical content is being reset).

Anonymous

menuBar = new TopMenuBar(null);

what should I change this to

ChatGPT

You should change the line where you instantiate `TopMenuBar` with `null` to pass the actual instance of `drawingPanel` that you've created. You should do this after you've initialized `drawingPanel`. Here's how you can modify the code in your `FloorDesignApp` class:

First, ensure that you create `drawingPanel` before you create `menuBar`. Then, pass `drawingPanel` to the constructor of `TopMenuBar` like this:

java

```java
// Create the drawing panel first
drawingPanel = new DrawingPanel(false);
drawingPanel.setPreferredSize(new Dimension(1000, 1000)); // Set size of the drawing canvas

// Now pass this drawingPanel instance to TopMenuBar
menuBar = new TopMenuBar(drawingPanel);
```

This way, `TopMenuBar` will have a reference to the actual `drawingPanel` instance and will be able to call methods like `clear()` on it. Make sure you adjust the order of instantiation in `initUI()` in your `FloorDesignApp` class to reflect this change. Here's the adjusted order to make it clear:

java

```java
private void initUI() {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (UnsupportedLookAndFeelException | ClassNotFoundException |
InstantiationException | IllegalAccessException e) {
        e.printStackTrace();
    }

    furnitureManager = new FurnitureManager();
    furnitureManager.loadFurnitureImages("src/main/java/com/ECS160/Icons");

    // Initialize drawingPanel before menuBar
    drawingPanel = new DrawingPanel(false);
    drawingPanel.setPreferredSize(new Dimension(1000, 1000));

    // Now create menuBar and pass drawingPanel to it
    menuBar = new TopMenuBar(drawingPanel);

    // The rest of your UI initialization continues here...
    ...
}
```

After this change, `TopMenuBar` should be able to interact with `drawingPanel` correctly, including clearing it when needed.

Anonymous

```
private void initUI() {
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (UnsupportedLookAndFeelException | ClassNotFoundException |
InstantiationException | IllegalAccessException e) {
        e.printStackTrace();
    }

    // Initialize the FurnitureManager and load furniture images
    furnitureManager = new FurnitureManager();

    furnitureManager.loadFurnitureImages("src/main/java/com/ECS160/Icons"); // Provide the
correct path

    PageSwapper pageSwapper = new PageSwapper(furnitureManager);
    // Main panel that will hold both the sidebar and the tabbed pane
    JPanel mainPanel = new JPanel(new BorderLayout());

    // Add the tabbed pane to the main panel
    mainPanel.add(pageSwapper, BorderLayout.CENTER);


    // Optionally, you can add the button to the top or bottom of the mainPanel or another panel,
as preferred
    mainPanel.add(pageSwapper.getControlPanel(), BorderLayout.NORTH);

    // Center panel containing canvas and sidebar
    JPanel centerPanel = new JPanel(new BorderLayout());


    // Create a scroll pane to contain the drawing panel
    JScrollPane scrollPane = new JScrollPane();
    scrollPane.setPreferredSize(new Dimension(1100, 800)); // Set preferred size for the visible area

    // Create the drawing panel
    drawingPanel = new DrawingPanel(false);
    drawingPanel.setPreferredSize(new Dimension(1000, 1000)); // Set size of the drawing canvas
    scrollPane.setViewportView(drawingPanel); // Set the drawing panel as the viewport of the scroll
pane

    centerPanel.add(scrollPane, BorderLayout.CENTER);

    // Add the sidebar with furniture

    Sidebar sidebar = new Sidebar(furnitureManager); // Pass the furnitureManager to Sidebar
```

```
        mainPanel.add(sidebar, BorderLayout.WEST);

        // Menu bar
        // Assuming you want a single menu bar for the application window and not individual ones for each tab
        menuBar = new TopMenuBar(null); // Modify TopMenuBar to handle a possibly null DrawingPanel or update logic accordingly
        setJMenuBar(menuBar);

        setContentPane(mainPanel); // Set the main panel as the content pane of the frame
        pack();

        // Set a minimum size for the frame and other properties
        setMinimumSize(new Dimension(800, 600));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }
```

This is my full code, my drawing canvas is in PageSwapper

ChatGPT

If the `drawingPanel` that needs interaction with the `TopMenuBar` is actually contained within your `PageSwapper` and not directly accessible from `FloorDesignApp`, you'll need a way to reference that specific `drawingPanel` from the `TopMenuBar`. Since the direct access is not evident in the provided code snippets, you'll need to implement a method or mechanism to retrieve or interact with the `drawingPanel` within `PageSwapper`.

Here's an adjusted approach considering your setup:

1. **Access DrawingPanel through PageSwapper**: First, you need to ensure that there's a way to access or control the `drawingPanel` from `PageSwapper`. If `PageSwapper` manages the `drawingPanel`, you should provide a method within `PageSwapper` to expose the required `drawingPanel` actions (like clear, undo, redo, etc.) that `TopMenuBar` needs to trigger.

2. **Modify TopMenuBar**: Update `TopMenuBar` to accept `PageSwapper` instead of `DrawingPanel`. Then, use the provided methods of `PageSwapper` to perform the necessary actions.

3. **Update FloorDesignApp**: Adjust the instantiation of `TopMenuBar` to pass `pageSwapper` instead of `null` or `drawingPanel`.

Here's how you might start these adjustments:

In PageSwapper (Assuming you have control methods or can add them):

```java
public void clearDrawingPanel() {
    // Assuming DrawingPanel is managed here and has a clear() method.
    this.drawingPanel.clear();
}
```

```
// Similarly, you might want to add methods for undo and redo etc.
```

In TopMenuBar:
Change the constructor to accept a `PageSwapper`:

java

```java
private PageSwapper pageSwapper;

public TopMenuBar(PageSwapper pageSwapper) {
    this.pageSwapper = pageSwapper;
    // rest of the initialization...
}
```

Then adjust `clearAction()` method to call `pageSwapper.clearDrawingPanel()`.

In FloorDesignApp:

java

```java
menuBar = new TopMenuBar(pageSwapper);
```

These changes establish a proper communication path from `TopMenuBar` through `PageSwapper` to the `DrawingPanel`. Ensure all action handlers in `TopMenuBar` are adapted to interact with `pageSwapper`'s exposed methods for controlling the `drawingPanel`.

Anonymous

public PageSwapper(FurnitureManager furnitureManager) {
this.furnitureManager = furnitureManager;
this.initialPageCount = 3;
this.pageCount = initialPageCount; // Initialize page count
setLayout(new BorderLayout());
tabbedPane = new JTabbedPane();
initializeTabs();
}

private void initializeTabs() {
for (int i = 1; i <= initialPageCount; i++) {
JPanel pagePanel = createPage(i); // Create a panel for each tab/page
tabbedPane.addTab("Page " + i, pagePanel);
}
add(tabbedPane, BorderLayout.CENTER);
}

This is some code from a java class I made called PageSwapper,

I also added a function to this class called
public JPanel getControlPanel() {
JPanel controlPanel = new JPanel(new BorderLayout());

```
JButton addPageButton = new JButton("Add Page");
addPageButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        addPage();
    }
});
controlPanel.add(addPageButton, BorderLayout.EAST);

JButton renamePageButton = new JButton("Rename Current Page");
renamePageButton.addActionListener(new ActionListener() {
```

This is some of its code. Right now, the main app file gets the control panel from the function and adds it to the panel. However, if possible, I want to add the control panel to PageSwapper directly above the tabbed panels. I want one control panel just to be clear. Can i do this?

ChatGPT

Yes, you can add the control panel directly to the `PageSwapper` class, above the tabbed panels, ensuring there's only one control panel shared across all pages. You would typically do this by modifying the `PageSwapper` layout to include this control panel at the top (North) and the tabbed pane in the center.

Here's how you can modify your `PageSwapper` class to include the control panel directly within it:

1. **Modify `PageSwapper` Constructor**: Update the constructor to call a method that initializes the control panel and adds it to the `PageSwapper`.
2. **Initialize Control Panel**: Create a method within `PageSwapper` that initializes the control panel and places it above the tabbed panels.

Here is how you can adjust your `PageSwapper` class to include these changes:

java

```java
public class PageSwapper extends JPanel {
    private FurnitureManager furnitureManager;
    private int initialPageCount;
    private int pageCount;
    private JTabbedPane tabbedPane;

    public PageSwapper(FurnitureManager furnitureManager) {
        this.furnitureManager = furnitureManager;
        this.initialPageCount = 3;
        this.pageCount = initialPageCount;
        setLayout(new BorderLayout());

        // Initialize and add the control panel
        JPanel controlPanel = getControlPanel();
        add(controlPanel, BorderLayout.NORTH);

        // Initialize and add the tabbed pane
        tabbedPane = new JTabbedPane();
        initializeTabs();
    }
```

```
    private void initializeTabs() {
        for (int i = 1; i <= initialPageCount; i++) {
            JPanel pagePanel = createPage(i);
            tabbedPane.addTab("Page " + i, pagePanel);
        }
        add(tabbedPane, BorderLayout.CENTER);
    }

    public JPanel getControlPanel() {
        JPanel controlPanel = new JPanel(new BorderLayout());
        JButton addPageButton = new JButton("Add Page");
        addPageButton.addActionListener(e -> addPage());
        controlPanel.add(addPageButton, BorderLayout.EAST);

        JButton renamePageButton = new JButton("Rename Current Page");
        // Add your action listener for renaming here...

        controlPanel.add(renamePageButton, BorderLayout.WEST);
        return controlPanel;
    }

    // Other methods like addPage, createPage, etc.
}
```

With these changes, the control panel will be directly integrated into your `PageSwapper` and positioned above the tabbed panels. Now, you don't need to manually add the control panel to the `PageSwapper` from `FloorDesignApp`; it's already included within `PageSwapper` itself.

Anonymous

private void initializeTabs() {
    for (int i = 1; i <= initialPageCount; i++) {
      JPanel pagePanel = createPage(i); // Create a panel for each tab/page
      tabbedPane.addTab("Page " + i, pagePanel);
    }
    add(tabbedPane, BorderLayout.CENTER);
  }

this is how i initialize each page, how do i make the clear function clear the current drawing panel?

ChatGPT

To implement a clear function that clears the current drawing panel within the `PageSwapper`, you first need to ensure that each `pagePanel` (i.e., each tab) maintains a reference to its associated `DrawingPanel`. I'll assume your `createPage` method creates a `pagePanel` that includes a `DrawingPanel`. If that's the case, you would need a way to access the `DrawingPanel` of the currently selected tab.

Here's how you could extend your `PageSwapper` class to include a method that clears the drawing on the currently selected tab:

1. **Modify `createPage` Method**: Ensure this method either returns a `DrawingPanel` or a `JPanel` that contains a `DrawingPanel`, and you have a way to retrieve the `DrawingPanel` later.

2. **Implement a Clear Method**: This method should retrieve the current tab, identify the `DrawingPanel` within it, and invoke its clear method.

Here is an adjusted outline of the approach:

java

```java
public class PageSwapper extends JPanel {
    private JTabbedPane tabbedPane;
    // Other attributes...

    // Assume createPage method now returns a JPanel containing a DrawingPanel
    private JPanel createPage(int pageNumber) {
        JPanel pagePanel = new JPanel(new BorderLayout());
        DrawingPanel drawingPanel = new DrawingPanel(false);
        // Set up your DrawingPanel and add it to pagePanel
        pagePanel.add(drawingPanel, BorderLayout.CENTER);
        return pagePanel;
    }

    private void initializeTabs() {
        for (int i = 1; i <= initialPageCount; i++) {
            JPanel pagePanel = createPage(i);
            tabbedPane.addTab("Page " + i, pagePanel);
        }
        add(tabbedPane, BorderLayout.CENTER);
    }

    // Clear the current DrawingPanel
    public void clearCurrentDrawingPanel() {
        int selectedIndex = tabbedPane.getSelectedIndex();
        if (selectedIndex != -1) { // If there's at least one tab selected
            JPanel currentPanel = (JPanel) tabbedPane.getComponentAt(selectedIndex);
            // Assuming the DrawingPanel is the sole or primary component:
            Component[] components = currentPanel.getComponents();
            for (Component comp : components) {
                if (comp instanceof DrawingPanel) {
                    ((DrawingPanel) comp).clear();
                    break;
                }
            }
        }
    }
}
```

This `clearCurrentDrawingPanel` method finds the currently selected tab, iterates through its components to find the `DrawingPanel`, and calls `clear()` on it. Make sure the actual structure of your `pagePanel` corresponds to this logic—particularly, this assumes the `DrawingPanel` is a direct child of the `pagePanel`. If there's a different containment hierarchy, you'll need to adjust the code to navigate it correctly.

Anonymous

private JPanel createPage(int pageNumber) {
    JPanel mainPanel = new JPanel(new BorderLayout());

    // Example setup for a drawing panel in each tab
    JScrollPane scrollPane = new JScrollPane();
    scrollPane.setPreferredSize(new Dimension(600, 400)); // Set preferred size

    DrawingPanel drawingPanel = new DrawingPanel(false);

```
    drawingPanel.setPreferredSize(new Dimension(1000,
            1000));
    scrollPane.setViewportView(drawingPanel);

    mainPanel.add(scrollPane, BorderLayout.CENTER);

    // You can add more components specific to each page here

    return mainPanel;
}
```

My create page looks like this, can you rewrite the clearCurrentDrawingPanel class based on this?

ChatGPT

Given your `createPage` implementation, the `DrawingPanel` is nested within a `JScrollPane`, which is, in turn, placed within the `mainPanel`. To clear the current drawing panel, you will need to access this `DrawingPanel` through its parent `JScrollPane`. Here's how you can rewrite the `clearCurrentDrawingPanel` method accordingly:

java

```java
public void clearCurrentDrawingPanel() {
    int selectedIndex = tabbedPane.getSelectedIndex();
    if (selectedIndex != -1) { // Check if there is a selected tab
        // Get the currently selected tab
        JPanel mainPanel = (JPanel) tabbedPane.getComponentAt(selectedIndex);

        // Assuming the JScrollPane is the only or the first component in the mainPanel
        JScrollPane scrollPane = (JScrollPane) mainPanel.getComponent(0);

        // Get the DrawingPanel from the JScrollPane's viewport
        DrawingPanel drawingPanel = (DrawingPanel) scrollPane.getViewport().getView();

        // Clear the drawing panel
        drawingPanel.clear();
    }
}
```

This method finds the currently selected tab, retrieves the `mainPanel`, and assumes that the first component in `mainPanel` is your `JScrollPane`. It then accesses the `DrawingPanel` from the viewport of this `JScrollPane` and calls the `clear()` method on the `DrawingPanel`.

Now, you can call `clearCurrentDrawingPanel()` from your `TopMenuBar` or any other place where you need to trigger the clear action for the currently displayed drawing panel.

Anonymous

```
switch (e.getKeyCode()) {
                case KeyEvent.VK_LEFT:
                    // Rotate counter-clockwise
                    draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() -
rotationIncrement);
                    break;
```

```
                case KeyEvent.VK_RIGHT:
                    // Rotate clockwise
                    draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() +
rotationIncrement);
                    break;
                case KeyEvent.VK_UP:
                    // Scale up
                    draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() +
scaleIncrement);
                    break;
                case KeyEvent.VK_DOWN:
                    // Scale down
                    draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() -
scaleIncrement);
                    break;
                case KeyEvent.
            }
```

Can you conitnue the case im writing in this switch statement please. I want the furniture to rotate 90 degrees to the right when the r key is clicked on the keyboard

ChatGPT

Certainly! To rotate the furniture 90 degrees to the right when the 'R' key is pressed, you should add a case for `KeyEvent.VK_R` and then adjust the rotation angle of the `draggedFurniture` by adding 90 degrees. Here's how you can continue and modify the switch statement:

java

```java
switch (e.getKeyCode()) {
    case KeyEvent.VK_LEFT:
        // Rotate counter-clockwise
        draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() -
rotationIncrement);
        break;
    case KeyEvent.VK_RIGHT:
        // Rotate clockwise
        draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() +
rotationIncrement);
        break;
    case KeyEvent.VK_UP:
        // Scale up
        draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() + scaleIncrement);
        break;
    case KeyEvent.VK_DOWN:
        // Scale down
        draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() - scaleIncrement);
        break;
    case KeyEvent.VK_R:
        // Rotate 90 degrees to the right
        draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() + 90);
        break;
    // Potentially other cases
}
```

This code assumes that the rotation angle is measured in degrees and that increasing the angle corresponds to a clockwise rotation. The rotation operation will rotate the furniture 90 degrees to the right whenever the 'R' key is pressed.

Anonymous

can you make the funriture to rotate to the nearest 90 degree increment (eg: 0,90,180,270)

ChatGPT

To rotate the furniture to the nearest 90-degree increment, you can round the current rotation angle to the nearest multiple of 90. This approach ensures that regardless of the current angle, pressing the key will align the rotation angle to 0, 90, 180, or 270 degrees. Here's how you can modify the case for the 'R' key to achieve this:

java

```java
switch (e.getKeyCode()) {
    case KeyEvent.VK_LEFT:
        // Rotate counter-clockwise
        draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() -
rotationIncrement);
        break;
    case KeyEvent.VK_RIGHT:
        // Rotate clockwise
        draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() +
rotationIncrement);
        break;
    case KeyEvent.VK_UP:
        // Scale up
        draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() + scaleIncrement);
        break;
    case KeyEvent.VK_DOWN:
        // Scale down
        draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() - scaleIncrement);
        break;
    case KeyEvent.VK_R:
        // Rotate to the nearest 90 degree increment
        int currentAngle = draggedFurniture.getRotationAngle();
        int angleRemainder = currentAngle % 90;
        int nearest90Increment;

        if (angleRemainder < 45) {
            // Closer to the lower multiple of 90
            nearest90Increment = currentAngle - angleRemainder;
        } else {
            // Closer to the upper multiple of 90
            nearest90Increment = currentAngle + (90 - angleRemainder);
        }

        draggedFurniture.setRotationAngle(nearest90Increment);
        break;
    // Potentially other cases
}
```

This code calculates the remainder of the current angle divided by 90. If the remainder is less than 45, it means the current angle is closer to the lower multiple of 90, and the furniture should be rotated back. If the remainder is 45 or greater, the angle is closer to the next higher multiple of 90, and the furniture should be rotated forward. The rotation angle of the furniture is then set to this nearest multiple of 90.

Anonymous

wait sorry, i meant could you rotate it 90 degrees but it snaps to the nearest 90 degree increment after the rotate

ChatGPT

Certainly! To rotate the furniture by 90 degrees and ensure it snaps to the nearest 90-degree increment after rotation, you can first add 90 degrees to the current angle, then round the result to the nearest 90-degree increment. This operation ensures a smooth rotation followed by alignment to the standard angle increments (0, 90, 180, 270). Here's the adjusted case for the 'R' key to implement this behavior:

java

```java
switch (e.getKeyCode()) {
    case KeyEvent.VK_LEFT:
        // Rotate counter-clockwise
        draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() -
rotationIncrement);
        break;
    case KeyEvent.VK_RIGHT:
        // Rotate clockwise
        draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() +
rotationIncrement);
        break;
    case KeyEvent.VK_UP:
        // Scale up
        draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() + scaleIncrement);
        break;
    case KeyEvent.VK_DOWN:
        // Scale down
        draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() - scaleIncrement);
        break;
    case KeyEvent.VK_R:
        // Add 90 then snap to the nearest 90 degree increment
        int newAngle = draggedFurniture.getRotationAngle() + 90;
        // Normalize the angle between 0 and 359 degrees
        newAngle %= 360;
        // Snap to the nearest 90 degrees increment
        int angleRemainder = newAngle % 90;
        if (angleRemainder >= 45) {
            newAngle += (90 - angleRemainder); // Snap up to the next increment
        } else {
            newAngle -= angleRemainder; // Snap down to the previous increment
        }

        draggedFurniture.setRotationAngle(newAngle);
        break;
    // Potentially other cases
}
```

This code ensures that every time the 'R' key is pressed, the furniture's rotation angle increases by 90 degrees and then aligns to the nearest 90-degree increment. The angle normalization step ensures that the resulting angle stays within a 0-359 degree range, maintaining consistency and preventing overflow.

Anonymous

i have a pom.xml file, how do i redownload dependencies

ChatGPT

To redownload dependencies in a Maven project (specified in your `pom.xml` file), you can use Maven's dependency-related goals to either update your dependencies or force a re-download. Here's how you can do it:

1. **Clean the local repository**: This command will delete the `target` directory and the resolved dependencies, forcing Maven to redownload them the next time it needs them.

   ```bash
   mvn clean
   ```

2. **Force update of dependencies**: This command will update your project's dependencies, forcing a check for updated releases and snapshots on remote repositories and re-downloading dependencies if necessary.

   ```bash
   mvn -U clean install
   ```

   The `-U` flag stands for "update-snapshots" and forces Maven to check for newer versions of snapshots and releases.

If you specifically want to force Maven to re-download all dependencies without considering the current copies in your local repository, you might need to manually delete the local repository's contents. The default location of your local Maven repository is `~/.m2/repository` on Unix-based systems and `C:\Users\<Your-Username>\.m2\repository` on Windows.

To delete the local repository and force Maven to redownload dependencies:

1. **Delete the repository folder**: Be cautious with this step as it will remove all locally cached artifacts.

   For Unix-based systems:

   ```bash
   rm -rf ~/.m2/repository
   ```

   For Windows:

   ```cmd
   rmdir /S /Q C:\Users\<Your-Username>\.m2\repository
   ```

2. **Run Maven build**: After deleting the local repository, you can run a Maven build. Maven will re-download all the necessary dependencies.

```bash
mvn clean install
```

Remember to replace `<Your-Username>` with your actual username on Windows. Additionally, always ensure that you have an internet connection while performing these operations, as Maven will need to download dependencies from remote repositories.

Anonymous

InteractiveFloorPlan git:(aroops-branch) ✗ git pull origin main
From https://github.com/dbwelch1996/InteractiveFloorPlan
 * branch   main   -> FETCH_HEAD
hint: You have divergent branches and need to specify how to reconcile them.
hint: You can do so by running one of the following commands sometime before
hint: your next pull:
hint:
hint: git config pull.rebase false # merge
hint: git config pull.rebase true   # rebase
hint: git config pull.ff only   # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
fatal: Need to specify how to reconcile divergent branches.

What do i do if i want to replace aroops-branch with main

ChatGPT

If you want to replace the content of `aroops-branch` with the content from `main`, essentially resetting `aroops-branch` to match `main`, you can follow these steps. Be cautious, as these steps will overwrite any changes on `aroops-branch` that are not in `main`.

First, ensure you are on `aroops-branch`:

```bash
git checkout aroops-branch
```

Then, you can use the following commands to reset your local `aroops-branch` to match `main`:

1. **Fetch the latest changes from the remote without merging them:**

```bash
```

```
git fetch origin main
```

2. **Reset your current branch (`aroops-branch`) to match `main`:**

bash

```
git reset --hard origin/main
```

This command will synchronize **`aroops-branch`** with **`main`**, discarding any divergent changes in **`aroops-branch`**.

3. **Push the changes to the remote, ensuring you are aware this will overwrite the remote branch:**

bash

```
git push --force origin aroops-branch
```

This command forcefully updates the remote **`aroops-branch`** to match **`main`**. Be very careful with this step, as it can overwrite changes on the remote branch. It's generally a good practice to ensure that no one else is working on the same branch or that you have communicated these changes to your team.

Now, **`aroops-branch`** should be identical to **`main`** both locally and remotely.

Anonymous

I will attach my furniture class. Can you make a function that deletes it, so i can eventually implement an eraser mechanic that can delete furniture

```
package com.ECS160.UI;

import java.awt.Image;
import java.awt.Point;
import javax.swing.ImageIcon;

public class Furniture {
    private String name;
    private String imagePath;
    private Point position; // Position of the furniture in the drawing area
    private double rotationAngle; // Stores the rotation angle in degrees
    private double scaleFactor = 1.0; // Default scale

    public Furniture(String name, String imagePath) {
        this.name = name;
        this.imagePath = imagePath;
        this.position = new Point(0, 0); // Default position
    }
```

```
// Getters and Setters
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getImagePath() {
    return imagePath;
}
public void setRotationAngle(double angle) {
    this.rotationAngle = angle;
}

public double getRotationAngle() {
    return rotationAngle;
}

public void setImagePath(String imagePath) {
    this.imagePath = imagePath;
}

public Point getPosition() {
    return position;
}

public void setPosition(Point position) {
    this.position = position;
}
public void moveBy(int dx, int dy) {
    position.translate(dx, dy); // Update the position
}

// Method to get the image
public Image getImage() {
    ImageIcon icon = new ImageIcon(imagePath);
    return icon.getImage();
}


// Methods to get x and y coordinates
public int getX() {
    return position.x;
}

public int getY() {
    return position.y;
}
```

```
    public void setScaleFactor(double scaleFactor) {
        this.scaleFactor = scaleFactor;
    }

    public double getScaleFactor() {
        return scaleFactor;
    }

    // Method to scale the image
    public Image getScaledImage() {
        ImageIcon icon = new ImageIcon(imagePath);
        Image image = icon.getImage();
        int scaledWidth = (int)(image.getWidth(null) * scaleFactor);
        int scaledHeight = (int)(image.getHeight(null) * scaleFactor);
        return image.getScaledInstance(scaledWidth, scaledHeight, Image.SCALE_SMOOTH);
    }
}
```

ChatGPT

To implement an eraser mechanic that can delete furniture objects, you would typically manage this at the level where you maintain a collection of `Furniture` instances, such as a list within your drawing panel or a dedicated furniture manager class. However, I can show you how you might flag an individual `Furniture` object as deleted, which your drawing mechanism can then interpret as a cue to remove the object from the display and potentially from any collections managing active furniture.

First, add a boolean flag to your `Furniture` class to indicate whether the furniture should be considered deleted:

java

```java
package com.ECS160.UI;

import java.awt.Image;
import java.awt.Point;
import javax.swing.ImageIcon;

public class Furniture {
    private String name;
    private String imagePath;
    private Point position;
    private double rotationAngle;
    private double scaleFactor = 1.0;
    private boolean isDeleted = false; // Flag to indicate deletion

    public Furniture(String name, String imagePath) {
        this.name = name;
        this.imagePath = imagePath;
        this.position = new Point(0, 0);
    }

    // Existing getters and setters...

    // Method to flag the furniture as deleted
```

```
    public void delete() {
        this.isDeleted = true;
    }

    // Method to check if the furniture is flagged as deleted
    public boolean isDeleted() {
        return isDeleted;
    }

    // Existing methods...
}
```

With this `delete` method in place, you can call `furniture.delete()` on any `Furniture` instance to mark it as deleted. In your drawing mechanism or furniture management, you would typically iterate over your `Furniture` instances, and when you encounter one that is flagged as deleted (i.e., `isDeleted()` returns `true`), you would remove it from your collection and omit it from the drawing.

Here is a conceptual example of how you might use this in a collection context:

java

```
ArrayList<Furniture> furnitureList = new ArrayList<>();
// Assume furnitureList is populated with Furniture instances

// When erasing, you might set a piece of furniture to deleted
furnitureList.get(indexOfFurnitureToErase).delete();

// Later, when drawing or iterating over the collection:
furnitureList.removeIf(Furniture::isDeleted);
```

This approach ensures that the actual deletion logic (removing a furniture object from your collection and updating the display) is managed at the level of your furniture collection or drawing mechanism, maintaining good separation of concerns. The `Furniture` class itself simply provides a mechanism to flag an object for deletion, leaving it up to the broader application logic to handle the consequences of that flag.

Anonymous

package com.ECS160.UI;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

public class FurnitureManager {
  private final List<Furniture> furnitureList;

  public FurnitureManager() {
    furnitureList = new ArrayList<>();
  }

  public List<Furniture> getFurnitureList() {
    return furnitureList;
  }

```
    // Method to load furniture images from a directory
    public void loadFurnitureImages(String directoryPath) {
        File directory = new File(directoryPath);
        File[] files = directory.listFiles();
        if (files != null) {
            for (File file : files) {
                if (file.isFile() && isImageFile(file)) {
                    String name = extractNameWithoutExtension(file.getName());
                    String imagePath = file.getAbsolutePath();
                    furnitureList.add(new Furniture(name, imagePath));
                }
            }
        }
    }

    // Helper method to check if the file is an image
    private boolean isImageFile(File file) {
        String name = file.getName().toLowerCase();
        return name.endsWith(".png") || name.endsWith(".jpg") || name.endsWith(".jpeg") ||
name.endsWith(".gif");
    }

    // Helper method to extract the file name without extension
    private String extractNameWithoutExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        return (dotIndex == -1) ? fileName : fileName.substring(0, dotIndex);
    }
}
```

This is furniture manager, could i add deleting a furniture in here?

ChatGPT

Yes, you can add functionality to delete furniture in the `FurnitureManager` class. This class manages a list of `Furniture` objects, so you can provide a method to remove a `Furniture` instance from `furnitureList` based on certain criteria, like its name or a direct reference to the `Furniture` object.

Here's how you could implement a method to delete a `Furniture` item by its name (assuming the names are unique). If there are multiple items with the same name and you only want to remove the first occurrence, this method would do so. If you want to remove all items with the same name, you would need to modify the loop to continue searching after finding an initial match.

java

```java
package com.ECS160.UI;

import java.io.File;
import java.util.ArrayList;
```

```
import java.util.List;
import java.util.Objects;

public class FurnitureManager {
    private final List<Furniture> furnitureList;

    public FurnitureManager() {
        furnitureList = new ArrayList<>();
    }

    public List<Furniture> getFurnitureList() {
        return furnitureList;
    }

    public void loadFurnitureImages(String directoryPath) {
        // Existing implementation...
    }

    private boolean isImageFile(File file) {
        // Existing implementation...
    }

    private String extractNameWithoutExtension(String fileName) {
        // Existing implementation...
    }

    // Method to delete a furniture item by name
    public void deleteFurnitureByName(String name) {
        Objects.requireNonNull(name, "Furniture name cannot be null");
        furnitureList.removeIf(furniture -> furniture.getName().equals(name));
    }

    // If you want to delete by object reference instead
    public void deleteFurniture(Furniture furniture) {
        Objects.requireNonNull(furniture, "Furniture object cannot be null");
        furnitureList.remove(furniture);
    }
}
```

In these methods:

- `deleteFurnitureByName`: It iterates over the `furnitureList` and removes any `Furniture` object whose name matches the specified name.
- `deleteFurniture`: It directly removes the `Furniture` object from `furnitureList`. This method requires that the caller has a reference to the `Furniture` object they wish to remove.

With these methods, you can manage furniture deletion operations directly from instances of `FurnitureManager`. This allows you to encapsulate all the furniture-related operations within this class, maintaining a clean and organized structure for your application's code.

Anonymous

package com.ECS160.UI;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Line2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

```
import java.util.ArrayList;
import java.util.List;
import java.awt.dnd.DropTarget;
import java.awt.dnd.DropTargetDropEvent;
import java.awt.dnd.DnDConstants;
import java.awt.datatransfer.Transferable;

public class DrawingPanel extends JPanel {
    private List<Shape> shapes;
    private Shape currentShape;
    private boolean isGridView;
    private BufferedImage gridImage;
    private List<Furniture> placedFurniture;
    private FurnitureManager furnitureManager;
    private Furniture draggedFurniture;
    private Point lastMousePosition;

    public DrawingPanel(boolean isGridView) {
        this.isGridView = isGridView;
        setBackground(Color.WHITE);
        shapes = new ArrayList<>();
        furnitureManager = new FurnitureManager();
        placedFurniture = new ArrayList<>();
        setupMouseListeners();
        setupDropTarget();
        setFocusable(true);
        setupKeyListeners();
    }

    private void setupKeyListeners() {
        addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (draggedFurniture != null) {
                    int rotationIncrement = 5;
                    double scaleIncrement = 0.1;
                    switch (e.getKeyCode()) {
                        case KeyEvent.VK_LEFT:
                            // Rotate counter-clockwise
                            draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() -
rotationIncrement);
                            break;
                        case KeyEvent.VK_RIGHT:
                            // Rotate clockwise
                            draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() +
rotationIncrement);
                            break;
                        case KeyEvent.VK_UP:
                            // Scale up
                            draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() +
scaleIncrement);
```

```
                    break;
                case KeyEvent.VK_DOWN:
                    // Scale down
                    draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() -
scaleIncrement);
                    break;
                case KeyEvent.VK_R:
                    // Add 90 then snap to the nearest 90 degree increment
                    int newAngle = (int)draggedFurniture.getRotationAngle() + 90;
                    // Normalize the angle between 0 and 359 degrees
                    newAngle %= 360;
                    // Snap to the nearest 90 degrees increment
                    int angleRemainder = newAngle % 90;
                    if (angleRemainder >= 45) {
                        newAngle += (90 - angleRemainder); // Snap up to the next increment
                    } else {
                        newAngle -= angleRemainder; // Snap down to the previous increment
                    }

                    draggedFurniture.setRotationAngle(newAngle);
                    break;
            }
            repaint();
        }
    }
  });
}

private void setupMouseListeners() {
  addMouseListener(new MouseAdapter() {
    @Override
    public void mousePressed(MouseEvent e) {
      requestFocusInWindow(); // Request focus when the mouse is pressed
      Furniture selected = getFurnitureAt(e.getPoint());
      if (selected != null) {
        draggedFurniture = selected;
        lastMousePosition = e.getPoint();
      } else {
        currentShape = new Line2D.Double(e.getX(), e.getY(), e.getX(), e.getY());
        shapes.add(currentShape);
      }
    }

    @Override
    public void mouseReleased(MouseEvent e) {
      draggedFurniture = null;
      lastMousePosition = null;
      currentShape = null;
    }
  });
```

```
        addMouseMotionListener(new MouseMotionAdapter() {
            @Override
            public void mouseDragged(MouseEvent e) {
                if (draggedFurniture != null && lastMousePosition != null) {
                    int dx = e.getX() - lastMousePosition.x;
                    int dy = e.getY() - lastMousePosition.y;
                    draggedFurniture.moveBy(dx, dy);
                    lastMousePosition = e.getPoint();
                    repaint();
                } else if (currentShape != null) {
                    Line2D line = (Line2D) currentShape;
                    line.setLine(line.getX1(), line.getY1(), e.getX(), e.getY());
                    repaint();
                }
            }
        });
    }

    private Furniture getFurnitureAt(Point point) {
        for (Furniture furniture : placedFurniture) {
            Rectangle bounds = new Rectangle(furniture.getX(), furniture.getY(),
                    (int)(50 * furniture.getScaleFactor()), (int)(50 * furniture.getScaleFactor())); // Assuming
50x50 is the furniture size
            if (bounds.contains(point)) {
                return furniture;
            }
        }
        return null;
    }

    private void setupDropTarget() {
        setDropTarget(new DropTarget() {
            @Override
            public synchronized void drop(DropTargetDropEvent dtde) {
                dtde.acceptDrop(DnDConstants.ACTION_COPY);
                if (processDrop(dtde)) {
                    dtde.dropComplete(true);
                } else {
                    dtde.dropComplete(false);
                }
                repaint();
            }

            private boolean processDrop(DropTargetDropEvent dtde) {
                Transferable transferable = dtde.getTransferable();
                if (transferable.isDataFlavorSupported(FurnitureTransferable.FURNITURE_FLAVOR)) {
                    try {
                        Furniture furniture = (Furniture)
transferable.getTransferData(FurnitureTransferable.FURNITURE_FLAVOR);
                        Point dropPoint = dtde.getLocation();
                        Furniture newFurniture = new Furniture(furniture.getName(),
```

```
furniture.getImagePath());
                    newFurniture.setPosition(dropPoint);
                    placedFurniture.add(newFurniture);
                    return true;
                } catch (Exception e) {
                    e.printStackTrace();
                    return false;
                }
            }
            return false;
        }
    });
}

public void toggleGridView() {
    isGridView = !isGridView;
    if (isGridView) {
        createGridImage();
    } else {
        gridImage = null;
    }
    repaint();
}
public void saveDrawingPanel() {
    BufferedImage image = new BufferedImage(getWidth(), getHeight(),
BufferedImage.TYPE_INT_ARGB);
    Graphics2D g2d = image.createGraphics();
    paint(g2d);
    g2d.dispose();

    try {
        File outputFile = new File("saved_drawing.png");
        ImageIO.write(image, "png", outputFile);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void clear() {
    shapes.clear();
    placedFurniture.clear();
    repaint();
}

private void createGridImage() {
    int gridSize = 25;
    gridImage = new BufferedImage(getWidth(), getHeight(), BufferedImage.TYPE_INT_ARGB);
    Graphics2D g2d = gridImage.createGraphics();
    g2d.setColor(Color.LIGHT_GRAY);

    for (int x = 0; x < getWidth(); x += gridSize) {
```

```
            g2d.drawLine(x, 0, x, getHeight());
        }

        for (int y = 0; y < getHeight(); y += gridSize) {
            g2d.drawLine(0, y, getWidth(), y);
        }

        g2d.dispose();
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;

        float lineThickness = 2.5f;  // Example line thickness
        g2d.setStroke(new BasicStroke(lineThickness));

        if (isGridView && gridImage != null) {
            g2d.drawImage(gridImage, 0, 0, null);
        }

        for (Shape shape : shapes) {
            g2d.draw(shape);
        }

        for (Furniture furniture : placedFurniture) {
            Image image = furniture.getImage();
            if (image != null) {
                g2d = (Graphics2D) g.create();
                double rotationRadians = Math.toRadians(furniture.getRotationAngle());
                g2d.rotate(rotationRadians, furniture.getX() + 50 / 2, furniture.getY() + 50 / 2); // Assuming
furniture size is 50x50
                g2d.drawImage(image, furniture.getX(), furniture.getY(),
                        (int)(50 * furniture.getScaleFactor()), (int)(50 * furniture.getScaleFactor()), this);
                g2d.dispose();
            }
        }
    }

    @Override
    public Dimension getPreferredSize() {
        return new Dimension(400, 300);
    }
}
```

This is drawingPanel. I will attach FurnitureManager:

```
package com.ECS160.UI;

import java.io.File;
```

```
import java.util.ArrayList;
import java.util.List;

public class FurnitureManager {
    private final List<Furniture> furnitureList;

    public FurnitureManager() {
        furnitureList = new ArrayList<>();
    }

    public List<Furniture> getFurnitureList() {
        return furnitureList;
    }

    // Method to load furniture images from a directory
    public void loadFurnitureImages(String directoryPath) {
        File directory = new File(directoryPath);
        File[] files = directory.listFiles();
        if (files != null) {
            for (File file : files) {
                if (file.isFile() && isImageFile(file)) {
                    String name = extractNameWithoutExtension(file.getName());
                    String imagePath = file.getAbsolutePath();
                    furnitureList.add(new Furniture(name, imagePath));
                }
            }
        }
    }

    // Helper method to check if the file is an image
    private boolean isImageFile(File file) {
        String name = file.getName().toLowerCase();
        return name.endsWith(".png") || name.endsWith(".jpg") || name.endsWith(".jpeg") ||
name.endsWith(".gif");
    }

    // Helper method to extract the file name without extension
    private String extractNameWithoutExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        return (dotIndex == -1) ? fileName : fileName.substring(0, dotIndex);
    }
}
```

Now I will attach my main app file:

```
package com.ECS160.Apps;

import com.ECS160.UI.*;

import javax.swing.*;
```

```
import java.awt.*;

public class FloorDesignApp extends JFrame {

    private DrawingPanel drawingPanel;
    private TopMenuBar menuBar;
    private FurnitureManager furnitureManager; // Add FurnitureManager

    public FloorDesignApp() {
        super("Interactive Floor Design");
        initUI();
    }

    private void initUI() {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (UnsupportedLookAndFeelException | ClassNotFoundException |
InstantiationException | IllegalAccessException e) {
            e.printStackTrace();
        }

        // Initialize the FurnitureManager and load furniture images
        furnitureManager = new FurnitureManager();

        furnitureManager.loadFurnitureImages("src/main/java/com/ECS160/Icons"); // Provide the
correct path

        PageSwapper pageSwapper = new PageSwapper(furnitureManager);
        // Main panel that will hold both the sidebar and the tabbed pane
        JPanel mainPanel = new JPanel(new BorderLayout());

        // Add the tabbed pane to the main panel
        mainPanel.add(pageSwapper, BorderLayout.CENTER);



        // Center panel containing canvas and sidebar
        JPanel centerPanel = new JPanel(new BorderLayout());



        // Create a scroll pane to contain the drawing panel
        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setPreferredSize(new Dimension(1500, 1000)); // Set preferred size for the visible
area

        // Create the drawing panel
        drawingPanel = new DrawingPanel(false);
        drawingPanel.setPreferredSize(new Dimension(1500, 1500)); // Set size of the drawing canvas
        scrollPane.setViewportView(drawingPanel); // Set the drawing panel as the viewport of the scroll
pane
```

```
        centerPanel.add(scrollPane, BorderLayout.CENTER);

        // Add the sidebar with furniture

        Sidebar sidebar = new Sidebar(furnitureManager); // Pass the furnitureManager to Sidebar
        mainPanel.add(sidebar, BorderLayout.WEST);

        // Menu bar
        // Assuming you want a single menu bar for the application window and not individual ones for each tab
        menuBar = new TopMenuBar(pageSwapper, drawingPanel); // Modify TopMenuBar to handle a possibly null DrawingPanel or update logic accordingly
        setJMenuBar(menuBar);

        setContentPane(mainPanel); // Set the main panel as the content pane of the frame
        pack();

        // Set a minimum size for the frame and other properties
        setMinimumSize(new Dimension(1100, 700));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
    }




    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            FloorDesignApp app = new FloorDesignApp();
            app.setVisible(true);
        });
    }
}
```

How do I make it so when I drag furniture and press delete key on keybaord, it deletes the furniture from the drawiing panel?

ChatGPT

To enable the functionality of deleting a furniture item from the `DrawingPanel` when the Delete key is pressed on the keyboard, you need to modify the `setupKeyListeners` method within your `DrawingPanel` class. Here, you'll add a case to handle the `KeyEvent.VK_DELETE` key event. When this event is detected, and if there is a `draggedFurniture` item selected (indicating the user has chosen a furniture item to delete), you will remove that item from the `placedFurniture` list.

Here's how you can update your `setupKeyListeners` method in the `DrawingPanel` class:

java

```java
private void setupKeyListeners() {
    addKeyListener(new KeyAdapter() {
        @Override
        public void keyPressed(KeyEvent e) {
            if (draggedFurniture != null) {
                switch (e.getKeyCode()) {
                    case KeyEvent.VK_LEFT:
                        // Rotate counter-clockwise logic...
                        break;
                    case KeyEvent.VK_RIGHT:
                        // Rotate clockwise logic...
                        break;
                    case KeyEvent.VK_UP:
                        // Scale up logic...
                        break;
                    case KeyEvent.VK_DOWN:
                        // Scale down logic...
                        break;
                    case KeyEvent.VK_R:
                        // Rotate to the nearest 90-degree increment logic...
                        break;
                    case KeyEvent.VK_DELETE:
                        // Remove the selected furniture item
                        placedFurniture.remove(draggedFurniture);
                        draggedFurniture = null; // Clear the reference to avoid re-deletion
                        repaint(); // Redraw to reflect the changes on the UI
                        break;
                }
            }
        }
    });
}
```

In this updated `setupKeyListeners` method, when the Delete key is pressed, the `draggedFurniture` item will be removed from the `placedFurniture` list, which effectively deletes it from the `DrawingPanel`. The `repaint` method call is crucial as it triggers the `DrawingPanel` to be redrawn, reflecting the removal of the furniture item.

Lastly, ensure your `DrawingPanel` is focusable and actually receives keyboard input. You've already called `setFocusable(true)` in the constructor, which is good. Users should click on the `DrawingPanel` first (and you've handled this in `mousePressed` to request focus) to ensure key events are received by the panel.

Anonymous

can you make it backspace insetead

ChatGPT

Certainly! To modify the functionality so that pressing the Backspace key deletes the selected furniture item, you need to update the case within the `keyPressed` method to respond to `KeyEvent.VK_BACK_SPACE` instead of `KeyEvent.VK_DELETE`. Here's how you can update the `setupKeyListeners` method in your `DrawingPanel` class:

java

```
private void setupKeyListeners() {
    addKeyListener(new KeyAdapter() {
        @Override
        public void keyPressed(KeyEvent e) {
            if (draggedFurniture != null) {
                switch (e.getKeyCode()) {
                    case KeyEvent.VK_LEFT:
                        // Rotate counter-clockwise logic...
                        break;
                    case KeyEvent.VK_RIGHT:
                        // Rotate clockwise logic...
                        break;
                    case KeyEvent.VK_UP:
                        // Scale up logic...
                        break;
                    case KeyEvent.VK_DOWN:
                        // Scale down logic...
                        break;
                    case KeyEvent.VK_R:
                        // Rotate to the nearest 90-degree increment logic...
                        break;
                    case KeyEvent.VK_BACK_SPACE:
                        // Remove the selected furniture item
                        placedFurniture.remove(draggedFurniture);
                        draggedFurniture = null; // Clear the reference to avoid re-deletion
                        repaint(); // Redraw to reflect the changes on the UI
                        break;
                }
            }
        }
    });
}
```

With this change, pressing the Backspace key will trigger the deletion of the selected (`draggedFurniture`) item from the `placedFurniture` list, and the drawing panel will be updated accordingly to reflect this change.

Anonymous

Now with the code i gave you could you add an eraser tool to this code

ChatGPT

To implement an eraser tool in your `DrawingPanel` class that allows users to delete furniture items by clicking on them, you'll need to add some additional logic to handle a dedicated eraser mode. When the eraser mode is active, clicking on a furniture item should delete it from the panel. Here's a step-by-step modification you can apply to introduce this functionality:

1. **Add an Eraser Mode Flag**: Include a boolean flag in `DrawingPanel` to indicate whether the eraser mode is active.
2. **Toggle Eraser Mode**: Provide a method to toggle this eraser mode on and off.
3. **Modify MouseListeners**: Adjust your mouse listeners to handle furniture deletion when the eraser mode is active.

Here is how you could modify your `DrawingPanel` class to include these changes:

```
java
```

```
public class DrawingPanel extends JPanel {
    private List<Shape> shapes;
    private Shape currentShape;
    private boolean isGridView;
    private BufferedImage gridImage;
    private List<Furniture> placedFurniture;
    private FurnitureManager furnitureManager;
    private Furniture draggedFurniture;
    private Point lastMousePosition;
    private boolean eraserMode = false; // Flag to indicate eraser mode

    public DrawingPanel(boolean isGridView) {
        // Existing constructor contents...
    }

    // Existing methods...

    // Toggle eraser mode on or off
    public void toggleEraserMode() {
        eraserMode = !eraserMode;
        draggedFurniture = null; // Clear selection when toggling eraser mode
    }

    private void setupMouseListeners() {
        addMouseListener(new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                requestFocusInWindow(); // Request focus when the mouse is pressed

                if (eraserMode) {
                    Furniture toErase = getFurnitureAt(e.getPoint());
                    if (toErase != null) {
                        placedFurniture.remove(toErase);
                        repaint();
                    }
                } else {
                    Furniture selected = getFurnitureAt(e.getPoint());
                    if (selected != null) {
                        draggedFurniture = selected;
                        lastMousePosition = e.getPoint();
                    } else {
                        currentShape = new Line2D.Double(e.getX(), e.getY(), e.getX(),
e.getY());
                        shapes.add(currentShape);
                    }
                }
            }

            @Override
            public void mouseReleased(MouseEvent e) {
                if (!eraserMode) {
                    draggedFurniture = null;
                    lastMousePosition = null;
                    currentShape = null;
                }
            }
        });

        // Existing MouseMotionListener...
    }

    // Existing methods including paintComponent...
}
```

In this implementation:

- The `eraserMode` flag is toggled by calling `toggleEraserMode()`. You might trigger this method from a button or menu item in your GUI.

- When `eraserMode` is true and the user clicks on a piece of furniture, that furniture is removed from `placedFurniture`.
- When not in eraser mode, the panel behaves as before, allowing for dragging furniture.

To integrate the eraser functionality fully, you'd need to add a mechanism in your UI (e.g., a button or menu item in `TopMenuBar` or elsewhere) that calls `drawingPanel.toggleEraserMode()` when clicked. This would allow the user to activate and deactivate the eraser mode. For a complete user experience, consider providing visual feedback to indicate when the eraser mode is active, such as changing the cursor icon or highlighting the eraser tool button.

Anonymous

can you add a section called erase mode under 'edit'. Can you make it so the user knows its selected when its clicked by some sort of visual indicator. Can you make it toggle erase mode: package com.ECS160.UI;

```
import org.apache.pdfbox.Loader;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.rendering.PDFRenderer;
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.swing.*;
import java.awt.event.ActionListener;

public class TopMenuBar extends JMenuBar {

    private ActionListener gridViewToggleListener;
    private PageSwapper pageSwapper;
    private DrawingPanel drawingPanel; // Reference to the DrawingPanel

    public TopMenuBar(PageSwapper pageSwapper, DrawingPanel drawingPanel) {
        this.pageSwapper = pageSwapper;
        this.drawingPanel = drawingPanel; // Set the DrawingPanel

        // File Menu
        JMenu fileMenu = new JMenu("File");
        JMenuItem saveItem = new JMenuItem("Save");
        JMenuItem loadItem = new JMenuItem("Load");
        JMenuItem clearItem = new JMenuItem("Clear");
        JMenuItem exitItem = new JMenuItem("Exit");

        saveItem.addActionListener(e -> saveAction());
        loadItem.addActionListener(e -> System.out.println("Load action"));
        exitItem.addActionListener(e -> exitAction());
        clearItem.addActionListener(e -> clearAction()); // Set action for clearItem
```

```
        fileMenu.add(saveItem);
        fileMenu.add(loadItem);
        fileMenu.addSeparator();
        fileMenu.add(clearItem);
        fileMenu.addSeparator();
        fileMenu.add(exitItem);

        add(fileMenu);

        // Edit Menu
        JMenu editMenu = new JMenu("Edit");
        JMenuItem undoItem = new JMenuItem("Undo");
        JMenuItem redoItem = new JMenuItem("Redo");

        undoItem.addActionListener(e -> undoAction());
        redoItem.addActionListener(e -> redoAction());

        editMenu.add(undoItem);
        editMenu.add(redoItem);

        add(editMenu);

        // View Menu
        JMenu viewMenu = new JMenu("View");
        JMenuItem gridViewItem = new JMenuItem("Grid View");
        gridViewItem.addActionListener(e -> pageSwapper.toggleGridView()); // Directly call the
pageSwapper's method
        viewMenu.add(gridViewItem);
        add(viewMenu);

        JMenu helpMenu = new JMenu("Help");
        JMenuItem userManualItem = new JMenuItem("User Manual");
        userManualItem.addActionListener(e -> viewUserManual());
        helpMenu.add(userManualItem);
        add(helpMenu);
    }
    private void viewUserManual() {
        try {
            String pdfPath = "doc/User Manual.pdf";
            PDDocument document = Loader.loadPDF(new File(pdfPath));
            PDFRenderer renderer = new PDFRenderer(document);

            // Panel to hold all the pages
            JPanel panel = new JPanel();
            panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

            // Render each page and add to the panel
            for (int i = 0; i < document.getNumberOfPages(); i++) {
                BufferedImage image = renderer.renderImage(i);
                JLabel label = new JLabel(new ImageIcon(image));
                panel.add(label);
```

```
            }

            // Scroll pane to allow scrolling through the pages
            JScrollPane scrollPane = new JScrollPane(panel);
            scrollPane.setPreferredSize(new Dimension(600, 1000));

            // Create a JFrame to display the PDF
            JFrame frame = new JFrame("User Manual");
            frame.add(scrollPane);
            frame.pack();
            frame.setLocationRelativeTo(null);
            frame.setVisible(true);

            document.close();
        } catch (IOException ex) {
            ex.printStackTrace();
            // Handle the exception as needed
        }
    }

    // Setter method to set the pageSwapper instance
    public void setPageSwapper(PageSwapper pageSwapper) {
        this.pageSwapper = pageSwapper;
    }

    // Method to trigger grid view toggle action
    public void toggleGridView() {
        if (pageSwapper != null) {
            pageSwapper.toggleGridView();
        }
    }


    // Method to clear the drawing panel
    private void clearAction() {
        if (pageSwapper != null) {
            pageSwapper.clear();
        }
    }

    private void saveAction() {
        if (drawingPanel != null) {
            drawingPanel.saveDrawingPanel(); // Call the method to save the drawing panel
        }
    }

    private void exitAction() {
        System.out.println("Exit action");
        System.exit(0);
    }
```

```
    private void undoAction() {
        System.out.println("Undo action");
    }

    private void redoAction() {
        System.out.println("Redo action");
    }
}
```

ChatGPT

To add an "Erase Mode" option under the "Edit" menu in your `TopMenuBar` class, which toggles the erase mode in `DrawingPanel` and provides a visual indicator when it's active, you can introduce a checkable menu item (`JCheckBoxMenuItem`). When checked, the erase mode is activated, and when unchecked, it's deactivated. The checkmark itself serves as the visual indicator. Here's how you can modify your `TopMenuBar` class to include this feature:

java

```java
public class TopMenuBar extends JMenuBar {

    private ActionListener gridViewToggleListener;
    private PageSwapper pageSwapper;
    private DrawingPanel drawingPanel; // Reference to the DrawingPanel
    private JCheckBoxMenuItem eraseModeItem; // Checkable menu item for erase mode

    public TopMenuBar(PageSwapper pageSwapper, DrawingPanel drawingPanel) {
        this.pageSwapper = pageSwapper;
        this.drawingPanel = drawingPanel;

        // File Menu
        JMenu fileMenu = new JMenu("File");
        // ...existing menu items...

        add(fileMenu);

        // Edit Menu
        JMenu editMenu = new JMenu("Edit");
        // ...existing undo and redo menu items...

        // Erase Mode toggle
        eraseModeItem = new JCheckBoxMenuItem("Erase Mode");
        eraseModeItem.addActionListener(e -> toggleEraseMode());
        editMenu.add(eraseModeItem);

        add(editMenu);

        // View Menu and Help Menu
        // ...existing view and help menu items...
    }

    // Toggle erase mode on or off and provide visual feedback
    private void toggleEraseMode() {
        if (drawingPanel != null) {
            drawingPanel.toggleEraserMode();
            eraseModeItem.setSelected(drawingPanel.isEraserMode()); // Update the check
state to reflect the current mode
        }
    }

    // ...remaining methods (like saveAction, exitAction, etc.)...
}
```

Then, in your `DrawingPanel` class, ensure you have methods to toggle the eraser mode and to check its state:

java

```java
public class DrawingPanel extends JPanel {
    // ...existing attributes...

    private boolean eraserMode = false; // Flag to indicate eraser mode

    // ...existing constructor and methods...

    // Toggle eraser mode on or off
    public void toggleEraserMode() {
        eraserMode = !eraserMode;
    }

    // Check if eraser mode is active
    public boolean isEraserMode() {
        return eraserMode;
    }

    // ...remaining methods (like paintComponent, setupMouseListeners, etc.)...
}
```

Now, when the user selects "Erase Mode" from the "Edit" menu, the eraser mode in `DrawingPanel` will be toggled, and the checkbox will visually indicate whether the mode is active.

Anonymous

when eraseMode is toggled, can you change the mouse pointer icon to eraser.png in icons/

ChatGPT

To change the mouse pointer icon to an eraser image when the erase mode is toggled, you'll need to modify the `toggleEraserMode` method in the `DrawingPanel` class to update the cursor accordingly. First, ensure that you have an image named `eraser.png` in the `icons/` directory within your project.

Here's how you can update the `DrawingPanel` class to change the cursor to an eraser icon when erase mode is activated and revert it to the default cursor when the mode is deactivated:

java

```java
public class DrawingPanel extends JPanel {
    // ...existing attributes...

    private boolean eraserMode = false; // Flag to indicate eraser mode

    // ...existing constructor and methods...

    // Toggle eraser mode on or off
    public void toggleEraserMode() {
        eraserMode = !eraserMode;
        if (eraserMode) {
            // Change cursor to eraser icon
            Toolkit toolkit = Toolkit.getDefaultToolkit();
            Image eraserImage = toolkit.getImage("icons/eraser.png");
            Cursor eraserCursor = toolkit.createCustomCursor(eraserImage, new Point(0, 0),
"eraser");
```

```
            setCursor(eraserCursor);
        } else {
            // Revert cursor to default
            setCursor(Cursor.getDefaultCursor());
        }
    }

    // Check if eraser mode is active
    public boolean isEraserMode() {
        return eraserMode;
    }

    // ...remaining methods...
}
```

In this updated version of `toggleEraserMode`, the method checks the current state of `eraserMode`. If the mode is activated, it loads the `eraser.png` image and sets it as the custom cursor for the `DrawingPanel`. When the mode is deactivated, it resets the cursor to the default system cursor.

Make sure the path `icons/eraser.png` correctly points to your eraser image relative to the working directory of your application. If your application's working directory is not the project root, you might need to adjust the path accordingly.

Anonymous

```
package com.ECS160.UI;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;



public class PageSwapper extends JPanel {
   private JTabbedPane tabbedPane;

   private JPanel controlPanel;
   private FurnitureManager furnitureManager;
   private int pageCount;
   private int initialPageCount;

   public PageSwapper(FurnitureManager furnitureManager) {
      this.furnitureManager = furnitureManager;
      this.initialPageCount = 1;
      this.pageCount = initialPageCount; // Initialize page count
      setLayout(new BorderLayout());

      controlPanel = getControlPanel();
      add(controlPanel, BorderLayout.NORTH);

      tabbedPane = new JTabbedPane();
      initializeTabs();
   }
```

```
    private void initializeTabs() {
        for (int i = 1; i <= initialPageCount; i++) {
            JPanel pagePanel = createPage(i); // Create a panel for each tab/page
            tabbedPane.addTab("Page " + i, pagePanel);
        }
        add(tabbedPane, BorderLayout.CENTER);
    }


    private JPanel createPage(int pageNumber) {
        JPanel mainPanel = new JPanel(new BorderLayout());

        // Example setup for a drawing panel in each tab
        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setPreferredSize(new Dimension(600, 400)); // Set preferred size

        DrawingPanel drawingPanel = new DrawingPanel(false);
        drawingPanel.setPreferredSize(new Dimension(1000,
                1000));
        scrollPane.setViewportView(drawingPanel);

        mainPanel.add(scrollPane, BorderLayout.CENTER);

        // You can add more components specific to each page here

        return mainPanel;
    }

    public void addPage() {
        pageCount++;
        JPanel pagePanel = createPage(pageCount); // Modified to use pageCount
        tabbedPane.addTab("Page " + pageCount, pagePanel);
        tabbedPane.setSelectedIndex(tabbedPane.getTabCount() - 1); // Switch to the new tab
immediately
    }

    public void renameCurrentPage(String newTitle) {
        int selectedIndex = tabbedPane.getSelectedIndex();
        if (selectedIndex != -1) {
            tabbedPane.setTitleAt(selectedIndex, newTitle);
        }
    }

    public JPanel getControlPanel() {
        JPanel controlPanel = new JPanel(new BorderLayout());

        JButton addPageButton = new JButton("Add Page");
        addPageButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
            addPage();
        }
    });
    controlPanel.add(addPageButton, BorderLayout.EAST);

    JButton renamePageButton = new JButton("Rename Current Page");
    renamePageButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            String newTitle = JOptionPane.showInputDialog(PageSwapper.this, "Enter new title:");
            if (newTitle != null && !newTitle.isEmpty()) {
                renameCurrentPage(newTitle);
            }
        }
    });
    controlPanel.add(renamePageButton, BorderLayout.WEST);

    return controlPanel;
}

private DrawingPanel getCurrentDrawingPanel() {
    int selectedIndex = tabbedPane.getSelectedIndex();
    if (selectedIndex != -1) {
        JPanel mainPanel = (JPanel) tabbedPane.getComponentAt(selectedIndex);
        JScrollPane scrollPane = (JScrollPane) mainPanel.getComponent(0);
        return (DrawingPanel) scrollPane.getViewport().getView();
    }
    return null;
}

public void clear() {
    getCurrentDrawingPanel().clear();
}

public void toggleGridView() {
    getCurrentDrawingPanel().toggleGridView();
}


}
```

Can you make a function that returns all drawingPanels in tabbedPanel

ChatGPT

To create a function in the `PageSwapper` class that returns all `DrawingPanel` instances contained within the `tabbedPane`, you can iterate over each tab, assuming each tab's main panel contains a `DrawingPanel` inside a `JScrollPane` as constructed in your `createPage` method. Here's how you can implement this function:

java

```java
public class PageSwapper extends JPanel {
    private JTabbedPane tabbedPane;
    // Other attributes and constructor...

    // Function to retrieve all DrawingPanels from the tabbedPane
    public List<DrawingPanel> getAllDrawingPanels() {
        List<DrawingPanel> drawingPanels = new ArrayList<>();

        for (int i = 0; i < tabbedPane.getTabCount(); i++) {
            JPanel mainPanel
```

Anonymous

actually, can you make a funciton inside this that toggles Eraser mode for all panels inside tabbed panel

package com.ECS160.UI;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class PageSwapper extends JPanel {
  private JTabbedPane tabbedPane;

  private JPanel controlPanel;
  private FurnitureManager furnitureManager;
  private int pageCount;
  private int initialPageCount;

  public PageSwapper(FurnitureManager furnitureManager) {
    this.furnitureManager = furnitureManager;
    this.initialPageCount = 1;
    this.pageCount = initialPageCount; // Initialize page count
    setLayout(new BorderLayout());

    controlPanel = getControlPanel();
    add(controlPanel, BorderLayout.NORTH);

    tabbedPane = new JTabbedPane();
    initializeTabs();
  }

  private void initializeTabs() {
    for (int i = 1; i <= initialPageCount; i++) {
      JPanel pagePanel = createPage(i); // Create a panel for each tab/page
      tabbedPane.addTab("Page " + i, pagePanel);

```
        }
        add(tabbedPane, BorderLayout.CENTER);
    }


    private JPanel createPage(int pageNumber) {
        JPanel mainPanel = new JPanel(new BorderLayout());

        // Example setup for a drawing panel in each tab
        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setPreferredSize(new Dimension(600, 400)); // Set preferred size

        DrawingPanel drawingPanel = new DrawingPanel(false);
        drawingPanel.setPreferredSize(new Dimension(1000,
                1000));
        scrollPane.setViewportView(drawingPanel);

        mainPanel.add(scrollPane, BorderLayout.CENTER);

        // You can add more components specific to each page here

        return mainPanel;
    }

    public void addPage() {
        pageCount++;
        JPanel pagePanel = createPage(pageCount); // Modified to use pageCount
        tabbedPane.addTab("Page " + pageCount, pagePanel);
        tabbedPane.setSelectedIndex(tabbedPane.getTabCount() - 1); // Switch to the new tab
immediately
    }

    public void renameCurrentPage(String newTitle) {
        int selectedIndex = tabbedPane.getSelectedIndex();
        if (selectedIndex != -1) {
            tabbedPane.setTitleAt(selectedIndex, newTitle);
        }
    }

    public JPanel getControlPanel() {
        JPanel controlPanel = new JPanel(new BorderLayout());

        JButton addPageButton = new JButton("Add Page");
        addPageButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                addPage();
            }
        });
        controlPanel.add(addPageButton, BorderLayout.EAST);
```

```
        JButton renamePageButton = new JButton("Rename Current Page");
        renamePageButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String newTitle = JOptionPane.showInputDialog(PageSwapper.this, "Enter new title:");
                if (newTitle != null && !newTitle.isEmpty()) {
                    renameCurrentPage(newTitle);
                }
            }
        });
        controlPanel.add(renamePageButton, BorderLayout.WEST);

        return controlPanel;
    }

    private DrawingPanel getCurrentDrawingPanel() {
        int selectedIndex = tabbedPane.getSelectedIndex();
        if (selectedIndex != -1) {
            JPanel mainPanel = (JPanel) tabbedPane.getComponentAt(selectedIndex);
            JScrollPane scrollPane = (JScrollPane) mainPanel.getComponent(0);
            return (DrawingPanel) scrollPane.getViewport().getView();
        }
        return null;
    }

    public void clear() {
        getCurrentDrawingPanel().clear();
    }

    public void toggleGridView() {
        getCurrentDrawingPanel().toggleGridView();
    }


}
```

ChatGPT

To implement a function in the `PageSwapper` class that toggles the eraser mode for all `DrawingPanel` instances within each tab of the `tabbedPane`, you can iterate over all the tabs, get each `DrawingPanel`, and toggle its eraser mode. Here's how you can add this function:

java

```java
public class PageSwapper extends JPanel {
    private JTabbedPane tabbedPane;
    // Other attributes and methods...

    // Function to toggle eraser mode for all DrawingPanels in the tabbedPane
    public void toggleEraserModeForAll() {
        for (int i = 0; i < tabbedPane.getTabCount(); i++) {
            JPanel mainPanel = (JPanel) tabbedPane.getComponentAt(i);
            JScrollPane scrollPane = (JScrollPane) mainPanel.getComponent(0);
```

```
            DrawingPanel drawingPanel = (DrawingPanel) scrollPane.getViewport().getView();
            drawingPanel.toggleEraserMode();
        }
    }

    // Existing methods...
}
```

This `toggleEraserModeForAll` method goes through each tab, retrieves the associated `DrawingPanel`, and toggles its eraser mode. When you call this method, all `DrawingPanel` instances across all tabs will have their eraser mode toggled to the new state.

You would typically call this `toggleEraserModeForAll` method from somewhere else in your UI, perhaps linked to a button or menu item that the user can interact with. For example, if you want to add this functionality to a menu item in the `TopMenuBar` class, you would need to pass a reference to the `PageSwapper` to `TopMenuBar` and then call `pageSwapper.toggleEraserModeForAll()` when that menu item is activated.

Anonymous

```
public void toggleEraseMode() {
    eraseMode = !eraseMode;
    if (eraseMode) {
        // Change cursor to eraser icon
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        Image eraserImage = toolkit.getImage("src/main/java/com/ECS160/UI/eraser.png");
        Cursor eraserCursor = toolkit.createCustomCursor(eraserImage, new Point(0, 0), "eraser");
        setCursor(eraserCursor);
    } else {
        // Revert cursor to default
        setCursor(Cursor.getDefaultCursor());
    }
}
```

Can you scale the image down its wayy too big rn

ChatGPT

To scale the eraser icon down because it's too large, you can use the `Image.getScaledInstance` method to create a scaled version of the image. You'll need to specify the desired width and height for the scaled image. Here's how you can modify the `toggleEraseMode` method in your `DrawingPanel` class to scale the eraser cursor image:

java

```
public void toggleEraseMode() {
    eraseMode = !eraseMode;
    if (eraseMode) {
        // Change cursor to eraser icon
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        Image eraserImage = toolkit.getImage("src/main/java/com/ECS160/UI/eraser.png");

        // Scale the image to a smaller size, e.g., 32x32 pixels
```

```
        Image scaledEraserImage = eraserImage.getScaledInstance(32, 32, Image.SCALE_SMOOTH);
        Cursor eraserCursor = toolkit.createCustomCursor(scaledEraserImage, new Point(0, 0),
"eraser");
        setCursor(eraserCursor);
    } else {
        // Revert cursor to default
        setCursor(Cursor.getDefaultCursor());
    }
}
```

In this modified method, `eraserImage.getScaledInstance(32, 32, Image.SCALE_SMOOTH)` scales the eraser image to 32x32 pixels. You can adjust the `32, 32` values to whatever dimensions you find appropriate for your cursor size. The `Image.SCALE_SMOOTH` option is used for a smoother, higher-quality scaling algorithm. After scaling, this image is then used to create the custom cursor.

Anonymous

in sidebar, I have a function called setEraseMode:
public void setEraseMode(boolean eraseMode) {

}

Can you make it so if eraseMode is set to false, then the cursor switches to cancelIcon.png and the sidebar is not usable to click on furniture. Thank u in advance. Here is the code for sidebar:
package com.ECS160.UI;

```
import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class Sidebar extends JPanel {
    private FurnitureManager furnitureManager;
    private PageSwapper pageSwapper;

    public Sidebar(FurnitureManager furnitureManager, PageSwapper pageSwapper) {
        this.furnitureManager = furnitureManager;
        setLayout(new GridLayout(0, 3, 10, 10));
        setBackground(Color.LIGHT_GRAY);
        setPreferredSize(new Dimension(200, 600));

        this.pageSwapper = pageSwapper;

        populateWithFurniture();
    }

    private void populateWithFurniture() {
        Border border = BorderFactory.createLineBorder(Color.BLACK, 1);
        FurnitureTransferHandler transferHandler = new FurnitureTransferHandler();
```

```
        for (Furniture furniture : furnitureManager.getFurnitureList()) {
            ImageIcon icon = new ImageIcon(furniture.getImagePath());
            Image scaledImage = icon.getImage().getScaledInstance(50, 50, Image.SCALE_SMOOTH);
            JLabel furnitureLabel = new JLabel(new ImageIcon(scaledImage));
            furnitureLabel.setBorder(border);
            furnitureLabel.setOpaque(true);
            furnitureLabel.setBackground(Color.WHITE);
            furnitureLabel.setToolTipText(furniture.getName());
            furnitureLabel.setTransferHandler(transferHandler);

            // Associate the furniture with the label
            furnitureLabel.putClientProperty("furniture", furniture);

            furnitureLabel.addMouseListener(new MouseAdapter() {
                @Override
                public void mousePressed(MouseEvent e) {
                    JComponent c = (JComponent) e.getSource();
                    TransferHandler handler = c.getTransferHandler();
                    handler.exportAsDrag(c, e, TransferHandler.COPY);
                }
            });

            add(furnitureLabel);
        }
    }

    public void setEraseMode(boolean eraseMode) {

    }
}
```

ChatGPT

To implement the `setEraseMode` functionality within your `Sidebar` class, so that when `eraseMode` is set to false the cursor switches to a `cancelIcon.png` and disables the ability to click on furniture, you need to update the cursor and modify the mouse listeners accordingly. Here's how you can adjust your `Sidebar` class:

1. **Update the Cursor:** Change the cursor to `cancelIcon.png` when `eraseMode` is false.
2. **Disable Furniture Interaction:** Prevent the furniture from being draggable by disabling/enabling the mouse listeners based on the `eraseMode` state.

Here's the modified `Sidebar` class with the `setEraseMode` method implementation:

java

```java
package com.ECS160.UI;

import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;
```

```
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import java.util.List;

public class Sidebar extends JPanel {
    private FurnitureManager furnitureManager;
    private List<JLabel> furnitureLabels;
    private MouseAdapter mouseAdapter;

    public Sidebar(FurnitureManager furnitureManager, PageSwapper pageSwapper) {
        this.furnitureManager = furnitureManager;
        this.furnitureLabels = new ArrayList<>();
        setLayout(new GridLayout(0, 3, 10, 10));
        setBackground(Color.LIGHT_GRAY);
        setPreferredSize(new Dimension(200, 600));
        populateWithFurniture();
    }

    private void populateWithFurniture() {
        Border border = BorderFactory.createLineBorder(Color.BLACK, 1);
        FurnitureTransferHandler transferHandler = new FurnitureTransferHandler();

        mouseAdapter = new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                JComponent c = (JComponent) e.getSource();
                TransferHandler handler = c.getTransferHandler();
                handler.exportAsDrag(c, e, TransferHandler.COPY);
            }
        };

        for (Furniture furniture : furnitureManager.getFurnitureList()) {
            ImageIcon icon = new ImageIcon(furniture.getImagePath());
            Image scaledImage = icon.getImage().getScaledInstance(50, 50,
Image.SCALE_SMOOTH);
            JLabel furnitureLabel = new JLabel(new ImageIcon(scaledImage));
            furnitureLabel.setBorder(border);
            furnitureLabel.setOpaque(true);
            furnitureLabel.setBackground(Color.WHITE);
            furnitureLabel.setToolTipText(furniture.getName());
            furnitureLabel.setTransferHandler(transferHandler);
            furnitureLabel.addMouseListener(mouseAdapter);
            furnitureLabel.putClientProperty("furniture", furniture);
            furnitureLabels.add(furnitureLabel);
            add(furnitureLabel);
        }
    }

    public void setEraseMode(boolean eraseMode) {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        Cursor cursor;
        if (!eraseMode) {
            Image cancelImage =
toolkit.getImage("src/main/java/com/ECS160/UI/cancelIcon.png");
            cursor = toolkit.createCustomCursor(cancelImage, new Point(0, 0), "cancel");
            disableFurnitureInteraction();
        } else {
            cursor = Cursor.getDefaultCursor();
            enableFurnitureInteraction();
        }
        setCursor(cursor);
    }

    private void disableFurnitureInteraction() {
        for (JLabel label : furnitureLabels) {
            for (MouseAdapter adapter : label.getMouseListeners()) {
                label.removeMouseListener(adapter);
            }
        }
    }

    private void enableFurnitureInteraction() {
        for (JLabel label : furnitureLabels) {
```

```
            label.addMouseListener(mouseAdapter);
        }
    }
}
```

This implementation provides the following functionality:

- When `eraseMode` is false, the cursor changes to `cancelIcon.png`, and mouse interaction with the furniture labels is disabled.
- When `eraseMode` is true, the cursor reverts to the default, and the furniture labels can be interacted with again.

The cursor image `cancelIcon.png` should be correctly sized for a cursor. Ensure its size is appropriate (typically around 32x32 pixels or similar) and located at the specified path (`"src/main/java/com/ECS160/UI/cancelIcon.png"`). Adjust the path as necessary based on your project structure.

Anonymous

Required type:
MouseListener
Provided:
MouseAdapter

I get this error for
private void disableFurnitureInteraction() {
    for (JLabel label : furnitureLabels) {
        for (MouseAdapter adapter : label.getMouseListeners()) {
            label.removeMouseListener(adapter);
        }
    }
}

ChatGPT

The issue you're encountering is due to a type mismatch: `label.getMouseListeners()` returns an array of `MouseListener`, but you're trying to assign each element to a `MouseAdapter` variable. While `MouseAdapter` is a subclass of `MouseListener` and thus you can add a `MouseAdapter` as a `MouseListener`, the opposite—treating a `MouseListener` as a `MouseAdapter`—is not directly permissible without casting, and you typically should not need to cast in this context.

When you disable the furniture interaction, you want to remove the specific `MouseListener` you added, which is stored in `mouseAdapter`. However, there's no need to iterate through all listeners; you can just remove the one you know about. Here's how you can adjust your `disableFurnitureInteraction` and `enableFurnitureInteraction` methods correctly:

java

```java
private void disableFurnitureInteraction() {
    for (JLabel label : furnitureLabels) {
        label.removeMouseListener(mouseAdapter);
```

```
    }
}

private void enableFurnitureInteraction() {
    for (JLabel label : furnitureLabels) {
        label.addMouseListener(mouseAdapter);
    }
}
```

This adjusted code properly removes or adds the specific `mouseAdapter` from each furniture label without iterating through all listeners. By directly referencing `mouseAdapter`, you avoid any type mismatch issues and ensure that only the interaction intended to be toggled is affected.

Anonymous

```
package com.ECS160.UI;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.Line2D;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.awt.dnd.DropTarget;
import java.awt.dnd.DropTargetDropEvent;
import java.awt.dnd.DnDConstants;
import java.awt.datatransfer.Transferable;

public class DrawingPanel extends JPanel {
    private List<Shape> shapes;
    private Shape currentShape;
    private boolean isGridView;
    private BufferedImage gridImage;
    private List<Furniture> placedFurniture;
    private FurnitureManager furnitureManager;
    private Furniture draggedFurniture;
    private Point lastMousePosition;
    private boolean eraseMode = false; // Flag to indicate erase mode

    public DrawingPanel(boolean isGridView) {
        this.isGridView = isGridView;
        setBackground(Color.WHITE);
        shapes = new ArrayList<>();
        furnitureManager = new FurnitureManager();
        placedFurniture = new ArrayList<>();
        setupMouseListeners();
        setupDropTarget();
        setFocusable(true);
```

```
        setupKeyListeners();
    }

    private void setupKeyListeners() {
        addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (draggedFurniture != null) {
                    int rotationIncrement = 5;
                    double scaleIncrement = 0.1;
                    switch (e.getKeyCode()) {
                        case KeyEvent.VK_LEFT:
                            // Rotate counter-clockwise
                            draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() -
rotationIncrement);
                            break;
                        case KeyEvent.VK_RIGHT:
                            // Rotate clockwise
                            draggedFurniture.setRotationAngle(draggedFurniture.getRotationAngle() +
rotationIncrement);
                            break;
                        case KeyEvent.VK_UP:
                            // Scale up
                            draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() +
scaleIncrement);
                            break;
                        case KeyEvent.VK_DOWN:
                            // Scale down
                            draggedFurniture.setScaleFactor(draggedFurniture.getScaleFactor() -
scaleIncrement);
                            break;
                        case KeyEvent.VK_R:
                            // Add 90 then snap to the nearest 90 degree increment
                            int newAngle = (int)draggedFurniture.getRotationAngle() + 90;
                            // Normalize the angle between 0 and 359 degrees
                            newAngle %= 360;
                            // Snap to the nearest 90 degrees increment
                            int angleRemainder = newAngle % 90;
                            if (angleRemainder >= 45) {
                                newAngle += (90 - angleRemainder); // Snap up to the next increment
                            } else {
                                newAngle -= angleRemainder; // Snap down to the previous increment
                            }

                            draggedFurniture.setRotationAngle(newAngle);
                            break;
                        case KeyEvent.VK_BACK_SPACE:
                            // Remove the selected furniture item
                            placedFurniture.remove(draggedFurniture);
                            draggedFurniture = null; // Clear the reference to avoid re-deletion
                            repaint(); // Redraw to reflect the changes on the UI
```

```
                    break;
                }
                repaint();
            }
        }
    });
}

private void setupMouseListeners() {
    addMouseListener(new MouseAdapter() {
        @Override
        public void mousePressed(MouseEvent e) {
            requestFocusInWindow(); // Request focus when the mouse is pressed

            if (eraseMode) {
                Furniture toErase = getFurnitureAt(e.getPoint());
                if (toErase != null) {
                    placedFurniture.remove(toErase);
                    repaint();
                }
            } else {
                Furniture selected = getFurnitureAt(e.getPoint());
                if (selected != null) {
                    draggedFurniture = selected;
                    lastMousePosition = e.getPoint();
                } else {
                    currentShape = new Line2D.Double(e.getX(), e.getY(), e.getX(), e.getY());
                    shapes.add(currentShape);
                }
            }
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            if (!eraseMode) {
                draggedFurniture = null;
                lastMousePosition = null;
                currentShape = null;
            }
        }
    });

    addMouseMotionListener(new MouseMotionAdapter() {
        @Override
        public void mouseDragged(MouseEvent e) {
            if (draggedFurniture != null && lastMousePosition != null) {
                int dx = e.getX() - lastMousePosition.x;
                int dy = e.getY() - lastMousePosition.y;
                draggedFurniture.moveBy(dx, dy);
                lastMousePosition = e.getPoint();
                repaint();
```

```
            } else if (currentShape != null) {
                Line2D line = (Line2D) currentShape;
                line.setLine(line.getX1(), line.getY1(), e.getX(), e.getY());
                repaint();
            }
        }
    });
}

private Furniture getFurnitureAt(Point point) {
    for (Furniture furniture : placedFurniture) {
        Rectangle bounds = new Rectangle(furniture.getX(), furniture.getY(),
                (int)(50 * furniture.getScaleFactor()), (int)(50 * furniture.getScaleFactor())); // Assuming
50x50 is the furniture size
        if (bounds.contains(point)) {
            return furniture;
        }
    }
    return null;
}

private void setupDropTarget() {
    setDropTarget(new DropTarget() {
        @Override
        public synchronized void drop(DropTargetDropEvent dtde) {
            dtde.acceptDrop(DnDConstants.ACTION_COPY);
            if (processDrop(dtde)) {
                dtde.dropComplete(true);
            } else {
                dtde.dropComplete(false);
            }
            repaint();
        }

        private boolean processDrop(DropTargetDropEvent dtde) {
            Transferable transferable = dtde.getTransferable();
            if (transferable.isDataFlavorSupported(FurnitureTransferable.FURNITURE_FLAVOR)) {
                try {
                    Furniture furniture = (Furniture)
transferable.getTransferData(FurnitureTransferable.FURNITURE_FLAVOR);
                    Point dropPoint = dtde.getLocation();
                    Furniture newFurniture = new Furniture(furniture.getName(),
furniture.getImagePath());
                    newFurniture.setPosition(dropPoint);
                    placedFurniture.add(newFurniture);
                    return true;
                } catch (Exception e) {
                    e.printStackTrace();
                    return false;
                }
            }
```

```
                return false;
            }
        });
    }

    public void toggleGridView() {
        isGridView = !isGridView;
        if (isGridView) {
            createGridImage();
        } else {
            gridImage = null;
        }
        repaint();
    }
    public void saveDrawingPanel() {
        BufferedImage image = new BufferedImage(getWidth(), getHeight(),
BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2d = image.createGraphics();
        paint(g2d);
        g2d.dispose();

        try {
            File outputFile = new File("saved_drawing.png");
            ImageIO.write(image, "png", outputFile);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public void clear() {
        shapes.clear();
        placedFurniture.clear();
        repaint();
    }

    private void createGridImage() {
        int gridSize = 25;
        gridImage = new BufferedImage(getWidth(), getHeight(), BufferedImage.TYPE_INT_ARGB);
        Graphics2D g2d = gridImage.createGraphics();
        g2d.setColor(Color.LIGHT_GRAY);

        for (int x = 0; x < getWidth(); x += gridSize) {
            g2d.drawLine(x, 0, x, getHeight());
        }

        for (int y = 0; y < getHeight(); y += gridSize) {
            g2d.drawLine(0, y, getWidth(), y);
        }

        g2d.dispose();
    }
```

```
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g;

        float lineThickness = 2.5f;  // Example line thickness
        g2d.setStroke(new BasicStroke(lineThickness));

        if (isGridView && gridImage != null) {
            g2d.drawImage(gridImage, 0, 0, null);
        }

        for (Shape shape : shapes) {
            g2d.draw(shape);
        }

        for (Furniture furniture : placedFurniture) {
            Image image = furniture.getImage();
            if (image != null) {
                g2d = (Graphics2D) g.create();
                double rotationRadians = Math.toRadians(furniture.getRotationAngle());
                g2d.rotate(rotationRadians, furniture.getX() + 50 / 2, furniture.getY() + 50 / 2); // Assuming
furniture size is 50x50
                g2d.drawImage(image, furniture.getX(), furniture.getY(),
                        (int)(50 * furniture.getScaleFactor()), (int)(50 * furniture.getScaleFactor()), this);
                g2d.dispose();
            }
        }
    }

    @Override
    public Dimension getPreferredSize() {
        return new Dimension(400, 300);
    }


    public void setEraseMode(boolean eraseMode) {
        this.eraseMode = eraseMode;
        if (eraseMode) {
            // Change cursor to eraser icon
            Toolkit toolkit = Toolkit.getDefaultToolkit();
            Image eraserImage = toolkit.getImage("src/main/java/com/ECS160/UI/eraser.png");

            // Scale the image to a smaller size, e.g., 32x32 pixels
            Image scaledEraserImage = eraserImage.getScaledInstance(32, 32, Image.SCALE_SMOOTH);
            Cursor eraserCursor = toolkit.createCustomCursor(scaledEraserImage, new Point(0, 0),
"eraser");
            setCursor(eraserCursor);
        } else {
            // Revert cursor to default
```

```
            setCursor(Cursor.getDefaultCursor());
        }
    }
    public boolean isEraseMode() {
        return eraseMode;
    }


}
```

Can you make eraseMode also delete lines that are drawn when they are clicked on

ChatGPT

To enable the `eraseMode` to delete lines when they are clicked on in the `DrawingPanel`, you need to adjust the mouse listener to check if a line (or shape) intersects with a small rectangle (or another shape) centered around the mouse click. If it does, you can remove that line from the shapes list. Here's how you can implement this in your `DrawingPanel` class:

1. Modify the `mousePressed` method to add logic for removing shapes in erase mode.
2. Determine if the mouse click is close to any shape.

java

```java
private void setupMouseListeners() {
    addMouseListener(new MouseAdapter() {
        @Override
        public void mousePressed(MouseEvent e) {
            requestFocusInWindow(); // Request focus when the mouse is pressed

            if (eraseMode) {
                // Determine if a furniture item or shape is clicked and remove it
                Furniture toErase = getFurnitureAt(e.getPoint());
                if (toErase != null) {
                    placedFurniture.remove(toErase);
                } else {
                    // Create a small rectangle around the click point to detect shape
intersection
                    Rectangle clickBounds = new Rectangle(e.getX() - 2, e.getY() - 2, 4, 4);
                    Shape toRemove = null;
                    for (Shape shape : shapes) {
                        // Check if the shape intersects the click bounds
                        if (shape.intersects(clickBounds)) {
                            toRemove = shape;
                            break;
                        }
                    }
                    if (toRemove != null) {
                        shapes.remove(toRemove);
                    }
                }
                repaint();
            } else {
                // Handle normal behavior when not in erase mode
                Furniture selected = getFurnitureAt(e.getPoint());
                if (selected != null) {
                    draggedFurniture = selected;
                    lastMousePosition = e.getPoint();
                } else {
                    currentShape = new Line2D.Double(e.getX(), e.getY(), e.getX(),
e.getY());
```

```
                    shapes.add(currentShape);
                }
            }
        }

        @Override
        public void mouseReleased(MouseEvent e) {
            if (!eraseMode) {
                draggedFurniture = null;
                lastMousePosition = null;
                currentShape = null;
            }
        }
    });

    addMouseMotionListener(new MouseMotionAdapter() {
        @Override
        public void mouseDragged(MouseEvent e) {
            if (!eraseMode && draggedFurniture != null && lastMousePosition != null) {
                int dx = e.getX() - lastMousePosition.x;
                int dy = e.getY() - lastMousePosition.y;
                draggedFurniture.moveBy(dx, dy);
                lastMousePosition = e.getPoint();
                repaint();
            } else if (currentShape != null && !eraseMode) {
                Line2D line = (Line2D) currentShape;
                line.setLine(line.getX1(), line.getY1(), e.getX(), e.getY());
                repaint();
            }
        }
    });
}
```

This update enhances your `**mousePressed**` method to allow the eraser to remove not only furniture but also drawn lines. When in erase mode and a mouse press occurs, the code checks for proximity to both furniture and lines. If a line is detected near the mouse click (within the `**clickBounds**`), it is marked for removal. The code then proceeds to update the display accordingly by calling `**repaint()**`.

This approach relies on detecting intersections with a small bounding box around the click point, which should suffice for most uses but can be fine-tuned for sensitivity or specific requirements related to the proximity necessary to consider a line "clicked."

Anonymous

private void disableFurnitureInteraction() {

}

private void enableFurnitureInteraction() {

}

Can you rewrite these, the older implementation still allowed the user to drag furniture onto the drawing panel from the sidebar even when eraseMode was true

ChatGPT

To properly disable and enable furniture interaction in the context of your `DrawingPanel`, we need to focus on the actions allowed or disallowed during each state. In `eraseMode`, furniture items should not be draggable onto the panel. When not in `eraseMode`, the furniture items should be interactable again.

However, since the methods you're asking about (`disableFurnitureInteraction` and `enableFurnitureInteraction`) are empty, and there's no direct interaction logic provided within `DrawingPanel` that enables or disables dragging from the `Sidebar`, I assume you need a way to signal to the `Sidebar` or within `DrawingPanel` to ignore drag-and-drop events based on the `eraseMode` state.

I'll provide an implementation that conceptually toggles interaction with the furniture items based on a presumed structure. If you need to disable dragging from `Sidebar`, you should manage the draggable state of items based on the `eraseMode`:

In `DrawingPanel`, you could maintain a reference to the `Sidebar` and directly call methods on it to disable or enable item dragging:

java

```java
public class DrawingPanel extends JPanel {
    // Assuming you have a way to set the Sidebar reference
    private Sidebar sidebar;

    public void setSidebar(Sidebar sidebar) {
        this.sidebar = sidebar;
    }

    private void disableFurnitureInteraction() {
        if (sidebar != null) {
            sidebar.setDragEnabled(false);
        }
    }

    private void enableFurnitureInteraction() {
        if (sidebar != null) {
            sidebar.setDragEnabled(true);
        }
    }
}
```

Then, in your `Sidebar` class, you'd need to add logic to respond to these enable/disable requests. Here is a potential way to handle this by controlling the draggability of your furniture labels:

java

```java
public class Sidebar extends JPanel {
    private boolean dragEnabled = true;
    // Existing fields and constructor...

    public void setDragEnabled(boolean enabled) {
        dragEnabled = enabled;
        Component[] components = getComponents();
        for (Component component : components) {
            if (component instanceof JLabel) {
                JLabel label = (JLabel) component;
                MouseListener[] mouseListeners = label.getMouseListeners();
                for (MouseListener listener : mouseListeners) {
                    label.removeMouseListener(listener);
                }
                if (enabled) {
                    label.addMouseListener(createMouseListener());
```

```
                }
            }
        }
    }

    private MouseListener createMouseListener() {
        return new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                if (!dragEnabled) return;

                JComponent c = (JComponent) e.getSource();
                TransferHandler handler = c.getTransferHandler();
                handler.exportAsDrag(c, e, TransferHandler.COPY);
            }
        };
    }

    // Existing methods...
}
```

With these changes, when `eraseMode` is enabled, `DrawingPanel` will call `sidebar.setDragEnabled(false)`, which will loop through each `JLabel` in the sidebar, remove its mouse listeners, and thus prevent drag operations. When `eraseMode` is disabled, the mouse listeners are added back, allowing for furniture dragging to resume. Ensure that the original mouse listeners for dragging are set in a way that respects the current `dragEnabled` state.

Anonymous

private void disableFurnitureInteraction() {

}

private void enableFurnitureInteraction() {

}

These are in sidebar, can you maek them disable dragging and enable dragging

ChatGPT

To enable and disable furniture dragging within your `Sidebar` class, you need to control whether the mouse events initiate a drag action based on a flag that represents whether dragging is enabled. You can achieve this by adding a boolean field to `Sidebar` that tracks whether dragging is enabled and checking this field in your mouse event handler before starting a drag action.

Here's how you can modify the `Sidebar` class to include this functionality:

java

```
public class Sidebar extends JPanel {
    private FurnitureManager furnitureManager;
    private boolean isDraggingEnabled = true; // Flag to enable/disable dragging

    public Sidebar(FurnitureManager furnitureManager, PageSwapper pageSwapper) {
        this.furnitureManager = furnitureManager;
        setLayout(new GridLayout(0, 3, 10, 10));
```

```
        setBackground(Color.LIGHT_GRAY);
        setPreferredSize(new Dimension(200, 600));

        populateWithFurniture();
    }

    private void populateWithFurniture() {
        Border border = BorderFactory.createLineBorder(Color.BLACK, 1);
        FurnitureTransferHandler transferHandler = new FurnitureTransferHandler();

        for (Furniture furniture : furnitureManager.getFurnitureList()) {
            ImageIcon icon = new ImageIcon(furniture.getImagePath());
            Image scaledImage = icon.getImage().getScaledInstance(50, 50,
Image.SCALE_SMOOTH);
            JLabel furnitureLabel = new JLabel(new ImageIcon(scaledImage));
            furnitureLabel.setBorder(border);
            furnitureLabel.setOpaque(true);
            furnitureLabel.setBackground(Color.WHITE);
            furnitureLabel.setToolTipText(furniture.getName());
            furnitureLabel.setTransferHandler(transferHandler);

            furnitureLabel.addMouseListener(new MouseAdapter() {
                @Override
                public void mousePressed(MouseEvent e) {
                    if (isDraggingEnabled) {
                        JComponent c = (JComponent) e.getSource();
                        TransferHandler handler = c.getTransferHandler();
                        handler.exportAsDrag(c, e, TransferHandler.COPY);
                    }
                }
            });

            add(furnitureLabel);
        }
    }

    public void setEraseMode(boolean eraseMode) {
        // This method might toggle the erase mode and could be expanded based on your
implementation.
    }

    public void disableFurnitureInteraction() {
        isDraggingEnabled = false;
    }

    public void enableFurnitureInteraction() {
        isDraggingEnabled = true;
    }
}
```

In this implementation, `isDraggingEnabled` is checked in the `mousePressed` event handler. When `disableFurnitureInteraction` is called, it sets `isDraggingEnabled` to `false`, which prevents dragging. Conversely, `enableFurnitureInteraction` sets it back to `true`.

To integrate this functionality fully, ensure that you call `disableFurnitureInteraction` and `enableFurnitureInteraction` from the appropriate context in your application, possibly in response to UI actions that toggle the erase mode.

Anonymous

package com.ECS160.UI;

import javax.swing.*;
import javax.swing.border.Border;
import java.awt.*;

```
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import java.util.List;

public class Sidebar extends JPanel {
    private FurnitureManager furnitureManager;
    private PageSwapper pageSwapper;
    private List<JLabel> furnitureLabels;
    private MouseAdapter mouseAdapter;
    private boolean isDraggingEnabled = true;
    public Sidebar(FurnitureManager furnitureManager, PageSwapper pageSwapper) {
        this.furnitureManager = furnitureManager;
        setLayout(new GridLayout(0, 3, 10, 10));
        setBackground(Color.LIGHT_GRAY);
        setPreferredSize(new Dimension(200, 600));
        this.furnitureLabels = new ArrayList<>();
        this.pageSwapper = pageSwapper;

        populateWithFurniture();
    }

    private void populateWithFurniture() {
        Border border = BorderFactory.createLineBorder(Color.BLACK, 1);
        FurnitureTransferHandler transferHandler = new FurnitureTransferHandler();

        mouseAdapter = new MouseAdapter() {
            @Override
            public void mousePressed(MouseEvent e) {
                if (isDraggingEnabled) {
                    JComponent c = (JComponent) e.getSource();
                    TransferHandler handler = c.getTransferHandler();
                    handler.exportAsDrag(c, e, TransferHandler.COPY);
                }
            }
        };

        for (Furniture furniture : furnitureManager.getFurnitureList()) {
            ImageIcon icon = new ImageIcon(furniture.getImagePath());
            Image scaledImage = icon.getImage().getScaledInstance(50, 50, Image.SCALE_SMOOTH);
            JLabel furnitureLabel = new JLabel(new ImageIcon(scaledImage));
            furnitureLabel.setBorder(border);
            furnitureLabel.setOpaque(true);
            furnitureLabel.setBackground(Color.WHITE);
            furnitureLabel.setToolTipText(furniture.getName());
            furnitureLabel.setTransferHandler(transferHandler);
            furnitureLabel.addMouseListener(mouseAdapter);
            furnitureLabel.putClientProperty("furniture", furniture);
            furnitureLabels.add(furnitureLabel);
            add(furnitureLabel);
        }
```

```
    }

    public void setEraseMode(boolean eraseMode) {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        Cursor cursor;
        if (eraseMode) {
            Image cancelImage = toolkit.getImage("src/main/java/com/ECS160/UI/cancelIcon.png");
            cursor = toolkit.createCustomCursor(cancelImage, new Point(0, 0), "cancel");
            disableFurnitureInteraction();
            // Scale the image to a smaller size, e.g., 32x32 pixels
            Image scaledEraserImage = cancelImage.getScaledInstance(32, 32, Image.SCALE_SMOOTH);
            Cursor cancelCursor = toolkit.createCustomCursor(scaledEraserImage, new Point(0, 0),
"eraser");
            setCursor(cancelCursor);
        } else {
            cursor = Cursor.getDefaultCursor();
            enableFurnitureInteraction();
            setCursor(cursor);
        }

    }

    private void disableFurnitureInteraction() {

    }

    private void enableFurnitureInteraction() {

    }
}
```

for setEraseMode(true), can you make it so the entire sidebar is uninteractable until setEraseMode(false) is called

ChatGPT

To make the entire `Sidebar` non-interactive when `setEraseMode(true)` is called and restore interactivity with `setEraseMode(false)`, you can disable all components within the `Sidebar`. This approach will prevent any interaction with the `Sidebar` components until they are enabled again. Here's how you can modify the `setEraseMode` method, along with the helper methods `disableFurnitureInteraction` and `enableFurnitureInteraction`:

java

```java
public class Sidebar extends JPanel {
    // Existing attributes and constructor...

    public void setEraseMode(boolean eraseMode) {
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        if (eraseMode) {
            Image cancelImage =
toolkit.getImage("src/main/java/com/ECS160/UI/cancelIcon.png");
            Image scaledEraserImage = cancelImage.getScaledInstance(32, 32,
```

```
Image.SCALE_SMOOTH);
            Cursor cancelCursor = toolkit.createCustomCursor(scaledEraserImage, new Point(0,
0), "cancel");
            setCursor(cancelCursor);
            disableFurnitureInteraction();
        } else {
            setCursor(Cursor.getDefaultCursor());
            enableFurnitureInteraction();
        }
    }

    private void disableFurnitureInteraction() {
        for (JLabel label : furnitureLabels) {
            label.setEnabled(false); // Disable each furniture label
        }
        this.isDraggingEnabled = false; // Update the dragging flag
    }

    private void enableFurnitureInteraction() {
        for (JLabel label : furnitureLabels) {
            label.setEnabled(true); // Enable each furniture label
        }
        this.isDraggingEnabled = true; // Update the dragging flag
    }
}
```

In this updated implementation:

- When `setEraseMode(true)` is called, the sidebar switches the cursor to the "cancel" icon and calls `disableFurnitureInteraction`, which disables all furniture labels in the sidebar and updates the `isDraggingEnabled` flag to `false`.
- When `setEraseMode(false)` is called, it restores the default cursor and calls `enableFurnitureInteraction`, which re-enables all furniture labels and sets `isDraggingEnabled` back to `true`.

With these changes, the entire sidebar will become non-interactive in erase mode, preventing users from starting any drag-and-drop operations or interacting with the furniture labels. When erase mode is disabled, full interactivity is restored.

Anonymous

github gitignore line to remove any .DS_Stores

ChatGPT

To ignore `.DS_Store` files in your Git repository, you can add the following line to your `.gitignore` file:

```
.DS_Store
```

This line tells Git to ignore `.DS_Store` files in all directories of your project. If you've already committed a `.DS_Store` file before adding this rule, you'll need to remove it from the repository. You can do this with the following commands:

```
bash
```

```
# Remove any tracked .DS_Store files
find . -name .DS_Store -print0 | xargs -0 git rm -f --ignore-unmatch

# Commit the changes
git commit -m "Remove .DS_Store files"

# Add the .gitignore rule
echo .DS_Store >> .gitignore
git add .gitignore
git commit -m "Ignore .DS_Store files"
```

These commands will remove any tracked `.DS_Store` files from your repository and update your `.gitignore` to prevent them from being included in future commits.

Anonymous

how to git reset all .DS_Store files

ChatGPT

To reset all `.DS_Store` files in your Git repository (which essentially means to unstage them if they have been staged but not committed), you can use the `git clean` command for untracked files or `git reset` for staged changes. Below are the steps for each scenario:

1. **If `.DS_Store` files are untracked:**

If you haven't added the `.DS_Store` files to your staging area (they are untracked), and you want to remove them completely from your working directory, you can use `git clean`. Be cautious with this command, as it will remove the files from your working directory:

bash

```
# Dry run to see which files will be deleted
git clean -fdXn

# If the output is correct, proceed to delete the files
git clean -fdX
```

The `-fdX` flags tell Git to force delete, restrict the operation to directories, and only remove files ignored by Git.

2. **If `.DS_Store` files are staged:**

If the `.DS_Store` files have been staged (added to the index) but not committed, you need to reset them:

bash

```
# Reset all staged .DS_Store files
git reset HEAD -- '*.DS_Store'
```

This command will unstage any `.DS_Store` files across the repository, without affecting other staged changes.

3. **Ignore `.DS_Store` globally:**

Since `.DS_Store` files are a common nuisance for macOS users, you might want to ignore them globally across all your Git repositories:

```bash
# Ignore .DS_Store files in all your Git repositories
echo .DS_Store >> ~/.gitignore_global
git config --global core.excludesfile ~/.gitignore_global
```

This way, you won't need to add `.DS_Store` to every `.gitignore` file in each of your repositories.

Anonymous

git clean -fdX

How do i undo this?

ChatGPT

Unfortunately, `git clean -fdX` is an irreversible operation. This command permanently removes untracked files from your working directory, and there is no direct way to undo this action or recover the files through Git.

Here are a few steps you can consider to potentially recover your files, depending on your situation:

1. **Check for Local Backups:**
    - If you have a habit of making regular backups of your system or your work directory, check your backups for the most recent copy of the files.
2. **Recovery Software:**
    - If the deleted files are crucial and you don't have any backups, you might consider using file recovery software to attempt to recover the deleted files. It's important to act quickly because the more you use your system after the deletion, the more likely the space used by the deleted files will be overwritten.
    - Stop using the affected disk immediately to avoid overwriting the deleted files.
    - Use a different computer or drive to download and install file recovery software.
    - Run the recovery software from a different drive than the one where your files were deleted and follow the software's instructions to scan for and recover deleted files.
3. **Re-clone or Restore from Remote:**
    - If the deleted files were part of a cloned repository and you haven't made any local-only changes that you need to keep, you could re-clone the repository from the remote source.
4. **Recreate the Files:**

    - If the deleted files can be recreated (e.g., auto-generated files or files that can be re-exported from other sources), you may proceed to regenerate them.

To prevent such irreversible loss in the future, consider the following practices:

- Always double-check the files listed by `**git clean -n**` (the dry run mode) before running `**git clean**` with force.
- Regularly back up important files.
- Be cautious with Git clean-up commands and understand their implications before use.

If you frequently find untracked files that should not be deleted, make sure to update your `**.gitignore**` files accordingly to prevent accidental deletion in the future.