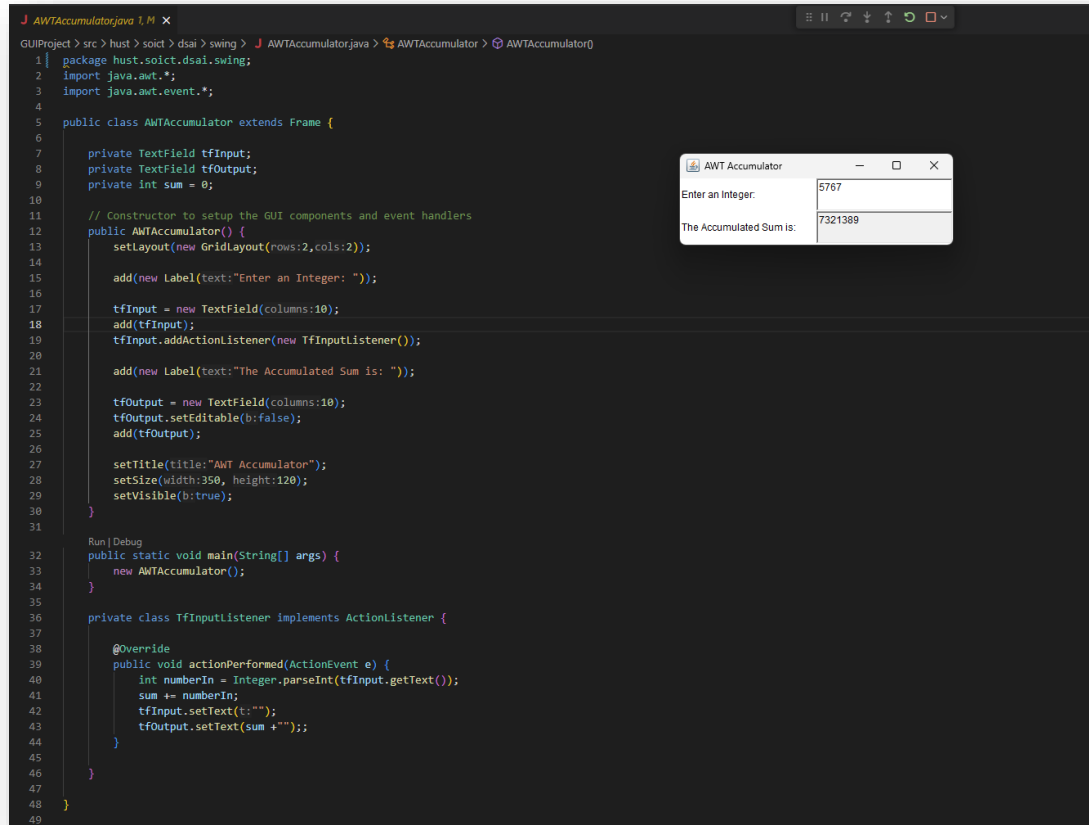


## LAB05 REPORT

### 1. Swing components

#### 1.1 AWTAccumulator



The screenshot shows an IDE window titled "J AWTAccumulator.java 1,1" with a dark theme. The code is as follows:

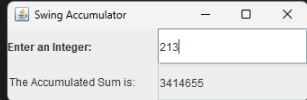
```
1 package hust.soict.dsai.swing;
2 import java.awt.*;
3 import java.awt.event.*;
4
5 public class AWTAccumulator extends Frame {
6
7     private TextField tfInput;
8     private TextField tfOutput;
9     private int sum = 0;
10
11     // Constructor to setup the GUI components and event handlers
12     public AWTAccumulator() {
13         setLayout(new GridLayout(2,2));
14
15         add(new Label(text:"Enter an Integer: "));
16
17         tfInput = new TextField(columns:10);
18         add(tfInput);
19         tfInput.addActionListener(new TfinputListener());
20
21         add(new Label(text:"The Accumulated Sum is: "));
22
23         tfOutput = new TextField(columns:10);
24         tfOutput.setEditable(false);
25         add(tfOutput);
26
27         setTitle(title:"AWT Accumulator");
28         setSize(width:350, height:120);
29         setVisible(b:true);
30     }
31
32     Run | Debug
33     public static void main(String[] args) {
34         new AWTAccumulator();
35     }
36
37     private class TfinputListener implements ActionListener {
38
39         @Override
40         public void actionPerformed(ActionEvent e) {
41             int numberIn = Integer.parseInt(tfInput.getText());
42             sum += numberIn;
43             tfInput.setText("");
44             tfOutput.setText(sum + "");
45         }
46     }
47
48 }
49
```

Overlaid on the code is a window titled "AWT Accumulator". It contains two rows:

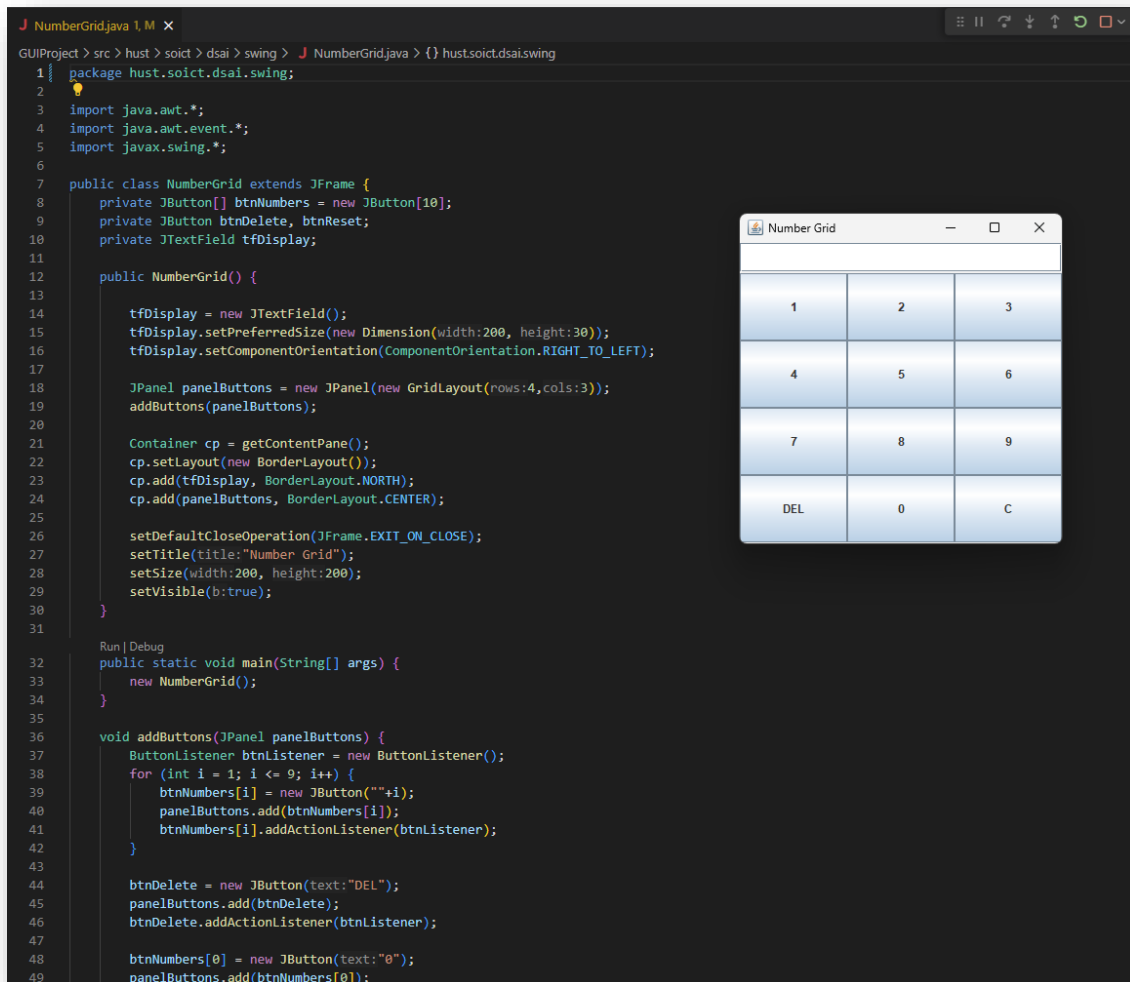
Enter an Integer:	5767
The Accumulated Sum is:	7321389

#### 1.2 SwingAccumulator

```
J SwingAccumulator.java 1,11 X
GUIProject > src > hust > soict > dsai > swing > J SwingAccumulator.java > ...
1 | package hust.soict.dsai.swing;
2
3 import java.awt.*;
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6
7 import javax.swing.*;
8
9 public class SwingAccumulator extends JFrame {
10
11     private JTextField tfInput;
12     private JTextField tfOutput;
13     private int sum = 0;
14
15     // Constructor to setup the GUI components and event handlers
16     public SwingAccumulator() {
17         Container cp = getContentPane();
18         cp.setLayout(new GridLayout(2,2));
19
20         cp.add(new JLabel(text:"Enter an Integer: "));
21
22         tfInput = new JTextField(columns:10);
23         cp.add(tfInput);
24         tfInput.addActionListener(new TfInputListener());
25
26         cp.add(new JLabel(text:"The Accumulated Sum is: "));
27
28         tfOutput = new JTextField(columns:10);
29         tfOutput.setEditable(b:false);
30         cp.add(tfOutput);
31
32         setTitle(title:"Swing Accumulator");
33         setSize(width:350, height:120);
34         setVisible(b:true);
35     }
36
37     Run | Debug
38     public static void main(String[] args) {
39         new SwingAccumulator();
40     }
41
42     private class TfInputListener implements ActionListener {
43
44         @Override
45         public void actionPerformed(ActionEvent e) {
46             int numberIn = Integer.parseInt(tfInput.getText());
47             sum += numberIn;
48             tfInput.setText("");
49             tfOutput.setText(sum + "");
50         }
51     }
52 }
```

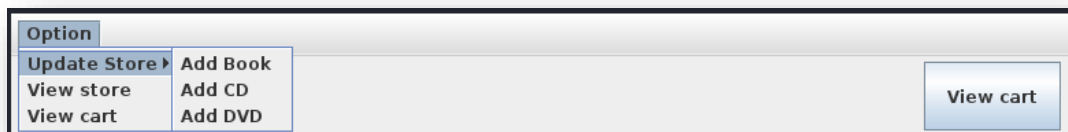


## 2. Organizing Swing components with Layout Managers



### 3. Create a graphical user interface for AIMS with Swing

#### 3.1. View Store Screen



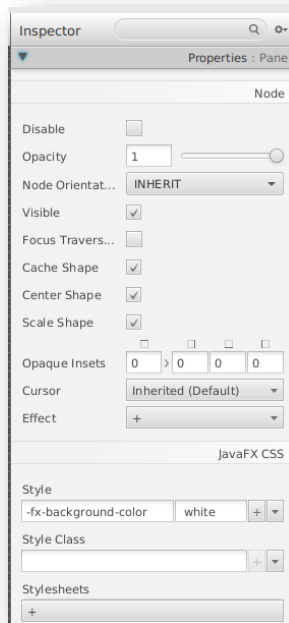
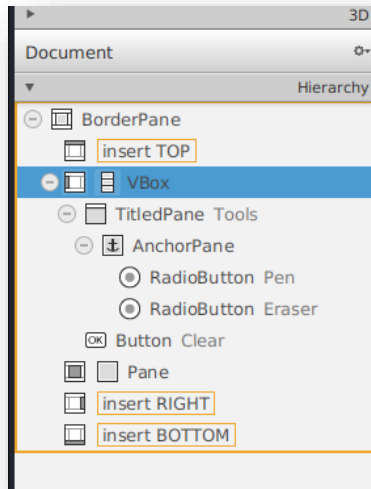
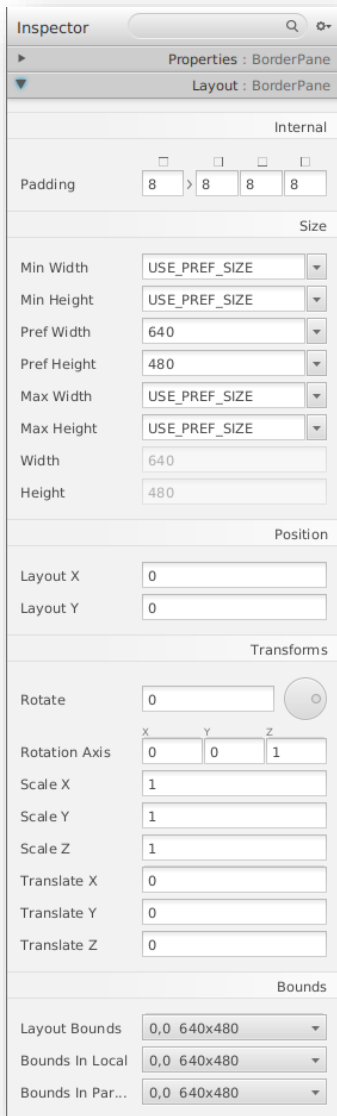
#### 3.2. Adding more user interaction

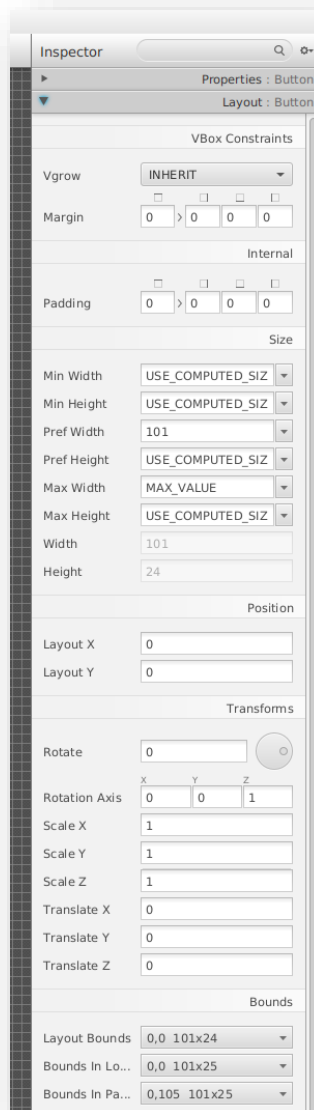
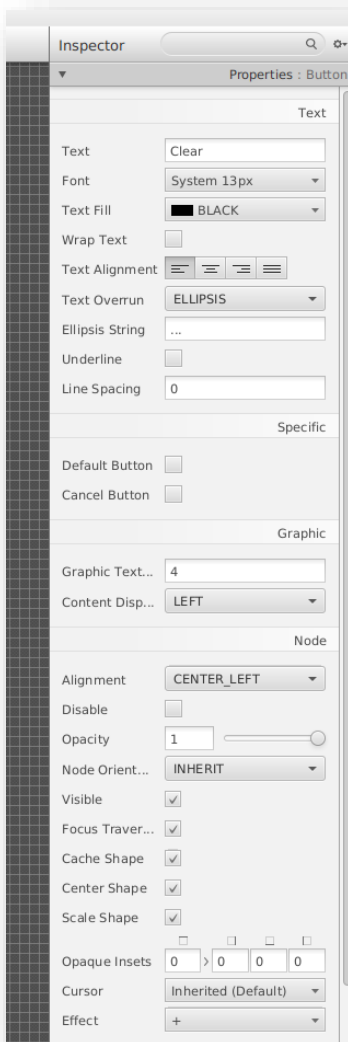
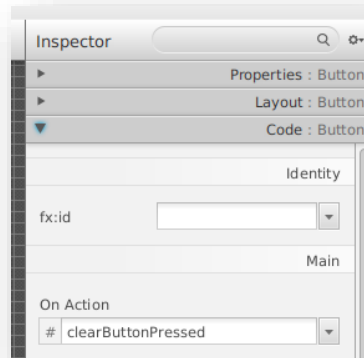
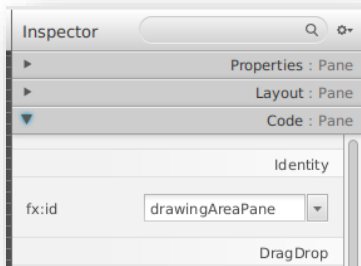
```

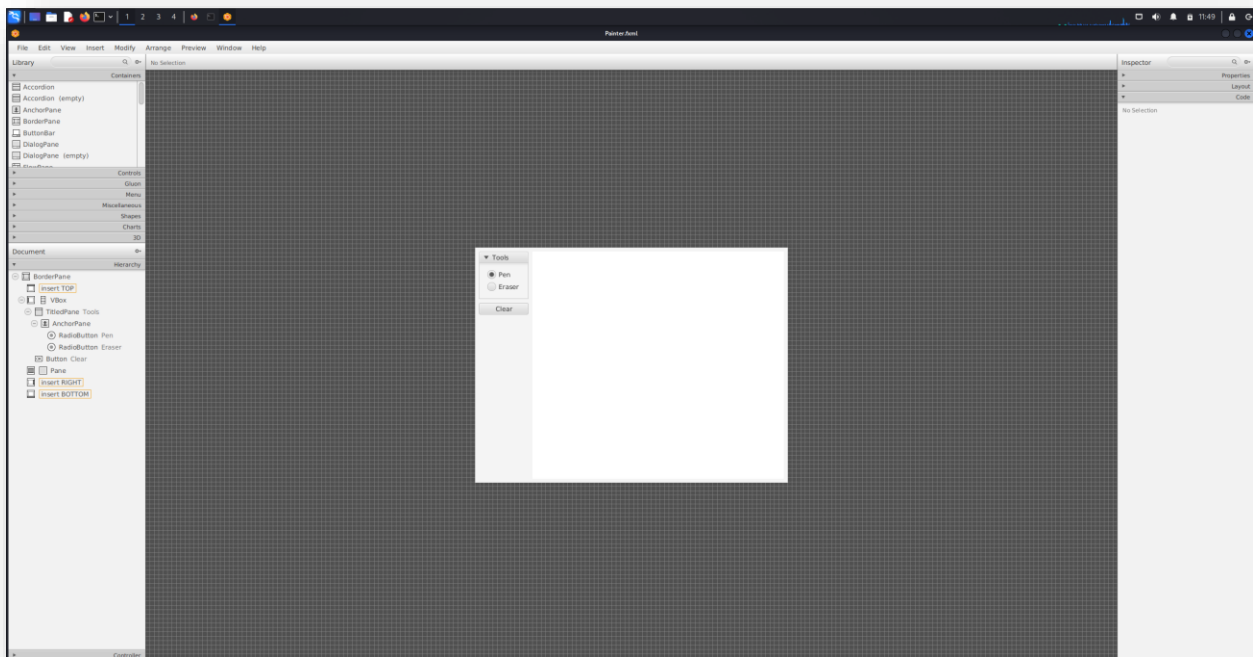
28     JButton addToCartButton = new JButton("Add to cart");
29     addToCartButton.addActionListener(actionPerformed(e) -> {
30         try {
31             String message = cart.addMedia(media);
32             JOptionPane.showMessageDialog(null, message);
33         } catch (LimitExceededException ex) {
34             JOptionPane.showMessageDialog(null, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
35         }
36     });
37
38     container.add(addToCartButton);
39
40
41
42
43
44
45     if (media instanceof Playable) {
46         JButton playButton = new JButton("Play");
47         playButton.addActionListener(actionPerformed(e) -> {
48
49             JDialog dialog = new JDialog();
50             dialog.setTitle(media.getTitle());
51             dialog.setSize(400, 300);
52
53             String mediaInfo = "";
54             try {
55                 mediaInfo = "<html>" + media.playGUI().replace("\n", "<br/>") + "</html>";
56                 JLabel mediaLabel = new JLabel(mediaInfo);
57                 mediaLabel.setVerticalAlignment(JLabel.CENTER);
58                 mediaLabel.setHorizontalAlignment(JLabel.CENTER);
59                 JScrollPane scrollPane = new JScrollPane(mediaLabel);
60                 scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
61
62                 dialog.add(scrollPane);
63                 dialog.setVisible(true);
64             } catch (PlayerException e1) {
65                 JOptionPane.showMessageDialog(null, e1.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
66             }
67
68         });
69         container.add(playButton);
70     }
71
72

```

## 4. JavaFX API







```
GUIProject > src > hust > soict > dsai > javafx > J Painter.java > ...
1 package hust.soict.dsai.javafx;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Parent;
6 import javafx.scene.Scene;
7 import javafx.stage.Stage;
8
9 public class Painter extends Application {
10
11     @Override
12     public void start(Stage stage) throws Exception {
13         Parent root = FXMLLoader.load(getClass()
14             .getResource(name: "/hust/soict/dsai/javafx/Painter.fxml"));
15
16         Scene scene = new Scene(root);
17         stage.setTitle(value: "Painter");
18         stage.setScene(scene);
19         stage.show();
20     }
21     Run | Debug
22     public static void main(String[] args) {
23         launch(args);
24     }
25
26 }
```

```

GUIProject > src > hust > soict > dsai > javafx > J PainterController.java > ...
1 | package hust.soict.dsai.javafx;
2
3 | import java.net.URL;
4 | import java.util.ResourceBundle;
5 | import javafx.event.ActionEvent;
6 | import javafx.fxml.FXML;
7 | import javafx.scene.control.RadioButton;
8 | import javafx.scene.input.MouseEvent;
9 | import javafx.scene.layout.Pane;
10 | import javafx.scene.paint.Color;
11 | import javafx.scene.shape.Circle;
12 | import javafx.scene.shape.Rectangle;
13
14 | public class PainterController {
15
16 |     @FXML
17 |     private RadioButton eraser;
18
19 |     @FXML
20 |     private RadioButton pen;
21
22 |     @FXML
23 |     private ResourceBundle resources;
24
25 |     @FXML
26 |     private URL location;
27
28 |     @FXML
29 |     private Pane drawingAreaPane;
30
31 |     @FXML
32 |     void clearButtonPressed(ActionEvent event) {
33 |         drawingAreaPane.getChildren().clear();
34 |     }
35
36 |     @FXML
37 |     void drawingAreaMouseDragged(MouseEvent event) {
38 |         Rectangle clipArea = new Rectangle(x:0, y:0, drawingAreaPane.getWidth(), drawingAreaPane.getHeight());
39 |         drawingAreaPane.setClip(clipArea);
40 |         Color inkColor = Color.BLACK;
41 |         if (eraser.isSelected()) {
42 |             inkColor = Color.WHITE;
43 |         }
44 |         Circle newCircle = new Circle(event.getX(), event.getY(), radius:4, inkColor);
45 |         drawingAreaPane.getChildren().add(newCircle);
46 |     }
47
48 |     @FXML
49 |     void initialize() {
50 |         assert drawingAreaPane != null : "fx:id=\"drawingAreaPane\" was not injected: check your FXML file 'Painter.fxml'.";
51 |     }
52 | }
53
54 |
55 |

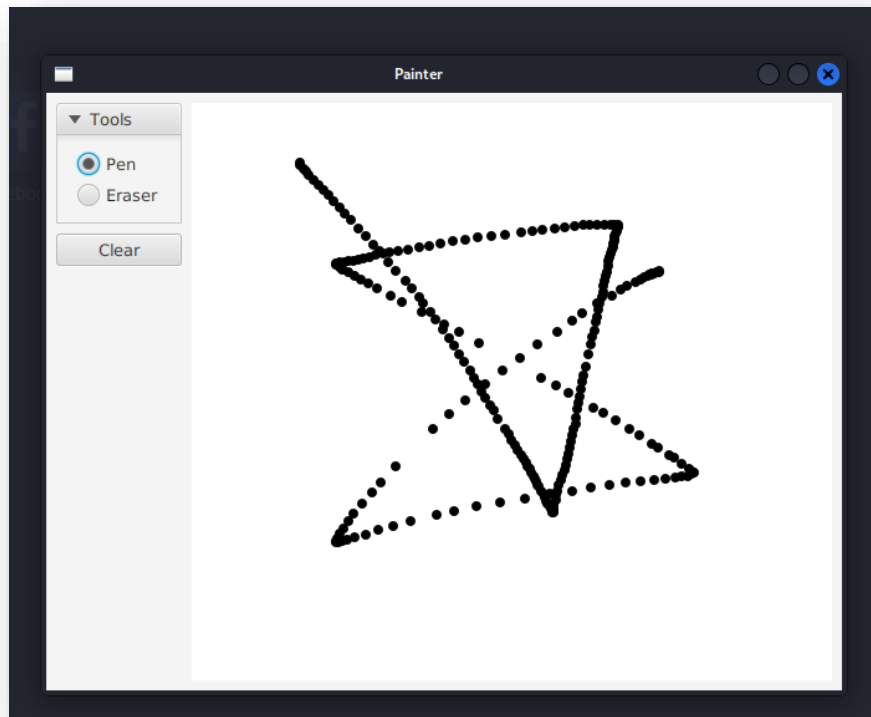
```

```

GUIProject > src > hust > soict > dsai > javafx > % Painter.fxml
1 | <?xml version="1.0" encoding="UTF-8"?>
2
3 | <!--import javafx.geometry.Insets;
4 | import javafx.scene.control.Button;
5 | import javafx.scene.control.RadioButton;
6 | import javafx.scene.control.TogglePane;
7 | import javafx.scene.control.ToggleGroup;
8 | import javafx.scene.layout.AnchorPane;
9 | import javafx.scene.layout.BorderPane;
10 | import javafx.scene.layout.Pane;
11 | import javafx.scene.layout.VBox;
12
13 | <!--
14 | padding
15 | <!-- insets bottom="8.0" left="8.0" right="8.0" top="8.0" /-->
16 | <!-- padding
17 | <!-- left
18 | <!-- VBox maxHeight="1.7976931348623157E308" prefHeight="149.0" prefWidth="181.0" spacing="8.0" BorderPane.alignment="CENTER"
19 | <!-- <BorderPane.margin
20 | <!-- insets right="8.0" /-->
21 | <!-- <BorderPane.margin
22 | <!-- children
23 | <!-- <TitledPane animated="false" prefHeight="97.0" prefWidth="181.0" text="Tools"
24 | <!-- <content
25 | <!-- <AnchorPane maxHeight="0.0" minWidth="0.0" prefHeight="0.0" prefWidth="99.0"
26 | <!-- <children
27 | <!-- <RadioButton fx:id="pen" layoutX="16.0" layoutY="14.0" mnemonicParsing="false" selected="true" text="Pen"
28 | <!-- <toggleGroup
29 | <!-- <ToggleGroup fx:id="tools" /-->
30 | <!-- </toggleGroup
31 | <!-- <RadioButton
32 | <!-- <RadioButton fx:id="eraser" layoutX="16.0" layoutY="39.0" mnemonicParsing="false" text="Eraser" toggleGroup="tools" /-->
33 | <!-- <children
34 | <!-- </AnchorPane
35 | <!-- </content
36 | <!-- <TitledPane
37 | <!-- <Button maxHeight="1.7976931348623157E308" mnemonicParsing="false" onAction="clearButtonPressed" prefWidth="181.0" text="Clear" /-->
38 | <!-- </children
39 | <!-- </VBox
40 | <!-- </left
41 | <!-- <center
42 | <!-- <Pane fx:id="drawingAreaPane" onMouseDragged="drawingAreaMouseDragged" prefHeight="200.0" prefWidth="200.0" style="-fx-background-color: white;" BorderPane.alignment="CENTER" /-->
43 | <!-- </center
44 | <!-- </BorderPane
45 |

```



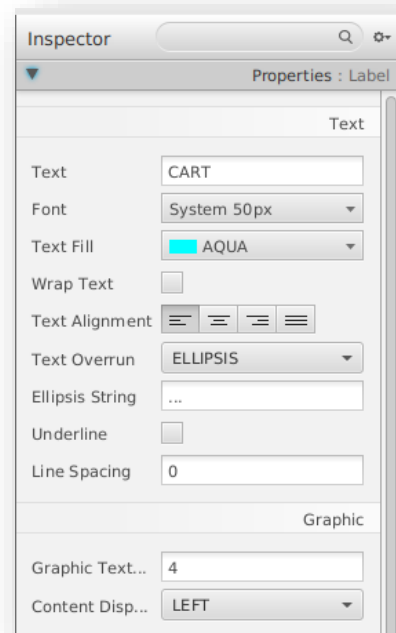
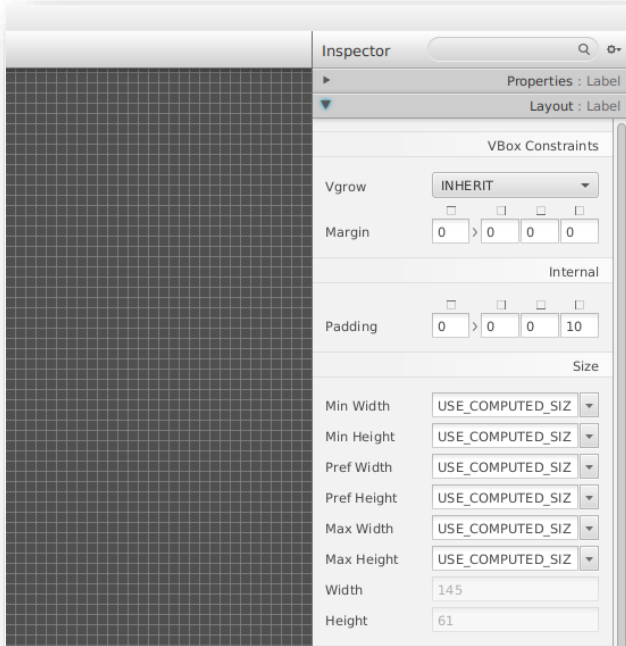
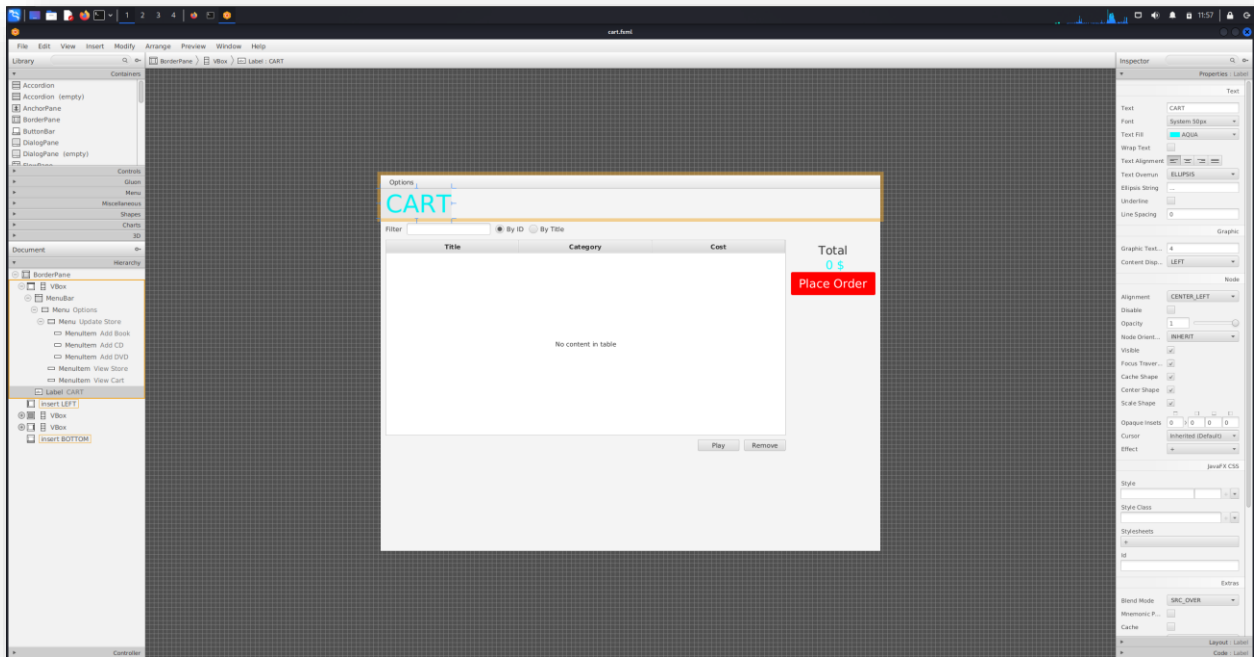


## Handle Erase Function

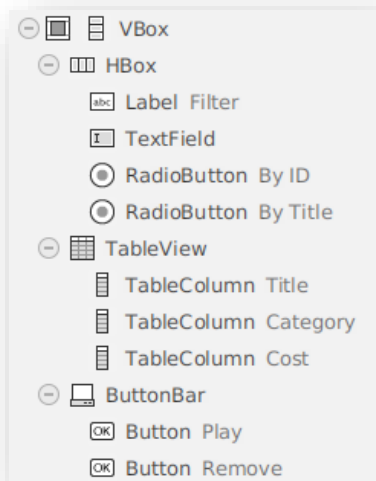
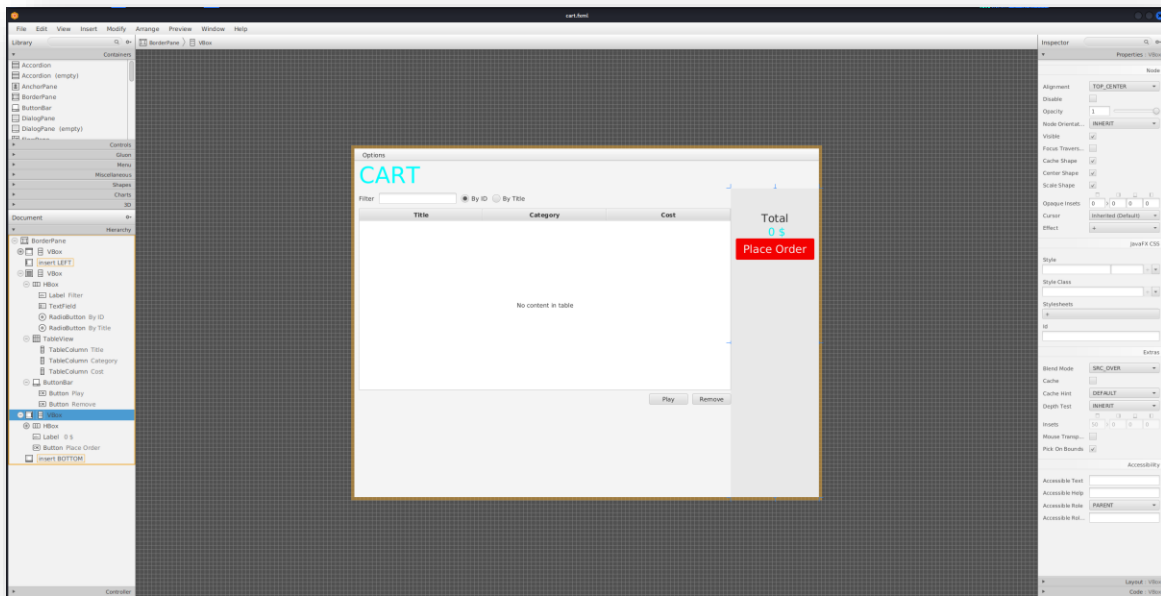
```
@FXML
void drawingAreaMouseDragged(MouseEvent event) {
    Rectangle clipArea = new Rectangle(x:0, y:0, drawingAreaPane.getWidth(), drawingAreaPane.getHeight());
    drawingAreaPane.setClip(clipArea);
    Color inkColor = Color.BLACK;
    if (eraser.isSelected()) {
        inkColor = Color.WHITE;
    }
    Circle newCircle = new Circle(event.getX(), event.getY(), radius:4, inkColor);
    drawingAreaPane.getChildren().add(newCircle);
}
```

## 5 Setting up the View Cart Screen with ScreenBuilder

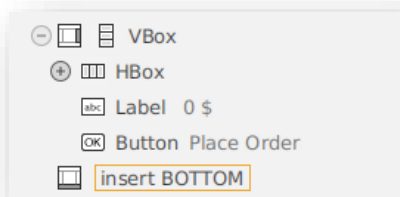
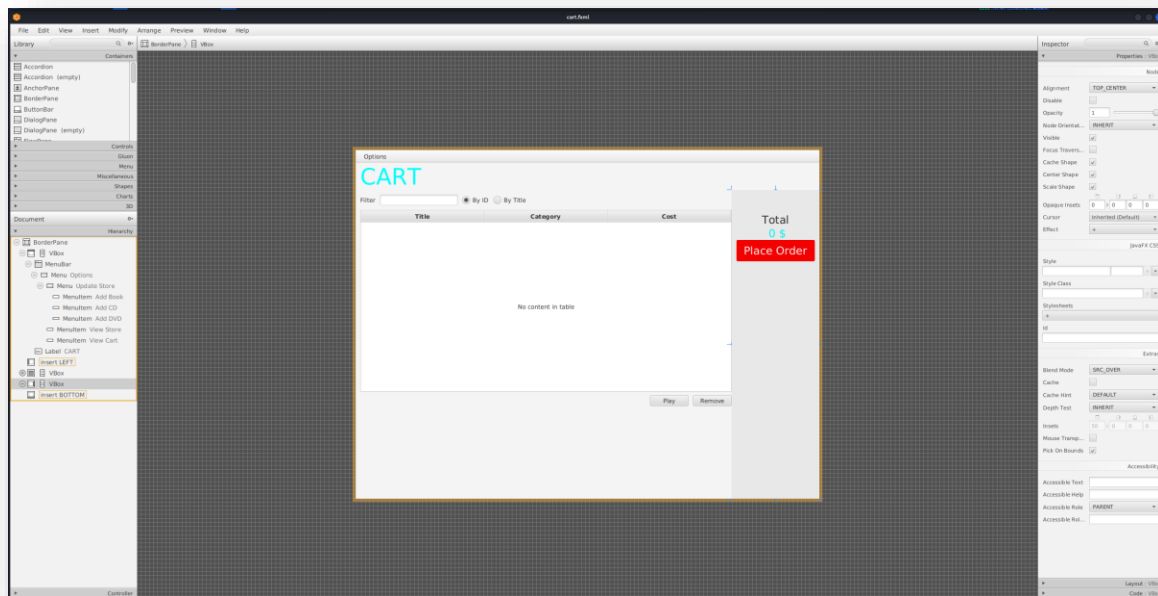
### 5.2 Setting up the TOP area



### 5.3 Setting up the CENTER area



## 5.4 Setting up the RIGHT area



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

```

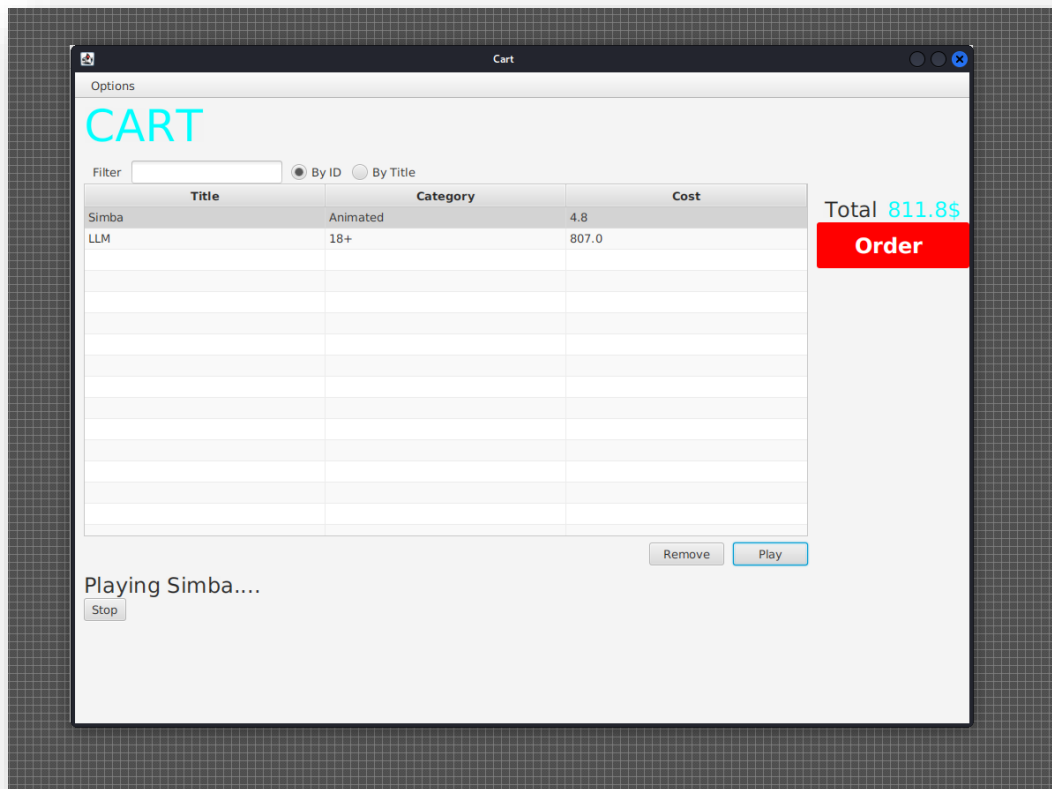
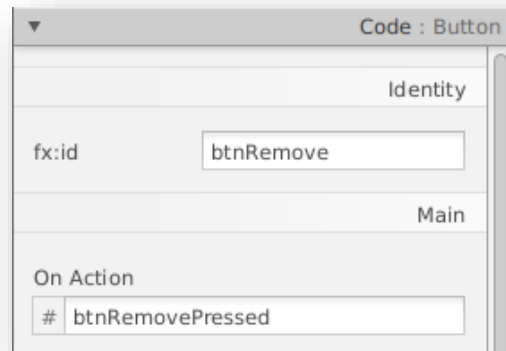
1 package hust.soict.dsai.aims.screen;
2
3 import javax.swing.JFrame;
4 import hust.soict.dsai.aims.screen.controller.CartScreenController;
5 import javafx.application.Platform;
6 import javafx.embed.swing.JFXPanel;
7 import javafx.fxml.FXMLLoader;
8 import javafx.scene.Parent;
9 import javafx.scene.Scene;
10 import hust.soict.dsai.aims.cart.Cart;
11
12 public class CartScreen extends JFrame { 2 usages  ▲ get-wright*
13
14     private static Cart cart; 2 usages
15
16     > public static void main(String[] args) { new CartScreen(cart); }
17
18     public CartScreen(Cart cart) { 1 usage  ▲ get-wright
19
20         super();
21
22         CartScreen.cart = cart;
23
24         JFXPanel fxPanel = new JFXPanel();
25         this.add(fxPanel);
26
27         this.setTitle("Cart");
28         this.setSize(1024, 768);
29         this.setVisible(true);
30         > Platform.runLater(run() → {
31             try {
32                 FXMLLoader loader = new FXMLLoader(getClass().getResource("/hust/soict/dsai/aims/screen/view/cart.fxml"));
33
34                 CartScreenController controller = new CartScreenController(cart);
35                 loader.setController(controller);
36                 Parent root = loader.load();
37                 fxPanel.setScene(new Scene(root));
38             } catch (Exception e) {
39                 e.printStackTrace();
40             }
41         });
42     }
43 }
44
45 }

```

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



## 8. Updating buttons based on selected item in TableView - ChangeListener



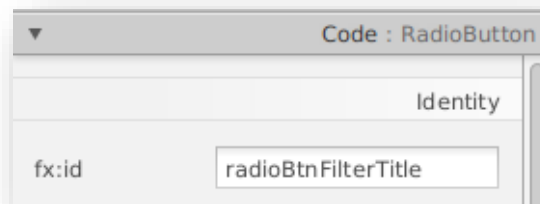
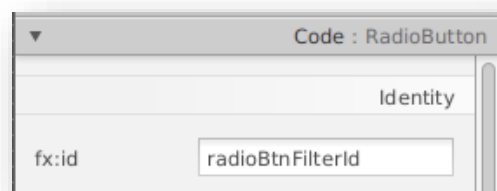
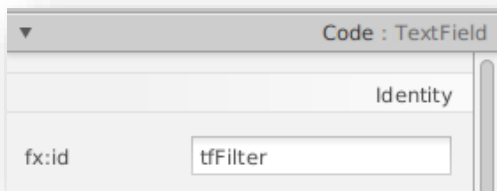
## 9. Deleting a media

```

81     @FXML
82     void btnRemovePressed(ActionEvent event) {
83         Media media = tblMedia.getSelectionModel().getSelectedItem();
84         cart.removeMedia(media);
85         costLabel.setText(cart.totalCost() + " $*");
86     }

```

## 10. Filter items in cart - FilteredList



```

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

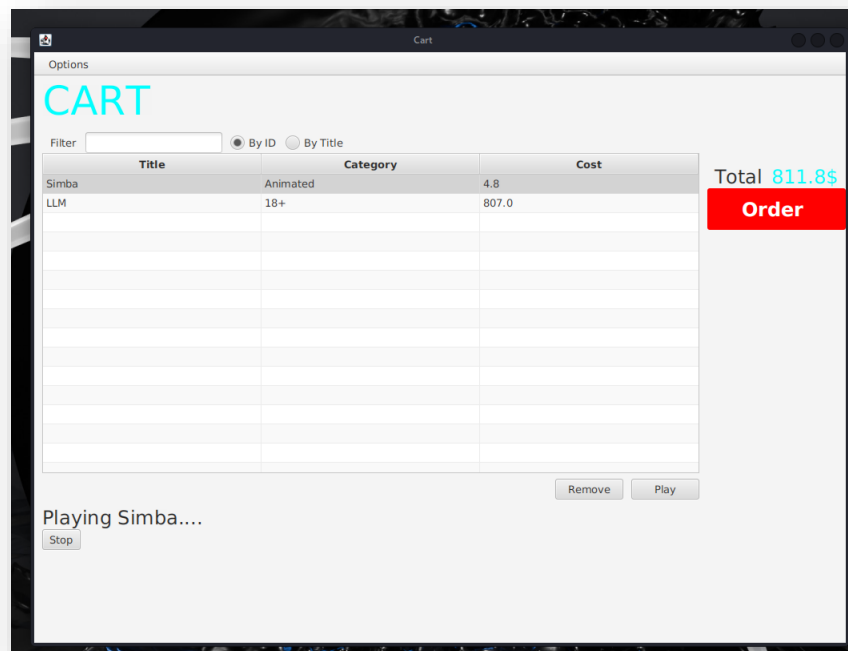
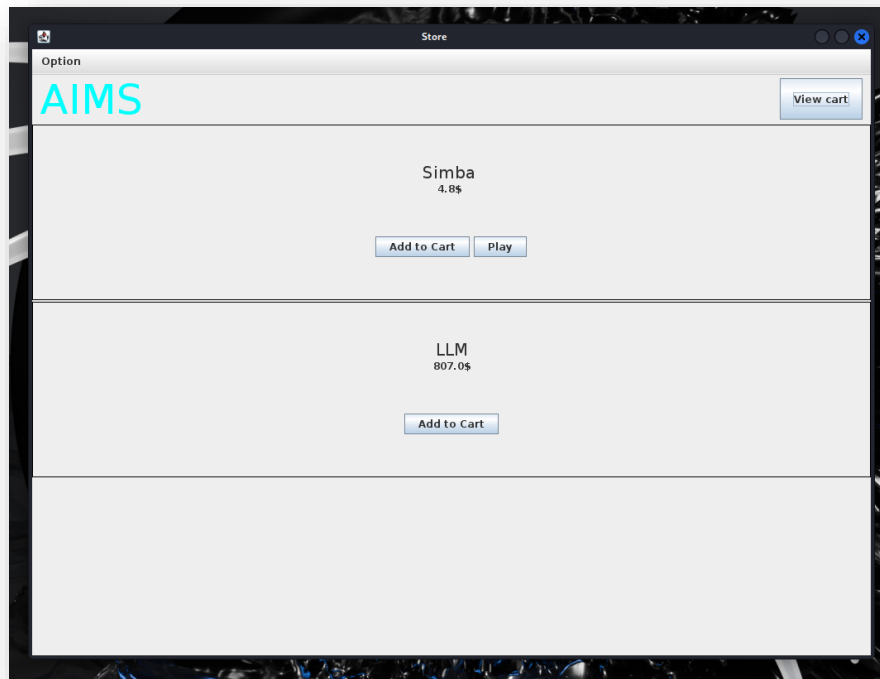
private void showFilteredMedia(String keyword) {
    FilteredList<Media> filteredList = new FilteredList<>(cart.getItemsOrdered());

    if (!keyword.isEmpty() && radioBtnFilterId.isSelected()) {
        filteredList.setPredicate(media -> {
            String idString = String.valueOf(media.getId());
            return idString.equals(keyword);
        });
    } else if (!keyword.isEmpty() && radioBtnFilterTitle.isSelected()) {
        filteredList.setPredicate(media -> {
            String title = media.getTitle().toLowerCase();
            return title.contains(keyword.toLowerCase());
        });
    } else {
        filteredList.setPredicate(null);
    }
    tblMedia.setItems(filteredList);
}
}

```



## 11. Complete the Aims GUI application



AIMS

View cart

title

category

cost

Authors(Names are separated by a comma)

add

AIMS

View cart

title

category

cost

artist

List track (each track separated by a comma Ex: track-length)

add

Option

# AIMS

View cart

title

category

cost

director

length

add

Filter by ID

Cart

Options

# CART

Filter 1 ☒ By ID ☐ By Title

Title	Category	Cost
Simba	Animated	4.8

Total 811.8\$

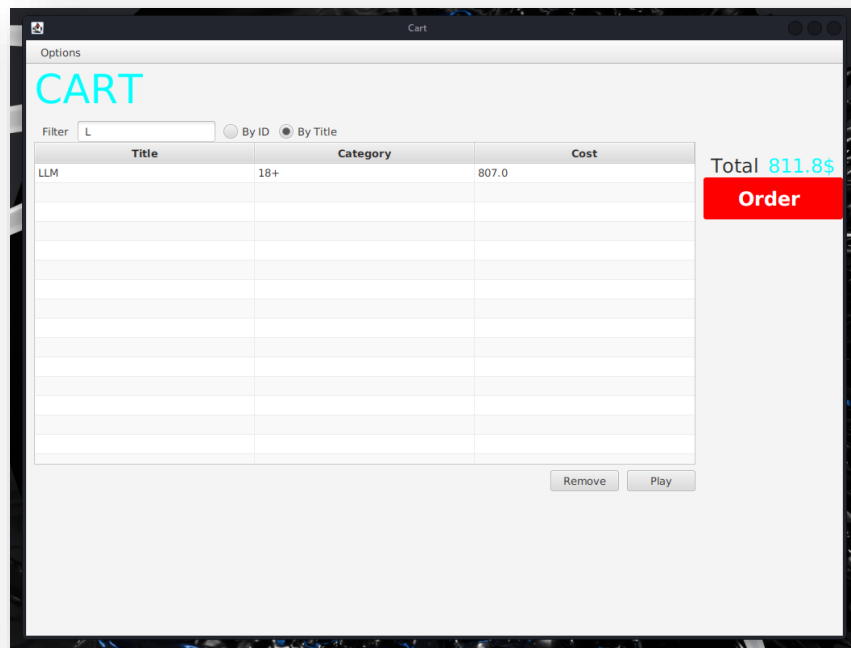
Order

Playing Simba....

Stop

Remove Play

Filter By Title



- AddItemToStoreScreen

```
public class AddItemToStoreScreen extends JFrame {
    protected Stone store;
    protected Cart cart;
    protected JTextField title;
    protected JTextField category;
    protected JTextField cost;
    protected JButton addBtn;
    protected ControllerScreen c;
    protected JPanel center;
    protected int numberInput=1;
    JMenuBar createMenuBar() {
        JMenu menu = new JMenu("Option");

        JMenu smUpdateStore = new JMenu("Update Store");
        JMenuItem addBookMenu= new JMenuItem("Add Book");
        addBookMenu.addActionListener(e->{
            c.showAddBookScreen();
        });
        smUpdateStore.add(addBookMenu);
        JMenuItem addCDMenu=new JMenuItem("Add CD");
        addCDMenu.addActionListener(e->{
            c.showAddCDScreen();
        });
        smUpdateStore.add(addCDMenu);
        JMenuItem addDVDMenu= new JMenuItem("Add DVD");
        addDVDMenu.addActionListener(e->{
            c.showAddDVDScreen();
        });
        smUpdateStore.add(addDVDMenu);

        menu.add(smUpdateStore);
        JMenuItem viewStoreMenu= new JMenuItem("view store");
        viewStoreMenu.addActionListener(e->{
            c.showStoreScreen();
        });
        menu.add(viewStoreMenu);

        JMenuItem viewCartMenu= new JMenuItem("View cart");
        viewCartMenu.addActionListener(e->{
            c.showCartScreen();
        });
        menu.add(viewCartMenu);

        JMenuBar menuBar = new JMenuBar();
        menuBar.setLayout(new FlowLayout(FlowLayout.LEFT));
        menuBar.add(menu);

        return menuBar;
    }
}
```

## - AddDigitalVideoDiscToStoreScreen

```
15 public class AddDigitalVideoDiscToStoreScreen extends AddItemToStoreScreen {
16     private JTextField director;
17     private JTextField length;
18
19     public AddDigitalVideoDiscToStoreScreen(Store store, Cart cart, ControllerScreen c) {
20         super(store, cart, c);
21     }
22
23     @Override
24     void updateAdd(JPanel panel) {
25         this.numberInput = 8;
26         JLabel directorLabel = new JLabel("director", JLabel.TRAILING);
27         panel.add(directorLabel);
28         director = new JTextField(20);
29         director.setPreferredSize(new Dimension(150, 20));
30         directorLabel.setLabelFor(director);
31         panel.add(director);
32         JLabel lengthLabel = new JLabel("length", JLabel.TRAILING);
33         panel.add(lengthLabel);
34         length = new JTextField(20);
35         lengthLabel.setLabelFor(length);
36         panel.add(length);
37         JButton tes = new JButton("add");
38         tes.setVisible(false);
39         panel.add(tes);
40         panel.setPreferredSize(new Dimension(100, 300));
41         addBtn = new JButton("add");
42         addBtn.addActionListener(e -> {
43             addMedia();
44         });
45         panel.add(addBtn);
46     }
47
48     public void addMedia() {
49         String title = this.title.getText();
50         String director = this.director.getText();
51         String category = this.category.getText();
52         Float cost = Float.parseFloat(this.cost.getText());
53         int length = Integer.parseInt(this.length.getText());
54         DigitalVideoDisc dvd = new DigitalVideoDisc(title, category, director, length, cost);
55         this.store.addMedia(dvd);
56         JOptionPane.showMessageDialog(null, "add DVD successfully!");
57         clearTextField();
58     }
59
60     public void clearTextField(){
61         this.title.setText("");
62         this.director.setText("");
63         this.cost.setText("");
64         this.length.setText("");
65         this.category.setText("");
66     }
67
68 }
```

## - AddCompactDiscToStoreScreen

```

16 public class AddCompactDiscToStoreScreen extends AddItemToStoreScreen {
17     private JTextField artist;
18     private JTextField listTrack;
19
20     public AddCompactDiscToStoreScreen(Store store, Cart cart, ControllerScreen c) {
21         super(store, cart, c);
22     }
23
24
25     @Override
26     void updateAdd(JPanel panel) {
27         this.numberInput = 4;
28
29         JLabel artistLabel = new JLabel("artist", JLabel.TRAILING);
30         panel.add(artistLabel);
31         artist = new JTextField(2);
32         artist.setPreferredSize(new Dimension(150, 20));
33         artistLabel.setLabelFor(artist);
34         panel.add(artist);
35         JLabel listTrackLabel = new JLabel("List track (each track separated by a comma Ex: track-length)",
36             JLabel.TRAILING);
37
38         panel.add(listTrackLabel);
39         listTrack = new JTextField(2);
40         listTrackLabel.setLabelFor(listTrack);
41         panel.add(listTrack);
42         JButton tes = new JButton("add");
43         tes.setVisible(false);
44         panel.add(tes);
45         panel.setPreferredSize(new Dimension(100, 300));
46         addBtn = new JButton("add");
47         addBtn.addActionListener(e -> {
48             addMedia();
49         });
50         panel.add(addBtn);
51     }
52
53     public void addMedia() {
54         String title = this.title.getText();
55         String listTrack = this.listTrack.getText();
56         String category = this.category.getText();
57         float cost = Float.parseFloat(this.cost.getText());
58         String artist = (this.artist.getText());
59         String[] arrayTrack = listTrack.trim().split(",");
60
61         CompactDisc cd = new CompactDisc(artist, title, category, "", cost);
62         ;
63         for (String track : arrayTrack) {
64             String titleTrack = track.split("-")[0].trim();
65             int lengthTrack = Integer.parseInt(track.split("-")[1].trim());
66             Track newTrack = new Track(titleTrack, lengthTrack);
67             cd.addTrack(newTrack);

```

- AddBookToStoreScreen

```

public class AddBookToStoreScreen extends AddItemToStoreScreen {
    private JTextField listAuthor;

    public AddBookToStoreScreen(Store store, Cart cart, ControllerScreen c) {
        super(store, cart, c);
    }

    @Override
    void updateAdd(JPanel panel) {
        this.numberInput = 5;

        JLabel listAuthorLabel = new JLabel("Authors(Names are separated by a comma)", JLabel.TRAILING);
        panel.add(listAuthorLabel);
        listAuthor = new JTextField(20);
        listAuthor.setPreferredSize(new Dimension(150, 20));
        listAuthorLabel.setLabelFor(listAuthor);
        panel.add(listAuthor);

        JButton tes = new JButton("add");
        tes.setVisible(false);
        panel.add(tes);
        panel.setPreferredSize(new Dimension(100, 300));
        addBtn = new JButton("add");
        addBtn.addActionListener(e -> {
            addMediaToStore();
        });
        panel.add(addBtn);
    }

    public void addMediaToStore() {
        String title = this.title.getText();
        String listAuthor = this.listAuthor.getText();
        String[] arrayAuthor = listAuthor.split(",");
        String category = this.category.getText();
        float cost = Float.parseFloat(this.cost.getText());
        Book book = new Book(title, category, cost);
        for(String author:arrayAuthor) {
            book.addAuthor(author);
        }
        this.store.addMedia(book);
        JOptionPane.showMessageDialog(null, "add Book successfully!");
        clearTextField();
    }

    public void clearTextField(){
        this.title.setText("");
        this.listAuthor.setText("");
        this.cost.setText("");
        this.category.setText("");
    }
}

```

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

```

31     public void addMedia(Media media) throws LimitExceededException {
32         if ((itemsOrdered.size()) >= MAX_ORDERED) {
33             throw new LimitExceededException("Full");
34         }
35         else if (itemsOrdered.contains(media)) {
36             System.out.println("This is already in your order!");
37         }
38         else {
39             itemsOrdered.add(media);
40             System.out.println("Media added!");
41         }
42     }

```

13. Create a class which inherits from Exception

```

package hust.soict.dsai.aims.exception;

public class PlayerException extends RuntimeException {

    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 writableStackTrace) {
        super(message, cause, enableSuppression, writableStackTrace);
        // TODO Auto-generated constructor stub
    }
}

```

## - CompactDisc

```

@Override
public void play() throws PlayerException{
    System.out.println("CompactDisc Artist: " + this.getArtist());
    System.out.println("Total length: " + this.getLength());

    if (this.getLength() > 0) {
        System.out.println("Compactdisc: " + this.getTitle());
        System.out.println("CompactDisc Artist: " + this.getArtist());
        System.out.println("Total length: " + this.getLength());
        java.util.Iterator iter = tracks.iterator();
        Track nextTrack;
        while (iter.hasNext()) {
            nextTrack = (Track) iter.next();
            try {
                nextTrack.play();
            }
            catch(PlayerException e ) {
                throw e;
            }
        }
    }
    else {
        throw new PlayerException("Error: CD length is non-positive!");
    }
}

```

## - DVD



```

@Override
public void play() throws PlayerException{
    if (this.getLength() > 0) {
        System.out.println("Playing DVD: " + this.getTitle());
        System.out.println("DVD length: " + this.getLength());
    }
    else {throw new PlayerException("Error: DVD length is non-positive!");}
}

```

## - Track

```

@Override
public void play() throws PlayerException {
    if (this.getLength() > 0) {
        System.out.println("Playing track: " + this.getTitle());
        System.out.println("Track length: " + this.getLength());
    }
    else {throw new PlayerException("Error: Track length is non-positive!");}
}

```

## Updated Aim class

```

switch (choice) {
    case 1:
        cart.addMedia(media);
        System.out.println("Media added to cart.");
        break;
    case 2:
        if (media instanceof Playable) {
            if (media instanceof DigitalVideoDisc){
                try {
                    ((DigitalVideoDisc) media).play();
                } catch (PlayerException e) {
                    // TODO Auto-generated catch block
                    e.getMessage();
                    e.toString();
                    e.printStackTrace();
                }
            }
            else if (media instanceof CompactDisc){
                try {
                    ((CompactDisc) media).play();
                } catch (PlayerException e) {
                    // TODO Auto-generated catch block
                    e.getMessage();
                    e.toString();
                    e.printStackTrace();
                }
            }
        }
        else {
            System.out.println("This media cannot be played.");
        }
        break;
    case 0:
        break;
    default:
        System.out.println("Invalid choice! Please choose a number between 0-2.");
}
} while (choice != 0);
}

```

```

public static void playMedia(Scanner scanner) {
    System.out.print("Enter the title of the media to play: ");
    String title = scanner.nextLine();
    Media media = store.searchByTitle(title);

    if (media != null) {
        if (media instanceof Playable) {
            if (media instanceof DigitalVideoDisc) {
                try {
                    ((DigitalVideoDisc) media).play();
                } catch (PlayerException e) {
                    // TODO Auto-generated catch block
                    e.getMessage();
                    e.toString();
                    e.printStackTrace();
                }
            }
            else if (media instanceof CompactDisc) {
                try {
                    ((CompactDisc) media).play();
                } catch (PlayerException e) {
                    // TODO Auto-generated catch block
                    e.getMessage();
                    e.toString();
                    e.printStackTrace();
                }
            }
        } else {
            System.out.println("This media cannot be played.");
        }
    } else {
        System.out.println("Media not found in the store.");
    }
}

```

- Modified equals() method of Media class

```

@Override new *
public boolean equals(Object o) {
    if (o instanceof Media) {
        Media media = (Media) o;
        if (this.title.equals(media.title)) {
            return true;
        } else {
            return false;
        }
    }
    return false;
}

```

**Update Aims class diagram**



