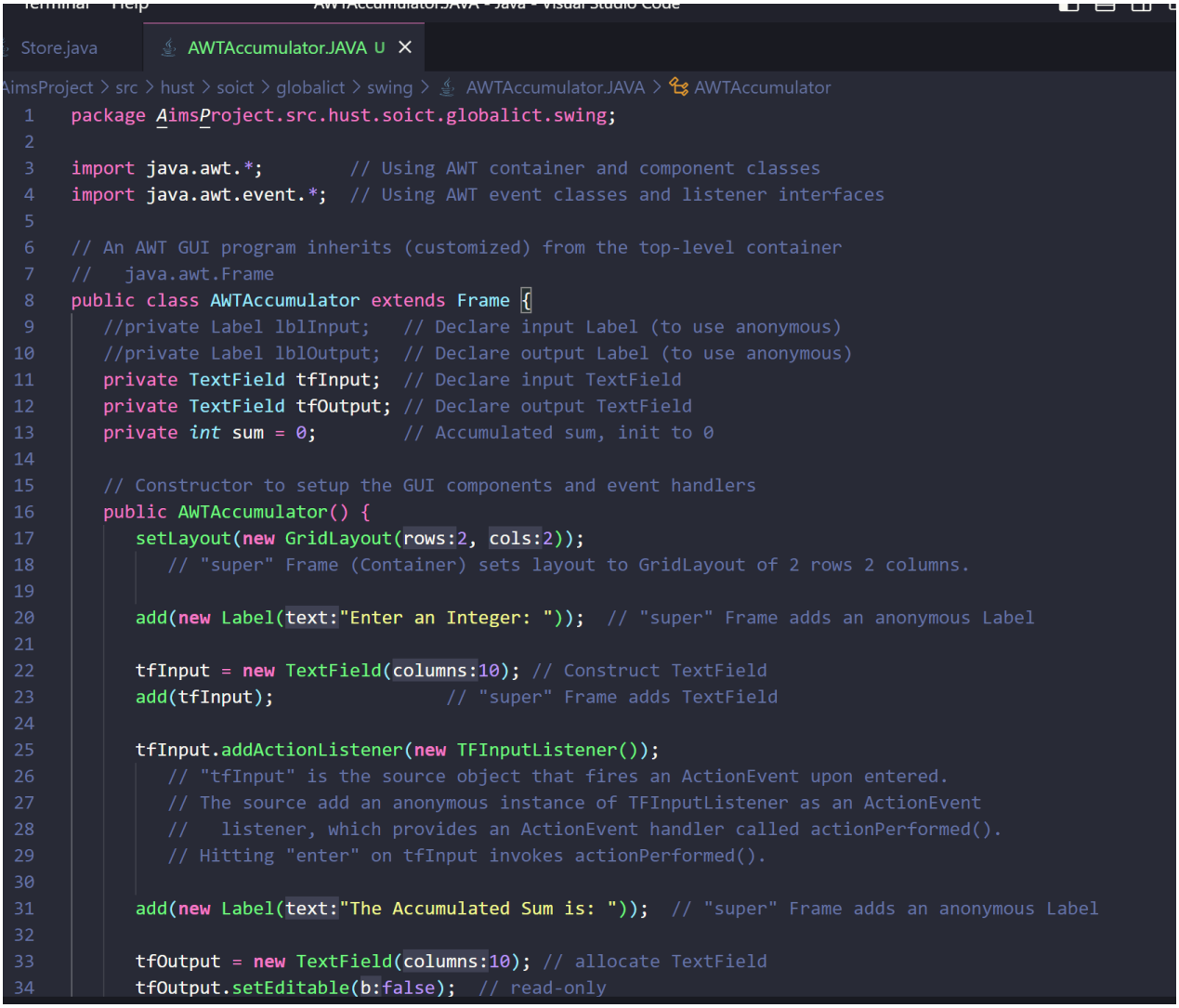
Lab 05: GUI Programming

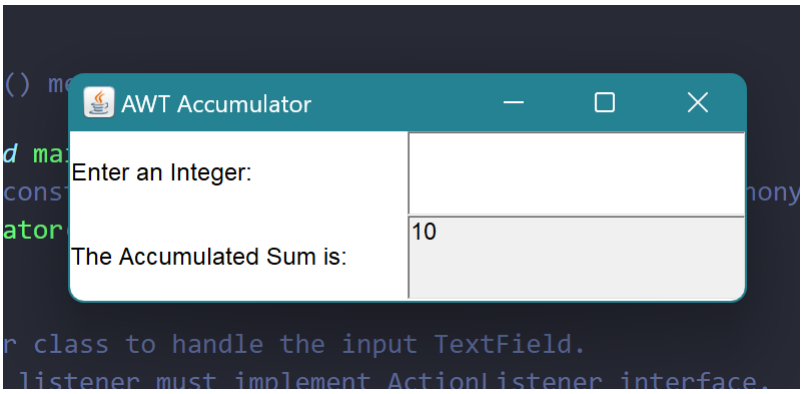
Lê Hà Ngân – 20215230

1. Swing components

1. AWTAccumulator



```
      add(new Label(text:"The Accumulated Sum is: "));  // "super" Frame adds an anonymous Label

      tfOutput = new TextField(columns:10); // allocate TextField
      tfOutput.setEditable(b:false);  // read-only
      add(tfOutput);                // "super" Frame adds TextField

      setTitle(title:"AWT Accumulator");  // "super" Frame sets title
      setSize(width:350, height:120);  // "super" Frame sets initial window size
      setVisible(b:true);   // "super" Frame shows
   }

   // The entry main() method
   Run | Debug
   public static void main(String[] args) {
      // Invoke the constructor to setup the GUI, by allocating an anonymous instance
      new AWTAccumulator();
   }

   // Define an inner class to handle the input TextField.
   // An ActionEvent listener must implement ActionListener interface.
   private class TFInputListener implements ActionListener {
      // ActionEvent handler - Called back upon hitting "enter" key on TextField
      @Override
      public void actionPerformed(ActionEvent evt) {
         // Get the String entered into the TextField tfInput, convert to int
         int numberIn = Integer.parseInt(tfInput.getText());
         sum += numberIn;      // Accumulate numbers entered into sum
         tfInput.setText(t:"");  // Clear input TextField
         tfOutput.setText(sum + ""); // Display sum on the output TextField
                                     // convert int to String
      }
   }
```



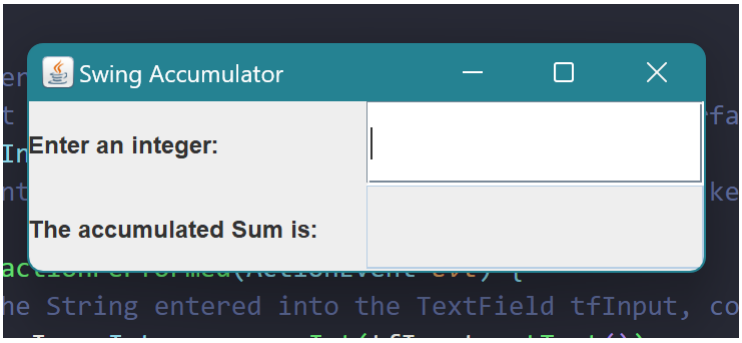
2. SwingAccumulator

Store.java | AWTAccumulator.JAVA U | SwingAccumulator.java U

msProject > src > hust > soict > globalict > swing > SwingAccumulator.java > SwingAccumulator

```java
package AimsProject.src.hust.soict.globalict.swing;

import java.awt.*; // Using AWT's layouts
import java.awt.event.*; // Using AWT's event classes and listener interfaces
import javax.swing.*; // Using Swing components and containers

public class SwingAccumulator extends JFrame {
    private JTextField tfInput;
    private JTextField tfOutput;
    private int sum = 0;

    public SwingAccumulator() {
        Container cp = getContentPane();
        cp.setLayout(new GridLayout(rows:2, cols:2));

        cp.add(new JLabel(text:"Enter an integer: "));

        tfInput = new JTextField(columns:10);
        cp.add(tfInput);
        tfInput.addActionListener(new TFInputListener());

        cp.add(new JLabel(text:"The accumulated Sum is:"));

        tfOutput = new JTextField(columns:10);
        tfOutput.setEditable(b:false);
        cp.add(tfOutput);

        setTitle(title:"Swing Accumulator");
        setSize(width:350, height:120);
        setVisible(b:true);

    }

```

```
Store.java    AWTAccumulator.JAVA U    SwingAccumulator.java U
msProject > src > hust > soict > globalict > swing > SwingAccumulator.java > SwingAccumulator
        setTitle(title: "Swing Accumulator");
        setSize(width:350, height:120);
        setVisible(b:true);

    }

    Run | Debug
    public static void main(String[] args) {
        new SwingAccumulator();
    }

    // Define an inner class to handle the input TextField.
    // An ActionEvent listener must implement ActionListener interface.
    private class TFInputListener implements ActionListener {
        // ActionEvent handler - Called back upon hitting "enter" key on TextField
        @Override
        public void actionPerformed(ActionEvent evt) {
            // Get the String entered into the TextField tfInput, convert to int
            int numberIn = Integer.parseInt(tfInput.getText());
            sum += numberIn; // Accumulate numbers entered into sum
            tfInput.setText(t:""); // Clear input TextField
            tfOutput.setText(sum + ""); // Display sum on the output TextField
                                        // convert int to String
        }
    }

}
```



2. Organizing Swing components with Layout Managers

2.2.1. Create class NumberGrid


```java
package AimsProject.src.hust.soict.globalict.swing;

import java.awt.*; // Using AWT's layouts
import java.awt.event.*; // Using AWT's event classes and listener interfaces
import javax.swing.*; // Using Swing components and containers

public class NumberGrid extends JFrame {
    private JButton[] btnNumbers = new JButton[10];
    private JButton btnDelete, btnReset;
    private JTextField tfDisplay;

    public NumberGrid() {
        tfDisplay = new JTextField();
        tfDisplay.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
        JPanel panelButtons = new JPanel(new GridLayout(rows:4, cols:3));
        addButton(panelButtons);
        Container cp = getContentPane();
        cp.setLayout(new BorderLayout());
        cp.add(tfDisplay, BorderLayout.NORTH);
        cp.add(panelButtons, BorderLayout.CENTER);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setTitle(title:"Number Grid");
        setSize(width:200, height:200);
        setVisible(b:true);
    }

    void addButton(JPanel panelButtons) {
        ButtonListener btnListener = new ButtonListener();
        for (int i = 1; i <= 9; i++) {
            btnNumbers[i] = new JButton("" + i);
            panelButtons.add(btnNumbers[i]);
            btnNumbers[i].addActionListener(btnListener);
```

2.2.2. Adding buttons

```
    void addButton(JPanel panelButtons) {
        ButtonListener btnListener = new ButtonListener();
        for (int i = 1; i <= 9; i++) {
            btnNumbers[i] = new JButton("" + i);
            panelButtons.add(btnNumbers[i]);
            btnNumbers[i].addActionListener(btnListener);
        }

        btnDelete = new JButton(text:"DEL");
        panelButtons.add(btnDelete);
        btnDelete.addActionListener(btnListener);

        btnNumbers[0] = new JButton(text:"0");
        panelButtons.add(btnNumbers[0]);
        btnNumbers[0].addActionListener(btnListener);

        btnReset = new JButton(text:"C");
        panelButtons.add(btnReset);
        btnReset.addActionListener(btnListener);
    }
```

2.2.3. Complete inner class ButtonListener

```
    private class ButtonListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String button = e.getActionCommand();
            if (button.charAt(index:0) >= '0' && button.charAt(index:0) <= '9') {
                tfDisplay.setText(tfDisplay.getText() + button);
            } else if (button.equals(anObject:"DEL")) {
                // handle the "DEL" case
                // Delete the last character from the display text
                String currentText = tfDisplay.getText();
                if (!currentText.isEmpty()) {
                    tfDisplay.setText(currentText.substring(beginIndex:0, currentText.length() - 1));
                }
            } else {
                // Hanles the "C" case
                // Clear the display text
                tfDisplay.setText(t:"");
            }
        }
    }

    Run | Debug
    public static void main(String[] args) {
        new NumberGrid();
    }
}
```

3. Create a graphical user interface for AIMS with Swing

3.1.1. Create the StoreScreen class

```
public class StoreScreen extends JFrame{
    private Store store;
    JPanel createNorth(){
        JPanel north = new JPanel();
        north.setLayout(new BoxLayout(north , BoxLayout.Y_AXIS));

        // sets the layout manager for the container north to be a vertical BoxLayout.
        // This means that components added to the north container will be arranged in a
        //  one on top of another.

        north.add(createMenuBar());
        north.add(createHeader());
        return north;
    }

    JMenuBar createMenuBar(){
        JMenu menu = new JMenu(s:"Options");

        JMenu smUpdateStore = new JMenu(s:"Update Store");
        smUpdateStore.add(new JMenuItem(text:"Add Book"));
        smUpdateStore.add(new JMenuItem(text:"Add CD"));
        smUpdateStore.add(new JMenuItem(text:"Add DVD"));
        menu.add(smUpdateStore);
        menu.add(new JMenuItem(text:"View store"));
        menu.add(new JMenuItem(text:"View cart"));

        JMenuBar menuBar = new JMenuBar();
        menuBar.setLayout(new FlowLayout(FlowLayout.LEFT));
        // This means that components added to the container will be arranged
        // in a flow from left to right, wrapping to the next line if necessary.

        menuBar.add(menu);
```

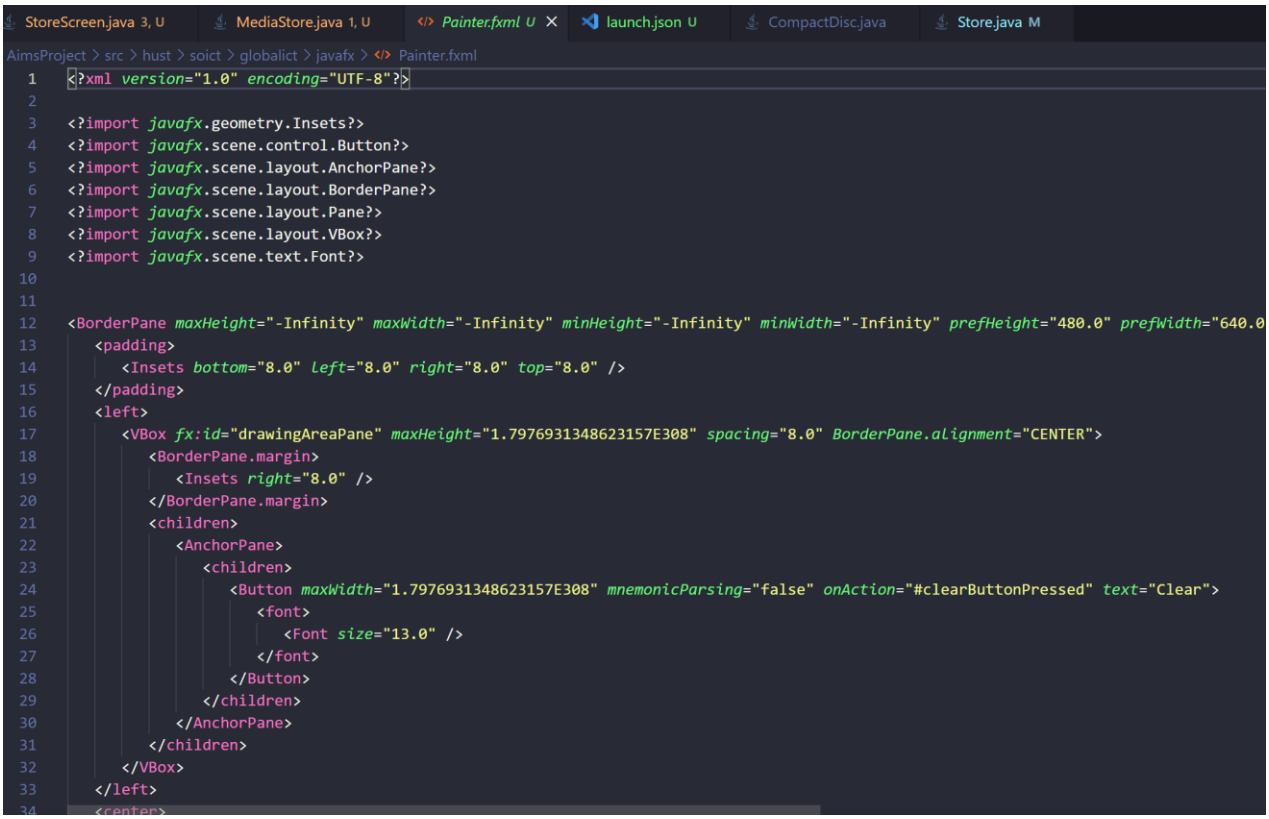
```
JPanel createHeader(){
    JPanel header = new JPanel();
    header.setLayout(new BoxLayout(header, BoxLayout.X_AXIS));

    JLabel title = new JLabel(text:"AIMS");
    title.setFont(new Font(title.getFont().getName() , Font.PLAIN , size:50));

    title.setForeground(Color.CYAN);

    JButton cart = new JButton(text:"View cart");
    cart.setPreferredSize(new Dimension(width:100, height:50));
    cart.setMaximumSize(new Dimension(width:100, height:50));

    header.add(Box.createRigidArea(new Dimension(width:10,height:10))); // create area
    header.add(title);
    header.add(Box.createHorizontalGlue());
    header.add(cart);
    header.add(Box.createRigidArea(new Dimension(width:10, height:10)));
    return header;
}

JPanel createCenter(){
    JPanel center = new JPanel();
    center.setLayout(new GridLayout(rows:3, cols:3, hgap:2, vgap:2));

    ArrayList<Media> mediaInstore = (ArrayList) store.getItemsInStore();
    for( int i = 0 ; i<9 ; i++){
        MediaStore cell = new MediaStore(mediaInstore.get(i));
        center.add(cell);
    }

    return center;
}
```

3.1.4. The MediaStore class

```java
public class MediaStore extends JPanel {
    private Media media;
    private Cart cart;

    public MediaStore(Media media) {
        this.media = media;
        this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));

        JLabel title = new JLabel(media.getTitle());
        title.setFont(new Font(title.getFont().getName(), Font.PLAIN, size:20));
        title.setAlignmentX(CENTER_ALIGNMENT);

        JLabel cost = new JLabel("" + media.getCost() + "$");
        cost.setAlignmentX(CENTER_ALIGNMENT);

        JPanel container = new JPanel();
        container.setLayout(new FlowLayout(FlowLayout.CENTER));

        JButton addToCartButton = new JButton(text:"Add to cart");
        container.add(addToCartButton);
        addToCartButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    cart.addMedia(media);
                    JOptionPane.showMessageDialog(parentComponent:null, "Media added to cart: " + media.getTitle(), title:"Playable M
                            JOptionPane.INFORMATION_MESSAGE);
                } catch (Exception exception) {
                    JOptionPane.showMessageDialog(parentComponent:null, exception.getMessage(), exception.toString(),
                            JOptionPane.ERROR_MESSAGE);
                }
            }
        });
```

```java
        // a “Play” button
        if (media instanceof Playable) {
            // container.add(new JButton("Play"));
            JButton playButton = new JButton(text:"Play");
            container.add(playButton);

            playButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    // Play button clicked, show media in a dialog window
                    JDialog dialog = new JDialog();
                    dialog.setSize(width:400, height:300); // Set dialog size as per your requirements
                    // Add code to display the media in the dialog
                    String playable = ((Playable)media).toString();
                    JLabel label = new JLabel("Playing " + playable );
                    dialog.add(label);
                    dialog.setVisible(b:true);
                }
            });
        }

        this.add(Box.createVerticalGlue());
        this.add(title);
        this.add(cost);
        this.add(Box.createHorizontalGlue());
        this.add(container);

        this.setBorder(BorderFactory.createLineBorder(Color.BLACK));

    }
}
```

4. JavaFX API



4.2. Create the controller class


AimsProject > src > hust > soict > globalict > javafx > PainterController.java > PainterController > drawingAreaMouseDragged(Mous
1 package AimsProject.src.hust.soict.globalict.javafx;
2
3 import javafx.event.ActionEvent;
4 import javafx.fxml.FXML;
5 import javafx.scene.Node;
6 import javafx.scene.control.RadioButton;
7 import javafx.scene.control.ToggleGroup;
8 import javafx.scene.input.MouseEvent;
9 import javafx.scene.layout.Pane;
10 import javafx.scene.shape.*;
11
12 public class PainterController {
13 @FXML
14 private Pane drawingAreaPane;
15
16 @FXML
17 private RadioButton eraserRadio;
18
19 @FXML
20 private ToggleGroup pen;
21
22 @FXML
23 private RadioButton penRadio;
24
25 @FXML
26 void clearButtonPressed(ActionEvent event) {
27 drawingAreaPane.getChildren().clear();
28 }
29
30 @FXML
31 void drawingAreaMouseDragged(MouseEvent event) {
32 if (penRadio.isSelected()) {
33 Circle newCircle = new Circle(event.getX(), event.getY(), 4);
34

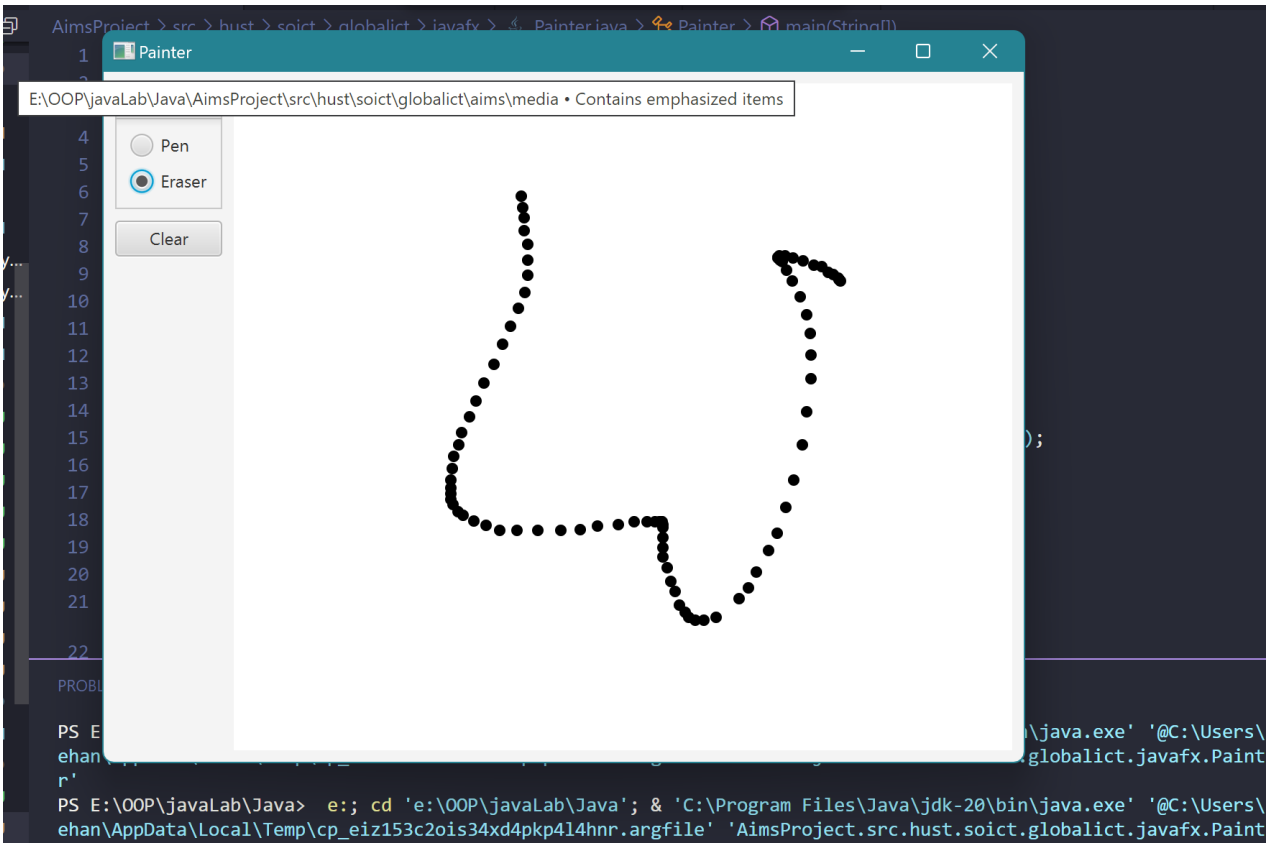
```java
    @FXML
    private RadioButton penRadio;

    @FXML
    void clearButtonPressed(ActionEvent event) {
        drawingAreaPane.getChildren().clear();
    }

    @FXML
    void drawingAreaMouseDragged(MouseEvent event) {
        if (penRadio.isSelected()) {
            Circle newCircle = new Circle(event.getX(), event.getY(), 4);

            drawingAreaPane.getChildren().add(newCircle);
        } else if (eraserRadio.isSelected()) {
            for (int i = drawingAreaPane.getChildren().size() - 1; i >= 0; i--) {
                Node child = drawingAreaPane.getChildren().get(i);
                if (child instanceof Circle) {
                    Circle circle = (Circle) child;
                    double circleX = circle.getCenterX();
                    double circleY = circle.getCenterY();

                    if (Math.abs(circle.getCenterX() - event.getX()) <= 8 &&
                            Math.abs(circle.getCenterY() - event.getY()) <= 8) {
                        drawingAreaPane.getChildren().remove(i);
                    }
                }
            }
        }
    }

}
```

4.3. Create the application

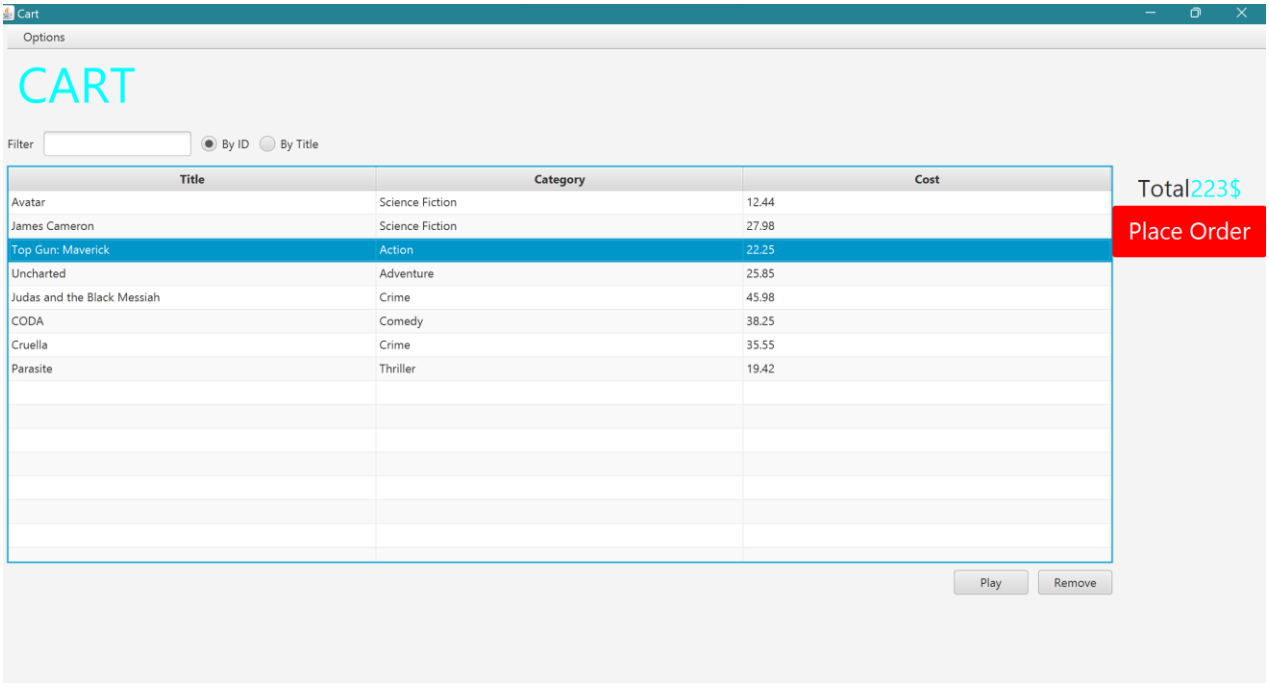
AimsProject > src > hust > soict > globalict > javafx > Painter.java > Painter > main(String[])

```java
package AimsProject.src.hust.soict.globalict.javafx;

import java.net.URL;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Painter extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource(name:"Painter.fxml"));
        Scene scene = new Scene(root);
        stage.setTitle("Painter");
        stage.setScene(scene);
        stage.show();
    }

    Run | Debug
    public static void main(String[] args) {
        launch(args);
    }
}
```

4.4. Practice exercise



5. **Setting up the View Cart Screen with ScreenBuilder**



6. **Integrating JavaFX into Swing application – The JFXPanel class**

```
import javafx.fxml.*;

public class CartScreen extends JFrame {
    private Cart cart;

    public CartScreen (Cart cart) {
        super();

        this.cart = cart;

        JFXPanel fxPanel = new JFXPanel();
        // we set up a JFXPanel in our JFrame.
        this.add(fxPanel);
        this.setTitle(title:"Cart");
        this.setVisible(b:true);

        Platform.runLater( new Runnable() {
            @Override
            public void run(){
                try{
                    FXMLLoader loader = new FXMLLoader(getClass().getResource(name:"cart.fxml"));
                    CartScreenController controller = new CartScreenController(cart);
                    loader.setController(controller);
                    Parent root = loader.load();
                    fxPanel.setScene(new Scene(root));
                }catch(IOException e){
                    e.printStackTrace();
                }
            }
        });
    }

```

7. **View the items in cart – JavaFX's data-driven UI**

```java
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;

public class CartScreenController {
    private Cart cart;

    @FXML
    private Button btnPlay;

    @FXML
    private Button btnRemove;

    @FXML
    private TableColumn<Media, Float> colMediaCost;

    @FXML
    private TableColumn<Media, String> colMediaTitle;

    @FXML
    private TableColumn<Media, String> colMediacategory;

    @FXML
    private ToggleGroup filterCategory;

    @FXML
    private TableView<Media> tblMedia;

    @FXML
    private RadioButton radioBtnFilterId;

    @FXML
    private RadioButton radioBtnFilterTitle;
```

```
    @FXML
    private RadioButton radioBtnFilterTitle;

    @FXML
    private TextField tfFilter;

    @FXML
    private Label totalCost;

    public CartScreenController(Cart cart) {
        super();
        this.cart = cart;
    }

    @FXML
    private void initialize() {
        colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"title"));
        colMediacategory.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"category"))
        colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, Float>(property:"cost"));
        tblMedia.setItems(this.cart.getItemsOrdered());

        btnPlay.setVisible(false);
        btnRemove.setVisible(false);
        int cost = 0;
        for (Media m : cart.getItemsOrdered()) {
            cost += m.getCost();
        }
        totalCost.setText(cost + "$");
        tblMedia.getSelectionModel().selectedItemProperty().addListener(
                new ChangeListener<Media>() {
                    @Override
                    public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media
                        if (newValue != null) {
```

8. **Updating buttons based on selected item in TableView –ChangeListener**

```
@FXML
private void initialize() {
    colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"title"));
    colMediacategory.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"category"))
    colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, Float>(property:"cost"));
    tblMedia.setItems(this.cart.getItemsOrdered());

    btnPlay.setVisible(false);
    btnRemove.setVisible(false);
    int cost = 0;
    for (Media m : cart.getItemsOrdered()) {
        cost += m.getCost();
    }
    totalCost.setText(cost + "$");
    tblMedia.getSelectionModel().selectedItemProperty().addListener(
            new ChangeListener<Media>() {
                @Override
                public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media
                    if (newValue != null) {
                        updateButtonBar(newValue);
                    }
                }

                void updateButtonBar(Media media) {
                    btnRemove.setVisible(true);
                    if (media instanceof Playable) {
                        btnPlay.setVisible(true);
                    } else {
                        btnPlay.setVisible(false);
                    }
                }
            });
```

9. Deleting a media

```
    opens com.foo to javafx.fxml;
@FXML
void btnRemovePressed(ActionEvent e) {
    Media media = tblMedia.getSelectionModel().getSelectedItem();
    cart.removeMedia(media);
    // changeTotalCost();
    // Note that we don't need to invoke an update for the TableView because the it
    // can already observe the
    // changes though the ObservableList and update its display
}
```

10.Filter items in cart – FilteredList

```
    tfFilter.textProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newV
            showFilteredMedia(newValue);
        }

        void showFilteredMedia(String value) {
            boolean filterById = radioBtnFilterId.isSelected();
            boolean filterByTitle = radioBtnFilterTitle.isSelected();
            // Create a FilteredList based on the original media list
            FilteredList<Media> filteredList = new FilteredList<>(cart.getItemsOrdered());

            filteredList.setPredicate(media -> {
                if (value.isEmpty()) {
                    return true;
                } else if (filterById) {
                    // Filter by ID
                    return String.valueOf(media.getId()).contains(value);
                } else if (filterByTitle) {
                    // Filter by title
                    return media.getTitle().contains(value);
                }
                return false;
            });
            tblMedia.setItems(filteredList);

        }
    });

}
```

12.Check all the previous source codes to catch/handle/delegate runtime exceptions

```
public void addMedia(Media media) throws LimitExceededException {
    if (!isFulled()) {
        itemsOrdered.add(media);
    } else {
        throw new LimitExceededException(explanation:"ERROR: The number of media has reached its limit

    }
}
```

13.Create a class which inherits from Exception

```java
package AimsProject.src.hust.soict.globalict.aims.exception;

import java.lang.Exception;

public class PlayerException extends Exception{
    public PlayerException() {
        // TODO Auto-generated constructor stub
    }

    public PlayerException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

    public PlayerException(Throwable cause) {
        super(cause);
        // TODO Auto-generated constructor stub
    }

    public PlayerException(String message, Throwable cause) {
        super(message, cause);
        // TODO Auto-generated constructor stub
    }

    public PlayerException(String message, Throwable cause, boolean enableSuppression, boolean writableSta
        super(message, cause, enableSuppression, writableStackTrace);
        // TODO Auto-generated constructor stub
    }
}
```

13.4.Update play() in CompactDisc

```java
@Override
public void play() throws PlayerException{
    if(this.getLength() >0 ){
        // TODO Play all tracks in the CD as you have implemented
        java.util.Iterator iter = tracks.iterator();
        Track nextTrack;
        while(iter.hasNext()){
            nextTrack = (Track) iter.next();
            try {
                nextTrack.play();
            } catch (PlayerException e) {
                // TODO: handle exception
                throw e;
            }
        }
    }
    else{
        throw new PlayerException(message:"ERROR: CD length is non-positive");
    }
}
```