

User

Hi ChatGPT, I'm continuing a discussion I've had with you, I asked you to summarize things for me to continue in a different chat and I'll put it right below this sentence:

Certainly! In our discussion, we explored the architecture for managing furniture objects within a canvas-based application. We outlined a design where each furniture object class (e.g., Desk, Bed) encapsulates its own properties such as position, size, and rotation. These properties are managed internally by each object, allowing for independent state management. Additionally, we discussed the concept of a separate Java file or class responsible for maintaining the overall scene on the canvas. This "scene management" class acts as a centralized repository for storing and manipulating furniture objects, along with any other elements present in the scene. It maintains a list of objects currently on the canvas, assigns unique IDs to each object, and tracks their positions and other attributes. Furthermore, we touched upon the role of this class in implementing features like undo/redo functionality by maintaining a history of user actions and managing command objects encapsulating these actions. By adopting this architecture, the application achieves modularity, separation of concerns, and maintainability, facilitating the implementation of various features and future enhancements.

Please ask any questions if you're still have any concern, since I'll be asking for your cooperation in a bit!

ChatGPT

Thank you for providing the summary. It seems like we've discussed a comprehensive architecture for managing furniture objects within a canvas-based application, emphasizing modularity, separation of concerns, and maintainability. What specific aspects or details would you like to delve into further in our continued discussion?

User

So first, I want to test if I can keep track of every object in the canvas as its own object (encapsulating object). To do that, I need another file that basically maintains/keeps track of the object in the canvas. I would like to name it ManageCanvas, but if there's a better name please rename it. Would you like to see some of the code that I have, such as the MainCanvasPanel (this is the main canvas)? or the DrawingTools (file contain to draw stuff on the canvas). For future testing, I want to try deleting an object if possible for now, I won't code features such as undo, move, and resize for now.

ChatGPT

Sure, I'd be happy to assist you with that. Let's start by creating the ManageCanvas (or a better-suited name if you prefer) class that will keep track of objects on the canvas. We can discuss the structure and responsibilities of this class before delving into specific code implementation.

Could you provide a brief overview of how objects are currently represented on the canvas and how you envision the ManageCanvas class interacting with them? This will help in designing the class effectively. Additionally, if you have any existing code related to the canvas or drawing

tools that you think would be relevant to understanding the context, feel free to share it. We can then proceed to discuss how to implement the functionality for deleting objects and any other features you'd like to explore.

User

So basically, I have a mainCanvasPanel, which is basically the first setup of the canvas. It will draw a grid-style canvas, and have zoom features (the zoom features will still clear the canvas but we will work on that for future reference). We also have a DrawingTools.java, that handles two main objects (StructureObject, and FurnitureObjects). The StructureObjects contains walls, doors, and windows, where users would click start and end points, to draw what they pick. On the furnitureObjects, we would pick furniture from a button, and then click anywhere on the canvas, and it will put down an object (what we click is going to be the center of the object).

How I would imagine ManageCanvas working is that it will keep managing objects (maybe in arrays) for StructureObjects, and FurnitureObjects. It will also keep down a history of what object (maybe a copy of the objects).

Is that enough information for you, or do you need more? Please let me know, or recap what you know so I can correct you if there's anything wrong or missing :)

ChatGPT

Thank you for providing the overview. Based on the information you've provided, it seems like the ManageCanvas class will serve as a centralized manager for handling objects on the canvas. Here's a recap of the key responsibilities and functionalities we discussed:

Object Management: The ManageCanvas class will maintain arrays or collections of StructureObject and FurnitureObject instances representing walls, doors, windows, and furniture placed on the canvas. These arrays will allow for easy access to and manipulation of these objects.

History Management: The ManageCanvas class will also be responsible for maintaining a history of actions performed on the canvas. This could involve keeping track of changes made to objects, such as creation, deletion, or movement. Storing a history of these actions will enable features like undo/redo functionality in the future.

Interactions with Drawing Tools: The ManageCanvas class may interact with the DrawingTools class to add newly created objects to the canvas and update the object arrays accordingly. It may also receive notifications from the drawing tools when objects are deleted or modified.

Encapsulation and Modularity: The ManageCanvas class will encapsulate the logic for managing objects on the canvas, ensuring modularity and separation of concerns within the application architecture.

With this understanding, we can proceed to design and implement the ManageCanvas class. We can start by defining its structure, including instance variables, methods for adding, deleting,

and updating objects, as well as managing the history of actions. Once we have a clear plan, we can begin coding and testing the functionality, starting with the ability to delete objects from the canvas. Let me know if you're ready to move forward, or if you have any questions or additional specifications.

User

Something that I never said, I'm not sure if you're assuming this, but the walls, window, and door extends from StructureObjects, while all the furniture extends from FurnitureObjects.

Otherwise, I think you got the basis down, for now, can we have it so that ManageCanvas stores structures and furniture. As long as it can store the objects, and I can still place each object on the canvas like normal, we can proceed. Let me know if you need any of the basic code from MainCanvasPanel or DrawingTools for you to start!

ChatGPT

Thank you for clarifying the inheritance structure of the objects. That helps in understanding the hierarchy better.

Let's start by designing the ManageCanvas class to store both StructureObject and FurnitureObject instances. We'll define methods for adding, retrieving, and deleting objects from the canvas. Here's a basic outline of what the class might look like:

java

Copy code

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

public class ManageCanvas {
    private List<StructureObject> structures;
    private List<FurnitureObject> furniture;

    public ManageCanvas() {
        structures = new ArrayList<>();
        furniture = new ArrayList<>();
    }

    // Method to add a StructureObject to the canvas
    public void addStructure(StructureObject structure) {
        structures.add(structure);
    }

    // Method to add a FurnitureObject to the canvas
    public void addFurniture(FurnitureObject piece) {
        furniture.add(piece);
    }
}
```

```

// Method to delete a StructureObject from the canvas
public void deleteStructure(StructureObject structure) {
    structures.remove(structure);
}

// Method to delete a FurnitureObject from the canvas
public void deleteFurniture(FurnitureObject piece) {
    furniture.remove(piece);
}

// Method to get all StructureObjects on the canvas
public List<StructureObject> getAllStructures() {
    return new ArrayList<>(structures);
}

// Method to get all FurnitureObjects on the canvas
public List<FurnitureObject> getAllFurniture() {
    return new ArrayList<>(furniture);
}
}

```

This class provides basic functionality for managing objects on the canvas. You can add new structures and furniture, delete existing ones, and retrieve lists of all structures and furniture currently on the canvas.

Before proceeding further, could you please provide any relevant code from the StructureObject and FurnitureObject classes, as well as any methods from the DrawingTools class that are involved in placing objects on the canvas? This will help ensure that the integration between ManageCanvas and the rest of your application works smoothly. Once we have this information, we can proceed with testing and refining the functionality.

User

Okay, I will be providing the code for StructureObject, FurnitureObject, and DrawingTools right below:

```

import java.awt.Point;
import java.awt.Graphics2D;

// @author ChatGPT
public abstract class StructureObject {
    protected Point startPoint;
    protected Point endPoint;

    public StructureObject(Point startPoint, Point endPoint) {

```

```

        this.startPoint = startPoint;
        this.endPoint = endPoint;
    }

    public Point getStartPoint() {
        return startPoint;
    }

    public void setStartPoint(Point startPoint) {
        this.startPoint = startPoint;
    }

    public Point getEndPoint() {
        return endPoint;
    }

    public void setEndPoint(Point endPoint) {
        this.endPoint = endPoint;
    }

    // Abstract method for drawing the object
    public abstract void draw(Graphics2D g2d);
}

-----
import java.awt.*;

public abstract class FurnitureObject {
    protected Point startPoint;
    protected int width;
    protected int height;

    public FurnitureObject(Point startPoint, int width, int height) {
        this.startPoint = startPoint;
        this.width = width;
        this.height = height;
    }

    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }
}

```

```

    public Point getStartPoint() {
        return startPoint;
    }

    public abstract void draw(Graphics2D g2d);
}

-----

import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.*;

public class DrawingTools {
    private JPanel canvas;
    private DrawingMode drawingMode = DrawingMode.NONE;
    private Point tempPoint1;
    private Point tempPoint2;

    public DrawingTools(JPanel canvas) {
        this.canvas = canvas;
        setupDrawing();
    }

    private void setupDrawing() {
        canvas.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                if (drawingMode == DrawingMode.NONE) {
                    return;
                }

                if (isStructure()) {
                    if (tempPoint1 == null) {
                        tempPoint1 = e.getPoint();
                    } else {
                        tempPoint2 = e.getPoint();
                        drawStructuralObject(createStructuralObject(tempPoint1, tempPoint2));
                        tempPoint1 = null;
                    }
                } else {
                    drawFurnitureObject(createFurnitureObject(e.getPoint(), 50, 100));
                }
            }
        });
    }
}

```

```

    });
}

public void setDrawingMode(DrawingMode mode) {
    this.drawingMode = mode;
}

private boolean isStructure() {
    return drawingMode == DrawingMode.DOOR || drawingMode == DrawingMode.WALL ||
drawingMode == DrawingMode.WINDOW;
}

private void drawStructuralObject(StructureObject object) {
    Graphics2D g2d = (Graphics2D) canvas.getGraphics();
    object.draw(g2d);
    g2d.dispose();
}

private void drawFurnitureObject(FurnitureObject furnitureObject) {
    Graphics2D g2d = (Graphics2D) canvas.getGraphics();
    furnitureObject.draw(g2d);
    g2d.dispose();
}

private StructureObject createStructuralObject(Point startPoint, Point endPoint) {
    switch (drawingMode) {
        case WALL:
            return new Wall(startPoint, endPoint);
        case WINDOW:
            return new Window(startPoint, endPoint);
        case DOOR:
            return new Door(startPoint, endPoint);
        default:
            throw new IllegalArgumentException("Unknown drawing mode: " + drawingMode);
    }
}

private FurnitureObject createFurnitureObject(Point point, int width, int height) {
    switch (drawingMode) {
        case BED:
            return new Bed(point, 100, 150);
        case DESK:
            return new Desk(point, width, height);
        case CHAIR:

```

```

        return new Chair(point, 25, 25);
    case TOILET:
        return new Toilet(point, 25, 25);
    case SHOWER:
        return new Shower(point, 40, 80);
    case SINK:
        return new Sink(point, 60, 45);
    default:
        throw new IllegalArgumentException("Unknown drawing mode: " + drawingMode);
    }
}

```

```

public enum DrawingMode {
    NONE,
    WALL,
    WINDOW,
    BED,
    DOOR,
    DESK,
    CHAIR,
    TOILET,
    SHOWER,
    SINK,
    // Add more drawing modes as needed
}

```

ChatGPT

Thank you for providing the code for StructureObject, FurnitureObject, and DrawingTools. Based on this code, we can proceed to integrate the ManageCanvas class into the application.

Here's how we can modify the DrawingTools class to utilize the ManageCanvas class for managing objects on the canvas:

```

java
Copy code
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.*;

public class DrawingTools {
    private JPanel canvas;
    private DrawingMode drawingMode = DrawingMode.NONE;
    private Point tempPoint1;

```



```

private Point tempPoint2;
private ManageCanvas manageCanvas;

public DrawingTools(JPanel canvas, ManageCanvas manageCanvas) {
    this.canvas = canvas;
    this.manageCanvas = manageCanvas;
    setupDrawing();
}

private void setupDrawing() {
    canvas.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            if (drawingMode == DrawingMode.NONE) {
                return;
            }

            if (isStructure()) {
                if (tempPoint1 == null) {
                    tempPoint1 = e.getPoint();
                } else {
                    tempPoint2 = e.getPoint();
                    StructureObject object = createStructuralObject(tempPoint1, tempPoint2);
                    drawStructuralObject(object);
                    manageCanvas.addStructure(object); // Add structure to ManageCanvas
                    tempPoint1 = null;
                }
            } else {
                FurnitureObject furnitureObject = createFurnitureObject(e.getPoint(), 50, 100);
                drawFurnitureObject(furnitureObject);
                manageCanvas.addFurniture(furnitureObject); // Add furniture to ManageCanvas
            }
        }
    });
}

// Other methods remain unchanged...

private void drawStructuralObject(StructureObject object) {
    Graphics2D g2d = (Graphics2D) canvas.getGraphics();
    object.draw(g2d);
    g2d.dispose();
}

```

```

private void drawFurnitureObject(FurnitureObject furnitureObject) {
    Graphics2D g2d = (Graphics2D) canvas.getGraphics();
    furnitureObject.draw(g2d);
    g2d.dispose();
}

// Other methods remain unchanged...
}

```

In this modified version, when a new structure or furniture object is created on the canvas, it's added to the ManageCanvas instance using the addStructure() or addFurniture() methods respectively.

Now, you should be able to use the ManageCanvas class to manage objects on the canvas. You can add, retrieve, and delete objects as needed. Let me know if you need further clarification or assistance with any part of the implementation.

User

Okay, this sounds okay, but I have another thing that I forgot to mention, I have a file that calls the DrawingTools since it contains all my buttons that give what MODE the user is drawing. I will give the code below, do I just change the parameters when calling by giving it a manageCanvas?

```

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

public class FeatureBarPanel extends JPanel {
    public FeatureBarPanel(JPanel canvas) {
        setLayout(new BorderLayout());
        setPreferredSize(new Dimension(200, 600));

        // Drawing tools setup
        DrawingTools drawingTools = new DrawingTools(canvas);

        // Feature menu panel with borders
        JPanel featureMenuPanel = new JPanel(new BorderLayout());
        featureMenuPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

        // Tools panel with icons
        JPanel toolsPanel = new JPanel(new GridLayout(2, 2));
        addButtonWithIcon(toolsPanel, "Mouse", "cursor.png", 80, 80); // Set preferred size for
buttons
        addButtonWithIcon(toolsPanel, "Rotate-Left", "rotate-left.png", 80, 80);
        addButtonWithIcon(toolsPanel, "Rotate-Right", "rotate-right.png", 80, 80);
        addButtonWithIcon(toolsPanel, "Rotate-Flip", "rotate-180.png", 80, 80);
    }
}

```

```

featureMenuPanel.add(toolsPanel, BorderLayout.NORTH);

// Search bar panel
JPanel searchBarPanel = new JPanel(new BorderLayout());
searchBarPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

// Title for search bar
JLabel searchTitleLabel = new JLabel("Search:");
searchBarPanel.add(searchTitleLabel, BorderLayout.WEST);

// Search text field with placeholder text
JTextField searchBar = new JTextField();
searchBar.setPreferredSize(new Dimension(150, 25));
searchBar.setMaximumSize(new Dimension(150, 25)); // Set maximum size for search bar
searchBar.setText("Search furniture...");
searchBarPanel.add(searchBar, BorderLayout.CENTER);

featureMenuPanel.add(searchBarPanel, BorderLayout.SOUTH);

// Feature categories tabbed pane
JTabbedPane tabbedPane = new JTabbedPane();
tabbedPane.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

// Add tabs
tabbedPane.addTab("Dining Room", PanelCreation.createPanel("Dining Room Features",
drawingTools));
tabbedPane.addTab("Living Room", PanelCreation.createPanel("Living Room Features",
drawingTools));
tabbedPane.addTab("Bathroom", PanelCreation.createPanel("Bathroom Features",
drawingTools));
tabbedPane.addTab("Kitchen", PanelCreation.createPanel("Kitchen Features",
drawingTools));
tabbedPane.addTab("Bedroom", PanelCreation.createPanel("Bedroom Features",
drawingTools));
tabbedPane.addTab("Structural", PanelCreation.createPanel("Structural Features",
drawingTools));

featureMenuPanel.add(tabbedPane, BorderLayout.CENTER);

add(featureMenuPanel, BorderLayout.CENTER);
}

private static void addButtonWithIcon(Container container, String buttonText, String
iconFileName, int width, int height) {

```

```

        String iconDirectory = "../assets"; // Make it static
        String iconPath = iconDirectory + "/" + iconFileName; // Concatenate directory path and file
name
        ImageIcon icon = new ImageIcon("../assets/cursor.png");
        JButton button = new JButton(buttonText, icon);
        button.setPreferredSize(new Dimension(width, height)); // Set preferred size for button
        container.add(button); // Add button to the specified container
    }
}

```

ChatGPT

To integrate the ManageCanvas class into the FeatureBarPanel class, you need to pass the ManageCanvas instance to the DrawingTools constructor when creating it. Here's how you can modify the FeatureBarPanel class to achieve this:

java

Copy code

```

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

```

```

public class FeatureBarPanel extends JPanel {
    public FeatureBarPanel(JPanel canvas, ManageCanvas manageCanvas) {
        setLayout(new BorderLayout());
        setPreferredSize(new Dimension(200, 600));

        // Drawing tools setup with ManageCanvas instance
        DrawingTools drawingTools = new DrawingTools(canvas, manageCanvas);

        // Other parts of the FeatureBarPanel remain unchanged...
    }

    // Other methods remain unchanged...
}

```

By passing the manageCanvas instance to the DrawingTools constructor, you ensure that the DrawingTools class has access to the ManageCanvas instance and can interact with it accordingly.

With this modification, the FeatureBarPanel class is now equipped to work with the ManageCanvas class. Let me know if you need further assistance or clarification!

User

This means I have to change the App.java, since it calls the featureBarPanel?

```

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

```

```

public class App extends JFrame {
    public App() {
        super("Simple Floor Planning Application");
        initUI();
    }

    private void initUI() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        MainCanvasPanel mainCanvasPanel = new MainCanvasPanel();
        FeatureBarPanel featureBarPanel = new FeatureBarPanel(mainCanvasPanel);
        MenuBar menuBar = new MenuBar(mainCanvasPanel);

        add(mainCanvasPanel, BorderLayout.CENTER);
        add(featureBarPanel, BorderLayout.WEST);
        setJMenuBar(menuBar);

        pack();
        setLocationRelativeTo(null);
    }

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

```

ChatGPT

Yes, you're correct. Since App.java is responsible for initializing and setting up the UI components, you'll need to modify it to pass the ManageCanvas instance to the FeatureBarPanel class.

Here's how you can modify App.java to create and pass a ManageCanvas instance to the FeatureBarPanel class:

```

java
Copy code
import javax.swing.*;
import java.awt.BorderLayout;

```

```

public class App extends JFrame {
    public App() {
        super("Simple Floor Planning Application");
        initUI();
    }

    private void initUI() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        MainCanvasPanel mainCanvasPanel = new MainCanvasPanel();
        ManageCanvas manageCanvas = new ManageCanvas(); // Create a ManageCanvas
instance
        FeatureBarPanel featureBarPanel = new FeatureBarPanel(mainCanvasPanel,
manageCanvas); // Pass ManageCanvas instance

        MenuBar menuBar = new MenuBar(mainCanvasPanel);

        add(mainCanvasPanel, BorderLayout.CENTER);
        add(featureBarPanel, BorderLayout.WEST);
        setJMenuBar(menuBar);

        pack();
        setLocationRelativeTo(null);
    }

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

```

With this modification, the App class now creates a ManageCanvas instance and passes it to the FeatureBarPanel class when initializing it. This ensures that the FeatureBarPanel class has access to the ManageCanvas instance and can interact with it accordingly.

Let me know if you need further clarification or assistance!

User

Okay, is it possible to print all the arrays in the terminal everytime i add a structure?, maybe like how many are on the list currently?

ChatGPT

Yes, you can add a method in the ManageCanvas class to print the current state of the arrays whenever a structure is added. Here's how you can modify the ManageCanvas class to achieve this:

java

Copy code

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

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

```
public class ManageCanvas {  
    private List<StructureObject> structures;  
    private List<FurnitureObject> furniture;
```

```
    public ManageCanvas() {  
        structures = new ArrayList<>();  
        furniture = new ArrayList<>();  
    }
```

```
    public void addStructure(StructureObject structure) {  
        structures.add(structure);  
        printStructureList(); // Print structure list after adding a structure  
    }
```

```
    public void addFurniture(FurnitureObject piece) {  
        furniture.add(piece);  
        printFurnitureList(); // Print furniture list after adding a furniture piece  
    }
```

```
    // Other methods remain unchanged...
```

```
    private void printStructureList() {  
        System.out.println("Structures on canvas:");  
        for (int i = 0; i < structures.size(); i++) {  
            System.out.println((i + 1) + ". " + structures.get(i).getClass().getSimpleName());  
        }  
    }
```

```
    private void printFurnitureList() {  
        System.out.println("Furniture on canvas:");  
        for (int i = 0; i < furniture.size(); i++) {  
            System.out.println((i + 1) + ". " + furniture.get(i).getClass().getSimpleName());  
        }  
    }  
}
```

In this modified version of the ManageCanvas class, I've added two private methods: `printStructureList()` and `printFurnitureList()`. These methods iterate over the structures and furniture lists respectively and print the class names of the objects currently stored in them.

Additionally, I've modified the `addStructure()` and `addFurniture()` methods to call the corresponding print methods after adding a new object to the lists.

Now, whenever a structure or furniture is added to the canvas, the current state of the arrays will be printed in the terminal. You can customize the print format as needed. Let me know if you need further assistance!

User

Im not familiar with .java, what does `getSimpleName()` means?

ChatGPT

`getSimpleName()` is a method provided by the Java Reflection API, specifically for the `Class` class. It returns the simple name of the underlying class as a `String`. The simple name of a class is the name of the class without the package name.

For example, if you have a class `MyClass` in package `com.example`, calling `MyClass.class.getSimpleName()` will return `"MyClass"`.

In the context of the `ManageCanvas` class, `getClass().getSimpleName()` is used to retrieve the simple name of the class of the objects stored in the structures and furniture lists. This allows you to print out a human-readable representation of the class names when printing the contents of the lists.

User

I see okay then, its works wonderfully, I think I'm going to start working on some buttons that will help me manipulate some of the objects.

2 / 2

ChatGPT

That sounds like a great next step! Implementing buttons to manipulate the objects on the canvas will enhance the functionality of your application. Here are some common operations you might consider implementing buttons for:

Delete: Allow users to delete selected objects from the canvas.

Move: Enable users to move selected objects to a different position on the canvas.

Resize: Allow users to resize selected objects, if applicable (e.g., furniture objects).

Rotate: Implement functionality to rotate selected objects.

Undo/Redo: Provide buttons to undo or redo actions performed on the canvas.

Save/Load: Implement buttons to save the current state of the canvas or load previously saved states.

You can create buttons for these operations and attach action listeners to handle the corresponding functionality. These action listeners would interact with the ManageCanvas class to perform the desired operations on the objects.

If you need assistance with implementing any specific operation or have questions about how to approach certain aspects of the implementation, feel free to ask! I'm here to help.

2 / 2

User

Okay, give me a moment to add more buttons that corresponds to some of these features. I also need to debug some code since I can't seem to use Imagemcon, and may opt for an alternative ChatGPT

Take your time! If you encounter any issues with using Imagemcon or need assistance with debugging, feel free to share the relevant code or error messages, and I'll do my best to help you resolve them. If you decide to explore alternatives to Imagemcon, I can also provide suggestions or guidance on alternative approaches for displaying images in your application.

User

Okay, I'm still having some trouble with the Imagemcon, but I think that's a small issue. Right now, I want to focus on deleting an object. What method do you think is efficient for me to handle this. When I click a delete button, would it activate a flag? and then if I click any other button, it would just change the flag. Because right now, the only flag that I have is for Structures and Furnitures in drawingTools using enum.

ChatGPT

Handling object deletion in a canvas-based application like yours can be done in several ways. One common approach is to implement a selection mechanism where users can click on an object to select it, and then perform actions such as deletion on the selected object.

Here's a suggested approach:

Selection Mechanism: Implement a mechanism to allow users to select objects on the canvas. This could involve clicking on an object to select it or using a selection tool to drag a selection box around multiple objects.

Delete Button: Add a "Delete" button to your UI. When the user clicks this button, it activates a delete mode.

Deletion Flag: You can indeed use a flag to indicate whether the application is in delete mode. This flag can be a boolean variable or an enum value representing the current mode of the application.

Object Deletion: When in delete mode, clicking on an object would trigger its deletion from the canvas. This involves removing the object from the ManageCanvas instance and updating the UI accordingly.

Exiting Delete Mode: Clicking on any other button or performing any other action would exit the delete mode, allowing users to interact with the canvas normally again.

Here's a basic outline of how you can implement this:

java

Copy code

// In DrawingTools.java

```
public class DrawingTools {
    private JPanel canvas;
    private DrawingMode drawingMode = DrawingMode.NONE;
    private boolean deleteMode = false; // Flag for delete mode
    // Other variables and methods...

    public void setDeleteMode(boolean deleteMode) {
        this.deleteMode = deleteMode;
    }

    // Other methods...

    private void deleteObject(Point point) {
        if (deleteMode) {
            // Iterate over objects in canvas and check if point is inside any object
            // If found, remove the object from ManageCanvas and redraw canvas
        }
    }
}
```

java

Copy code

// In FeatureBarPanel.java

```
public class FeatureBarPanel extends JPanel {
    public FeatureBarPanel(JPanel canvas, ManageCanvas manageCanvas, DrawingTools
drawingTools) {
        // Initialize UI components including the "Delete" button
        JButton deleteButton = new JButton("Delete");
        deleteButton.addActionListener(e -> {
            drawingTools.setDeleteMode(true); // Activate delete mode
            drawingTools.setDrawingMode(DrawingMode.NONE); // Deactivate drawing mode
        });
    }
}
```

```

    });
    // Other UI components...
}
}

```

This approach allows users to enter delete mode by clicking the "Delete" button. While in delete mode, clicking on objects on the canvas triggers their deletion. Exiting delete mode can be achieved by clicking any other button or performing other actions.

Let me know if you need further clarification or assistance with implementing this approach!

User

I think I would have some troubles with this since FeatureBarPanel create each of my tool buttons inside another function, please take a look and modify with your discretion. Please don't erase any code that are significant to the program. I would be checking over your code so please go ahead

```

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

public class FeatureBarPanel extends JPanel {
    private static final int button_width = 80;
    private static final int button_height = 40;

    public FeatureBarPanel(JPanel canvas, ManageCanvas manageCanvas) {
        setLayout(new BorderLayout());
        setPreferredSize(new Dimension(200, 600));

        // Drawing tools setup
        DrawingTools drawingTools = new DrawingTools(canvas, manageCanvas);

        // Feature menu panel with borders
        JPanel featureMenuPanel = new JPanel(new BorderLayout());
        featureMenuPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

        // Tools panel with icons
        JPanel toolsPanel = new JPanel(new GridLayout(3, 2));
        addButtonWithIcon(toolsPanel, "Mouse", "cursor.png", button_width, button_height); // Set
        preferred size for buttons
        addButtonWithIcon(toolsPanel, "Delete", "none.png", button_width, button_height);
        addButtonWithIcon(toolsPanel, "Resize", "none.png", button_width, button_height);
        addButtonWithIcon(toolsPanel, "Rotate-Left", "rotate-left.png", button_width,
        button_height);
        addButtonWithIcon(toolsPanel, "Rotate-Right", "rotate-right.png", button_width,
        button_height);
    }
}

```

```

        addButtonWithIcon(toolsPanel, "Rotate-Flip", "rotate-180.png", button_width,
button_height);
        featureMenuPanel.add(toolsPanel, BorderLayout.NORTH);

// Search bar panel
JPanel searchBarPanel = new JPanel(new BorderLayout());
searchBarPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

// Title for search bar
JLabel searchTitleLabel = new JLabel("Search:");
searchBarPanel.add(searchTitleLabel, BorderLayout.WEST);

// Search text field with placeholder text
JTextField searchBar = new JTextField();
searchBar.setPreferredSize(new Dimension(150, 25));
searchBar.setMaximumSize(new Dimension(150, 25)); // Set maximum size for search bar
searchBar.setText("Search furniture...");
searchBarPanel.add(searchBar, BorderLayout.CENTER);

featureMenuPanel.add(searchBarPanel, BorderLayout.SOUTH);

// Feature categories tabbed pane
JTabbedPane tabbedPane = new JTabbedPane();
tabbedPane.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

// Add tabs
tabbedPane.addTab("Dining Room", PanelCreation.createPanel("Dining Room Features",
drawingTools));
tabbedPane.addTab("Living Room", PanelCreation.createPanel("Living Room Features",
drawingTools));
tabbedPane.addTab("Bathroom", PanelCreation.createPanel("Bathroom Features",
drawingTools));
tabbedPane.addTab("Kitchen", PanelCreation.createPanel("Kitchen Features",
drawingTools));
tabbedPane.addTab("Bedroom", PanelCreation.createPanel("Bedroom Features",
drawingTools));
tabbedPane.addTab("Structural", PanelCreation.createPanel("Structural Features",
drawingTools));

featureMenuPanel.add(tabbedPane, BorderLayout.CENTER);

add(featureMenuPanel, BorderLayout.CENTER);
}

```

```

        private static void addButtonWithIcon(Container container, String buttonText, String
iconFileName, int width, int height) {
            String iconDirectory = "../assets"; // Make it static
            String iconPath = iconDirectory + "/" + iconFileName; // Concatenate directory path and file
name
            ImageIcon icon = new ImageIcon("../assets/cursor.png");
            JButton button = new JButton(buttonText, icon);
            button.setPreferredSize(new Dimension(width, height)); // Set preferred size for button
            container.add(button); // Add button to the specified container
        }
    }
}

```

ChatGPT

Certainly! I'll modify the FeatureBarPanel class to incorporate the delete functionality without altering any significant parts of your existing code. Here's the modified version:

```

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

public class FeatureBarPanel extends JPanel {
    private static final int button_width = 80;
    private static final int button_height = 40;

    public FeatureBarPanel(JPanel canvas, ManageCanvas manageCanvas) {
        setLayout(new BorderLayout());
        setPreferredSize(new Dimension(200, 600));

        // Drawing tools setup
        DrawingTools drawingTools = new DrawingTools(canvas, manageCanvas);

        // Feature menu panel with borders
        JPanel featureMenuPanel = new JPanel(new BorderLayout());
        featureMenuPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

        // Tools panel with icons
        JPanel toolsPanel = new JPanel(new GridLayout(3, 2));
        addButtonWithIcon(toolsPanel, "Mouse", "cursor.png", button_width, button_height,
drawingTools, DrawingTools.DrawingMode.NONE); // Set preferred size for buttons
        addButtonWithIcon(toolsPanel, "Delete", "delete.png", button_width, button_height,
drawingTools, DrawingTools.DrawingMode.DELETE);
        addButtonWithIcon(toolsPanel, "Resize", "resize.png", button_width, button_height,
drawingTools, DrawingTools.DrawingMode.RESIZE);
    }
}

```

```

        addButtonWithIcon(toolsPanel, "Rotate-Left", "rotate-left.png", button_width, button_height,
drawingTools, DrawingTools.DrawingMode.ROTATE_LEFT);
        addButtonWithIcon(toolsPanel, "Rotate-Right", "rotate-right.png", button_width,
button_height, drawingTools, DrawingTools.DrawingMode.ROTATE_RIGHT);
        addButtonWithIcon(toolsPanel, "Rotate-Flip", "rotate-180.png", button_width,
button_height, drawingTools, DrawingTools.DrawingMode.ROTATE_FLIP);
        featureMenuPanel.add(toolsPanel, BorderLayout.NORTH);

// Search bar panel
JPanel searchBarPanel = new JPanel(new BorderLayout());
searchBarPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

// Title for search bar
JLabel searchTitleLabel = new JLabel("Search:");
searchBarPanel.add(searchTitleLabel, BorderLayout.WEST);

// Search text field with placeholder text
JTextField searchBar = new JTextField();
searchBar.setPreferredSize(new Dimension(150, 25));
searchBar.setMaximumSize(new Dimension(150, 25)); // Set maximum size for search bar
searchBar.setText("Search furniture...");
searchBarPanel.add(searchBar, BorderLayout.CENTER);

featureMenuPanel.add(searchBarPanel, BorderLayout.SOUTH);

// Feature categories tabbed pane
JTabbedPane tabbedPane = new JTabbedPane();
tabbedPane.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

// Add tabs
tabbedPane.addTab("Dining Room", PanelCreation.createPanel("Dining Room Features",
drawingTools));
tabbedPane.addTab("Living Room", PanelCreation.createPanel("Living Room Features",
drawingTools));
tabbedPane.addTab("Bathroom", PanelCreation.createPanel("Bathroom Features",
drawingTools));
tabbedPane.addTab("Kitchen", PanelCreation.createPanel("Kitchen Features",
drawingTools));
tabbedPane.addTab("Bedroom", PanelCreation.createPanel("Bedroom Features",
drawingTools));
tabbedPane.addTab("Structural", PanelCreation.createPanel("Structural Features",
drawingTools));

featureMenuPanel.add(tabbedPane, BorderLayout.CENTER);

```

```

        add(featureMenuPanel, BorderLayout.CENTER);
    }

    private void addButtonWithIcon(Container container, String buttonText, String iconFileName,
int width, int height, DrawingTools drawingTools, DrawingTools.DrawingMode mode) {
        String iconDirectory = "../assets"; // Make it static
        String iconPath = iconDirectory + "/" + iconFileName; // Concatenate directory path and file
name
        // Image icon - Example: ImageIcon icon = new ImageIcon(iconPath);
        JButton button = new JButton(buttonText); // Create button without icon for now
        button.setPreferredSize(new Dimension(width, height)); // Set preferred size for button
        button.addActionListener(e -> drawingTools.setDrawingMode(mode)); // Set drawing mode
when button is clicked
        container.add(button); // Add button to the specified container
    }
}

```

Changes made:

Added a "Delete" button to the tools panel with an appropriate icon (you can replace "delete.png" with the actual icon file name).

Modified the addButtonWithIcon() method to accept an instance of DrawingTools and a DrawingTools.DrawingMode parameter. This allows setting the drawing mode when the button is clicked.

Modified the addButtonWithIcon() calls to pass the drawingTools instance and the corresponding drawing mode enum value for each button.

Removed the ImageIcon creation for now since you mentioned having trouble with it. You can add it back once the issue is resolved.

User

theres a lot of parameter in this one function, could we perhaps separate it so it wouldn't be a mess?

ChatGPT

Absolutely! We can refactor the addButtonWithIcon() method to make it cleaner and more readable. One way to do this is by breaking it down into smaller, more focused methods. Here's the refactored version:

java

Copy code

```

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

```

```

public class FeatureBarPanel extends JPanel {
    private static final int button_width = 80;
    private static final int button_height = 40;

```

```

public FeatureBarPanel(JPanel canvas, ManageCanvas manageCanvas) {
    setLayout(new BorderLayout());
    setPreferredSize(new Dimension(200, 600));

    // Drawing tools setup
    DrawingTools drawingTools = new DrawingTools(canvas, manageCanvas);

    // Feature menu panel with borders
    JPanel featureMenuPanel = new JPanel(new BorderLayout());
    featureMenuPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

    // Tools panel with icons
    JPanel toolsPanel = new JPanel(new GridLayout(3, 2));
    addDrawingButton(toolsPanel, "Mouse", DrawingTools.DrawingMode.NONE,
drawingTools);
    addDrawingButton(toolsPanel, "Delete", DrawingTools.DrawingMode.DELETE,
drawingTools);
    addDrawingButton(toolsPanel, "Resize", DrawingTools.DrawingMode.RESIZE,
drawingTools);
    addDrawingButton(toolsPanel, "Rotate-Left", DrawingTools.DrawingMode.ROTATE_LEFT,
drawingTools);
    addDrawingButton(toolsPanel, "Rotate-Right",
DrawingTools.DrawingMode.ROTATE_RIGHT, drawingTools);
    addDrawingButton(toolsPanel, "Rotate-Flip", DrawingTools.DrawingMode.ROTATE_FLIP,
drawingTools);
    featureMenuPanel.add(toolsPanel, BorderLayout.NORTH);

    // Search bar panel
    JPanel searchBarPanel = new JPanel(new BorderLayout());
    searchBarPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));
    JLabel searchTitleLabel = new JLabel("Search:");
    searchBarPanel.add(searchTitleLabel, BorderLayout.WEST);
    JTextField searchBar = new JTextField();
    searchBar.setPreferredSize(new Dimension(150, 25));
    searchBar.setMaximumSize(new Dimension(150, 25));
    searchBar.setText("Search furniture...");
    searchBarPanel.add(searchBar, BorderLayout.CENTER);
    featureMenuPanel.add(searchBarPanel, BorderLayout.SOUTH);

    // Feature categories tabbed pane
    JTabbedPane tabbedPane = new JTabbedPane();
    tabbedPane.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

```



```

        tabbedPane.addTab("Dining Room", PanelCreation.createPanel("Dining Room Features",
drawingTools));
        tabbedPane.addTab("Living Room", PanelCreation.createPanel("Living Room Features",
drawingTools));
        tabbedPane.addTab("Bathroom", PanelCreation.createPanel("Bathroom Features",
drawingTools));
        tabbedPane.addTab("Kitchen", PanelCreation.createPanel("Kitchen Features",
drawingTools));
        tabbedPane.addTab("Bedroom", PanelCreation.createPanel("Bedroom Features",
drawingTools));
        tabbedPane.addTab("Structural", PanelCreation.createPanel("Structural Features",
drawingTools));
        featureMenuPanel.add(tabbedPane, BorderLayout.CENTER);

        add(featureMenuPanel, BorderLayout.CENTER);
    }

    private void addDrawingButton(Container container, String buttonText,
DrawingTools.DrawingMode mode, DrawingTools drawingTools) {
        addButtonWithIcon(container, buttonText, "none.png", button_width, button_height, mode,
drawingTools);
    }

    private void addButtonWithIcon(Container container, String buttonText, String iconFileName,
int width, int height, DrawingTools.DrawingMode mode, DrawingTools drawingTools) {
        String iconDirectory = "../assets";
        String iconPath = iconDirectory + "/" + iconFileName;
        ImageIcon icon = new ImageIcon(iconPath);
        JButton button = new JButton(buttonText, icon);
        button.setPreferredSize(new Dimension(width, height));
        button.addActionListener(e -> drawingTools.setDrawingMode(mode));
        container.add(button);
    }
}

```

In this refactored version, I've introduced a new method `addDrawingButton()` to handle adding drawing buttons specifically. This method then calls `addButtonWithIcon()` with the appropriate parameters. This separation keeps the code organized and easier to maintain.

User

there still a lot of parameters, could you perhaps separate it, let's say container would be one the `addDrawingButton` but not on `addButtonWithIcon`, so that we can add the button after our customization.

To be more clear, can we make it that `container.add(button)` is only at the first function, the second function just return a button

2 / 2

ChatGPT

Sure, we can refactor the code to separate the button creation and the addition to the container. Here's the modified version:

java

Copy code

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

```
public class FeatureBarPanel extends JPanel {
    private static final int button_width = 80;
    private static final int button_height = 40;

    public FeatureBarPanel(JPanel canvas, ManageCanvas manageCanvas) {
        setLayout(new BorderLayout());
        setPreferredSize(new Dimension(200, 600));

        // Drawing tools setup
        DrawingTools drawingTools = new DrawingTools(canvas, manageCanvas);

        // Feature menu panel with borders
        JPanel featureMenuPanel = new JPanel(new BorderLayout());
        featureMenuPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));

        // Tools panel with icons
        JPanel toolsPanel = new JPanel(new GridLayout(3, 2));
        addButtonsToPanel(toolsPanel, createDrawingButton("Mouse", "none.png",
            DrawingTools.DrawingMode.NONE, drawingTools));
        addButtonsToPanel(toolsPanel, createDrawingButton("Delete", "delete.png",
            DrawingTools.DrawingMode.DELETE, drawingTools));
        addButtonsToPanel(toolsPanel, createDrawingButton("Resize", "resize.png",
            DrawingTools.DrawingMode.RESIZE, drawingTools));
        addButtonsToPanel(toolsPanel, createDrawingButton("Rotate-Left", "rotate-left.png",
            DrawingTools.DrawingMode.ROTATE_LEFT, drawingTools));
        addButtonsToPanel(toolsPanel, createDrawingButton("Rotate-Right", "rotate-right.png",
            DrawingTools.DrawingMode.ROTATE_RIGHT, drawingTools));
        addButtonsToPanel(toolsPanel, createDrawingButton("Rotate-Flip", "rotate-180.png",
            DrawingTools.DrawingMode.ROTATE_FLIP, drawingTools));
        featureMenuPanel.add(toolsPanel, BorderLayout.NORTH);
    }
}
```

```

// Search bar panel
JPanel searchBarPanel = new JPanel(new BorderLayout());
searchBarPanel.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));
JLabel searchTitleLabel = new JLabel("Search:");
searchBarPanel.add(searchTitleLabel, BorderLayout.WEST);
JTextField searchBar = new JTextField();
searchBar.setPreferredSize(new Dimension(150, 25));
searchBar.setMaximumSize(new Dimension(150, 25));
searchBar.setText("Search furniture...");
searchBarPanel.add(searchBar, BorderLayout.CENTER);
featureMenuPanel.add(searchBarPanel, BorderLayout.SOUTH);

// Feature categories tabbed pane
JTabbedPane tabbedPane = new JTabbedPane();
tabbedPane.setBorder(BorderFactory.createLineBorder(Color.LIGHT_GRAY));
tabbedPane.addTab("Dining Room", PanelCreation.createPanel("Dining Room Features",
drawingTools));
tabbedPane.addTab("Living Room", PanelCreation.createPanel("Living Room Features",
drawingTools));
tabbedPane.addTab("Bathroom", PanelCreation.createPanel("Bathroom Features",
drawingTools));
tabbedPane.addTab("Kitchen", PanelCreation.createPanel("Kitchen Features",
drawingTools));
tabbedPane.addTab("Bedroom", PanelCreation.createPanel("Bedroom Features",
drawingTools));
tabbedPane.addTab("Structural", PanelCreation.createPanel("Structural Features",
drawingTools));
featureMenuPanel.add(tabbedPane, BorderLayout.CENTER);

add(featureMenuPanel, BorderLayout.CENTER);
}

private JButton createDrawingButton(String buttonText, String iconFileName,
DrawingTools.DrawingMode mode, DrawingTools drawingTools) {
    String iconDirectory = "../assets";
    String iconPath = iconDirectory + "/" + iconFileName;
    ImageIcon icon = new ImageIcon(iconPath);
    JButton button = new JButton(buttonText, icon);
    button.setPreferredSize(new Dimension(button_width, button_height));
    button.addActionListener(e -> drawingTools.setDrawingMode(mode));
    return button;
}

```

```

        private void addButtonToPanel(Container container, JButton button) {
            container.add(button);
        }
    }

```

In this version, createDrawingButton() method is responsible for creating a button with the specified text, icon, and action listener. It returns the created button. Then, addButtonToPanel() method simply adds the provided button to the specified container. This separation makes the code cleaner and easier to understand.

User

Okay nice,

```

private void deleteObject(Point point) {
    if (deleteMode) {
        // Iterate over objects in canvas and check if point is inside any object
        // If found, remove the object from ManageCanvas and redraw canvas
    }
}

```

can you complete this code please?

ChatGPT

Certainly! Here's how you can complete the deleteObject method:

java

Copy code

```

private void deleteObject(Point point) {
    if (deleteMode) {
        // Iterate over objects in canvas and check if point is inside any object
        for (StructureObject structureObject : manageCanvas.getStructureObjects()) {
            if (isPointInsideObject(point, structureObject)) {
                manageCanvas.removeStructureObject(structureObject);
                redrawCanvas();
                return; // Exit the method once an object is deleted
            }
        }

        for (FurnitureObject furnitureObject : manageCanvas.getFurnitureObjects()) {
            if (isPointInsideObject(point, furnitureObject)) {
                manageCanvas.removeFurnitureObject(furnitureObject);
                redrawCanvas();
                return; // Exit the method once an object is deleted
            }
        }
    }
}

```

```
}
```

```
private boolean isPointInsideObject(Point point, StructureObject structureObject) {  
    // Check if the point is inside the boundaries of the structureObject  
    // Implement this method based on your specific object shapes and properties  
    return false; // Placeholder return, replace with actual implementation  
}
```

```
private boolean isPointInsideObject(Point point, FurnitureObject furnitureObject) {  
    // Check if the point is inside the boundaries of the furnitureObject  
    // Implement this method based on your specific object shapes and properties  
    return false; // Placeholder return, replace with actual implementation  
}
```

```
private void redrawCanvas() {  
    // Redraw the canvas after deleting an object  
    // You might need to call methods to clear the canvas and redraw all objects  
    // Implement this method based on your canvas implementation  
}
```

Explanation:

deleteObject: This method checks if the delete mode is active. If yes, it iterates over the structure and furniture objects stored in the ManageCanvas. For each object, it checks if the given point is inside the object. If found, it removes the object from ManageCanvas and redraws the canvas.

isPointInsideObject: This method is a placeholder for checking if a given point is inside a specific object. You'll need to implement this method based on your object shapes and properties. For example, for a rectangular object, you would check if the point's coordinates are within the object's bounds.

redrawCanvas: This method is called after deleting an object to redraw the canvas. You'll need to implement this method based on your canvas implementation. It might involve clearing the canvas and redrawing all objects.

User

is there any way to check if the point is around the width and height of the startingPoint(?)
ChatGPT

Yes, you can check if a point is within a certain distance from another point (e.g., the starting point) by calculating the Euclidean distance between the two points and comparing it to a threshold value. If the distance is less than or equal to the threshold, then the points are considered close to each other.

Here's how you can modify the `isPointInsideObject` method to check if a given point is within a certain distance from the starting point of the object:

java

Copy code

```
private boolean isPointInsideObject(Point point, StructureObject structureObject, int threshold) {  
    // Calculate the Euclidean distance between the given point and the starting point of the  
    object  
    double distance = point.distance(structureObject.getStartPoint());  
  
    // Check if the distance is within the threshold  
    return distance <= threshold;  
}
```

In this method, `threshold` represents the maximum distance allowed for the point to be considered "around" the starting point. You can adjust this threshold value based on your specific requirements. If the distance between the given point and the starting point of the object is within the threshold, the method returns `true`, indicating that the point is considered inside the object.

You can then use this method within the `deleteObject` method to determine if a given point is around the starting point of a structure object. Adjust the threshold value as needed for your application.

User

Okay, I want to record this chat, can you please summarize everything in headings and bulletpoints under it

ChatGPT

Certainly! Here's a summary of the key points discussed:

Application Architecture:

- Architecture designed for managing furniture objects within a canvas-based application.
- Each furniture object class encapsulates properties like position, size, and rotation.
- Concept of a separate class (e.g., `ManageCanvas`) to manage overall scene on the canvas.
- `ManageCanvas` acts as a centralized repository for storing, manipulating furniture objects, and maintaining other scene elements.
- Maintain lists of objects, assign unique IDs, track positions, and attributes.
- Use of command objects for features like undo/redo functionality.

Components:

StructureObject Class:

- Abstract class representing structural elements like walls, doors, and windows.
- Contains start and end points.
- Provides method to draw the object.

FurnitureObject Class:

- Abstract class representing furniture items like beds, desks, chairs, etc.
- Contains start point, width, and height.
- Provides method to draw the object.

DrawingTools Class:

- Handles mouse clicks and drawing functionalities on canvas.
- Sets up different drawing modes (e.g., wall, window, bed) based on user input.
- Draws structural and furniture objects based on selected mode.

FeatureBarPanel Class:

- Creates a panel with tools/buttons for drawing, deleting, resizing, and rotating objects.
- Includes a search bar and tabs for different feature categories.

App Class:

- Main class initiating the application.
- Sets up the main canvas panel, feature bar panel, and menu bar.

Functionality Implementation:

Deleting Objects:

- Implement deleteObject method in DrawingTools class.
- Iterate over objects in canvas and check if clicked point is inside any object.
- If found, remove the object from ManageCanvas and redraw canvas.

Checking Point Proximity:

- Use Euclidean distance to check if a point is within a certain distance from another point.
- Modify isPointInsideObject method to include a threshold parameter for proximity check.

Refactoring:

- Refactor code for button creation and addition to panels to improve readability and maintainability.
- Separate methods for creating buttons and adding them to panels.
- Encapsulate button creation logic in separate methods to reduce parameter count and improve clarity.

Summary:

The application architecture is designed to manage furniture objects within a canvas-based environment efficiently. Components like StructureObject, FurnitureObject, DrawingTools, FeatureBarPanel, and App are crucial for implementing various functionalities like drawing, deleting, and managing objects. Refactoring the code for better organization and readability enhances the development process.