BÁO CÁO THỰC HÀNH LAB 5
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Mục lục

Mục lục ảnh

1. Swing components

2. Organizing Swing components with Layout Managers

2.1. Create class NumberGrid

```
8   public class NumberGrid extends JFrame {
        6 usages
9       private JButton[] btnNumbers = new JButton[10];
        3 usages
10      private JButton btnDelete, btnReset;
        5 usages
11      private JTextField tfDisplay;
12
        no usages  new *
13      public  NumberGrid(){
14          tfDisplay = new JTextField();
15          tfDisplay.setComponentOrientation(
16                  ComponentOrientation.RIGHT_TO_LEFT);
17          JPanel panelButtons = new JPanel(new GridLayout( rows: 4,  cols: 3));
18          addButtons(panelButtons);
19
20          Container cp = getContentPane();
21          cp.setLayout(new BorderLayout());
22          cp.add(tfDisplay, BorderLayout.NORTH);
23          cp.add(panelButtons, BorderLayout.CENTER);
24
25          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26          setTitle("Number Grid");
27          setSize( width: 200,  height: 200);
28          setVisible(true);
29      }
```

Figure 7: NumberGrid source code(1)

2.2. Adding buttons

```
31      void addButtons(JPanel panelButtons){
32
33          ButtonListener btnListener = new ButtonListener();
34          for(int i = 1; i<=9; i++){
35              btnNumbers[i] = new  JButton( text: ""+i);
36              panelButtons.add(btnNumbers[i]);
37              btnNumbers[i].addActionListener(btnListener);
38          }
39
40          btnDelete = new JButton( text: "DELL");
41          panelButtons.add(btnDelete);
42          btnDelete.addActionListener(btnListener);
43
44          btnNumbers[0] = new JButton( text: "0");
45          panelButtons.add(btnNumbers[0]);
46          btnNumbers[0].addActionListener(btnListener);
47
48          btnReset = new JButton( text: "C");
49          panelButtons.add(btnReset);
50          btnReset.addActionListener(btnListener);
51      }
52
```

Figure 8: NumberGrid source code(2)

2.3. Complete inner class ButtonListener

```
53      private class ButtonListener implements ActionListener{
            new *
54          public void actionPerformed(ActionEvent e){
55              String button = e.getActionCommand();
56              if(button.charAt(0) >= '0' && button.charAt(0) <= '9'){
57                  tfDisplay.setText(tfDisplay.getText() + button);
58              }
59              else if (button.equals("DEL")){
60                  String currentText = tfDisplay.getText();
61                  String text = currentText.substring(0, currentText.length() - 1);
62                  tfDisplay.setText(text);
63              }
64              else {
65                  tfDisplay.setText("");
66              }
67          }
68      }
```

Figure 9: NumberGrid source code(3)

2.3. Demo

```
70      public static void main(String[] args) {
71          new NumberGrid();
72      }
73  }
```

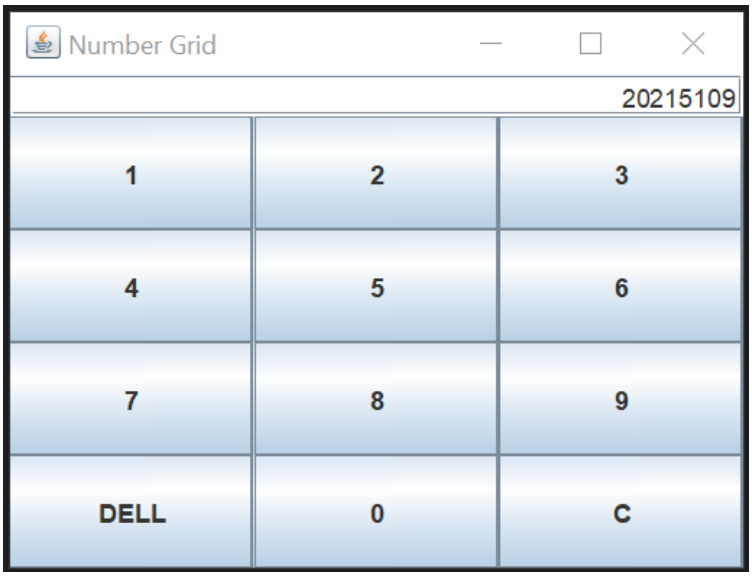

Figure 10: Number Gird

3. Create a graphical user interface for AIMS with Swing

3.1. View Store Screen

3.1.1. Create the StoreScreen class

```
7   public class StoreScreen extends JFrame {
        no usages
8       private Store store;
```

Figure 11: Declaration of StoreScreen class

3.1.2. The NORTH component

```
13      JPanel createNorth(){
14          JPanel north = new JPanel();
15          north.setLayout(new BoxLayout(north, BoxLayout.Y_AXIS));
16          north.add(createMenuBar());
17          north.add(createHeader());
18          return north;
19      }
20
```

Figure 12: createNorth() source code

```
21      JMenuBar createMenuBar(){
22          JMenu menu = new JMenu( s: "Options");
23          JMenu smUpdateStore = new JMenu( s: "Update Store");
24          smUpdateStore.add(new JMenuItem( text: "Add Book"));
25          smUpdateStore.add(new JMenuItem( text: "Add CD"));
26          smUpdateStore.add(new JMenuItem( text: "Add DVD"));
27          menu.add(smUpdateStore);
28          menu.add(new JMenuItem( text: "View store"));
29          menu.add(new JMenuItem( text: "View cart"));
30          JMenuBar menuBar = new JMenuBar();
31          menuBar.setLayout(new FlowLayout(FlowLayout.LEFT));
32          menuBar.add(menu);
33          return menuBar;
34      }
35
```

Figure 13: createMenuBar() source code

```
36      JPanel createHeader(){
37          JPanel header = new JPanel();
38          header.setLayout(new BoxLayout(header,BoxLayout.X_AXIS));
39          JLabel title = new JLabel( text: "AIMS");
40          title.setFont(new Font(title.getFont().getName(), Font.PLAIN, size: 50));
41          title.setForeground(Color.CYAN);
42          JButton cart = new JButton( text: "View cart");
43          cart.setPreferredSize(new Dimension( width: 100, height: 50));
44          cart.setMaximumSize(new Dimension( width: 100, height: 50));
45          header.add(Box.createRigidArea(new Dimension( width: 10, height: 10)));
46          header.add(title);
47          header.add(Box.createHorizontalGlue());
48          header.add(cart);
49          header.add(Box.createRigidArea(new Dimension( width: 10, height: 10)));
50          return header;
51      }
52
```

Figure 14: createHeader() source code

3.1.3. The CENTER component

```
53      JPanel createCenter(){
54          JPanel center = new JPanel();
55          center.setLayout(new GridLayout( rows: 3, cols: 3, hgap: 2, vgap: 2));
56          ArrayList<Media> mediaInStore = store.getItemInStore();
57          for (int i=0; i < 9; i++){
58              MediaStore cell = new MediaStore(mediaInStore.get(i));
59              center.add(cell);
60          }
61          return center;
62      }
63
```

Figure 15: createCenter() source code

3.1.4. The MediaStore class

```
64      public class MediaStore extends JPanel{
            1 usage
65          private Media media;
            1 usage  new *
66  @       public MediaStore(Media media){
67              this.media = media;
68              this.setLayout(new BoxLayout( target: this,BoxLayout.Y_AXIS));
69              JLabel title = new JLabel(media.getTitle());
70              title.setFont(new Font(title.getFont().getName(), Font.PLAIN, size: 20));
71              title.setAlignmentX(CENTER_ALIGNMENT);
72              JLabel cost = new JLabel( text: ""+media.getCost()+" $");
73              cost.setAlignmentX(CENTER_ALIGNMENT);
74              JPanel container = new JPanel();
75              container.setLayout(new FlowLayout(FlowLayout.CENTER));
76              container.add(new JButton( text: "Add to cart"));
77              if(media instanceof IPlayable){
78                  container.add(new JButton( text: "Play"));
79              }
80              this.add(Box.createVerticalGlue());
81              this.add(title);
82              this.add(cost);
83              this.add(Box.createVerticalGlue());
84              this.add(container);
85              this.setBorder(BorderFactory.createLineBorder(Color.BLACK));
86          }
87      }
```

Figure 16: MediaStore source code

3.1.5. Putting it all together

```
    no usages  new *
    public StoreScreen(Store store){
        this.store = store;
        Container cp = getContentPane();
        cp.setLayout(new BorderLayout());

        cp.add(createNorth(), BorderLayout.NORTH);
        cp.add(createCenter(), BorderLayout.CENTER);

        setVisible(true);
        setTitle("Store");
        setSize( width: 1024,  height: 768);
    }
}
```

Figure 17: StoreScreen constructor source code

3.2. Adding more user interaction

```
package hust.soict.aims.store;

import hust.soict.aims.cart.Cart;
import hust.soict.aims.media.Dics;
import hust.soict.aims.media.IPlayable;
import hust.soict.aims.media.Media;

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

5 usages  new *
public class MediaStore extends JPanel {
    1 usage
    private Media media;
    3 usages
    private static Cart cart;
    1 usage  new *
    public MediaStore(Media media){
        this.media = media;
        this.setLayout(new BoxLayout( target: 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( text: "" + media.getCost() + " $");
```

Figure 18: MediaStore lass source code(1)

```
23          cost.setAlignmentX(CENTER_ALIGNMENT);
24          JPanel container = new JPanel();
25          container.setLayout(new FlowLayout(FlowLayout.CENTER));
26          ButtonListener buttonListener = new  ButtonListener(media);
27          JButton addButton = new JButton( text: "Add to cart");
28          container.add(addButton);
29          addButton.addActionListener(buttonListener);
30          if(media instanceof IPlayable){
31              JButton playButton = new JButton( text: "Play");
32              container.add(addButton);
33              playButton.addActionListener(buttonListener);
34          }
35          this.add(Box.createVerticalGlue());
36          this.add(title);
37          this.add(cost);
38          this.add(Box.createVerticalGlue());
39          this.add(container);
40          this.setBorder(BorderFactory.createLineBorder(Color.BLACK));
41      }
42
            no usages   new *
43      public Cart getCart() {
44          return cart;
45      }
46
```

Figure 19: MediaStore lass source code(2)

```
47      public static void setCart(Cart cart) {
48          MediaStore.cart = cart;
49      }
50
            2 usages   new *
51      private class ButtonListener implements ActionListener{
            3 usages
52          private  Media media;
            1 usage   new *
53          public ButtonListener(Media media){
54              super();
55              this.media = media;
56          }
            new *
57          public  void  actionPerformed(ActionEvent e){
58              String button = e.getActionCommand();
59              if (button.equals("Add to cart")){
60                  cart.addMedia(media);
61              }else if (media instanceof Dics dics){
62                  dics.play();
63              }
64          }
65      }
66  }
```

Figure 20: MediaStore lass source code(3)

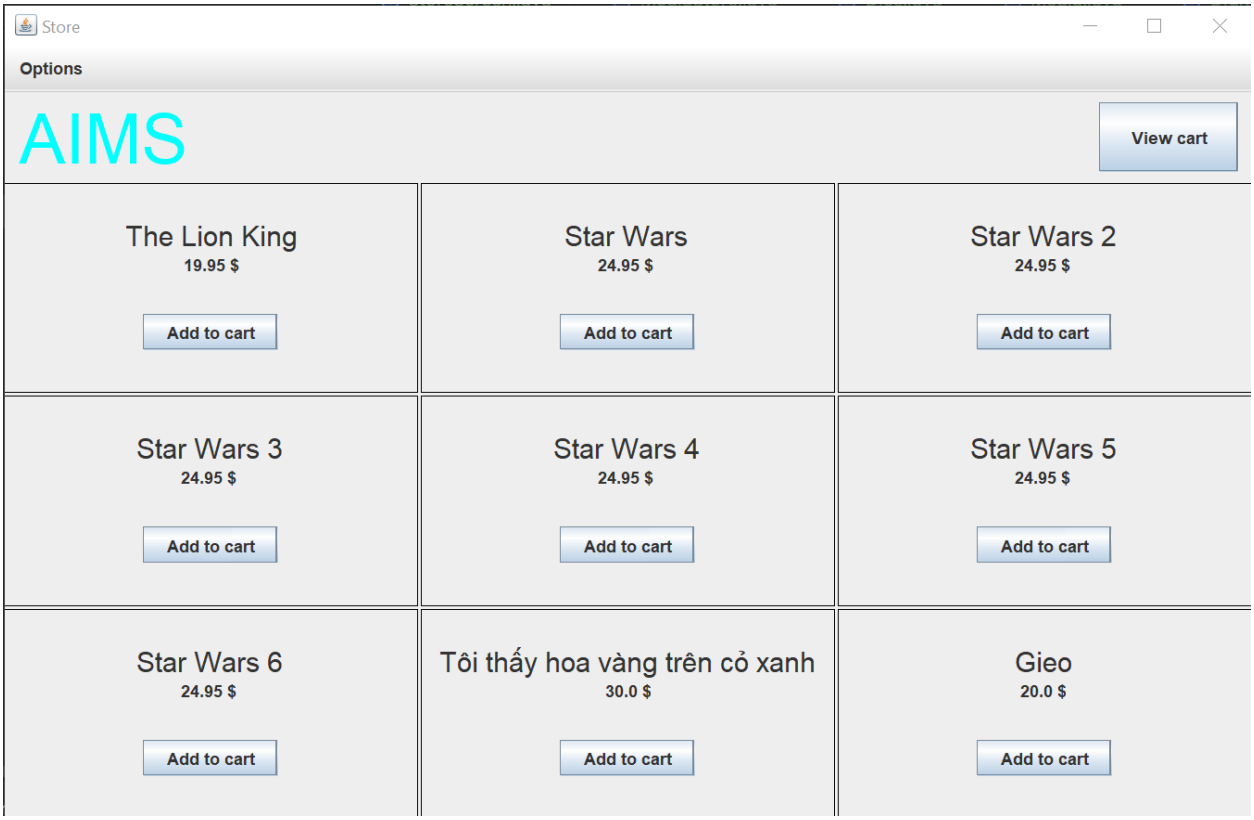

Figure 21: View Store Screen

4. JavaFX API

4.1. Create the FXML file

Step 1. Configuring the BorderPane – the root element of the scene

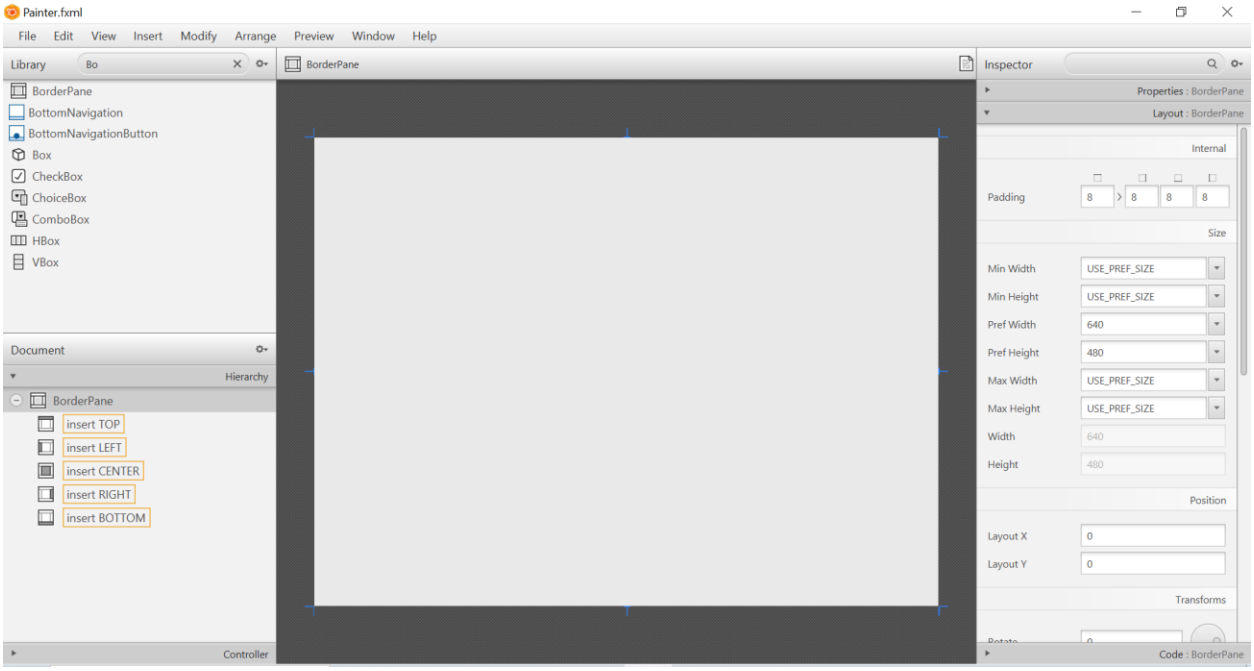

Figure 22: Configuring the BorderPane

Step 2. Adding the Vbox

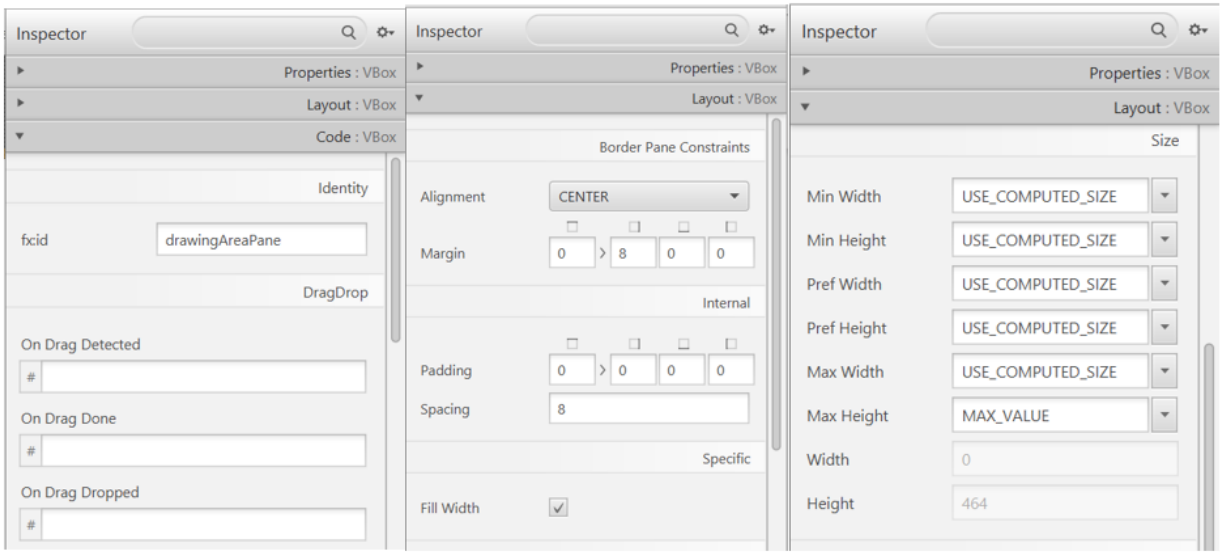

Figure 23: Configuring the VBox

Step 3. Adding the Pane

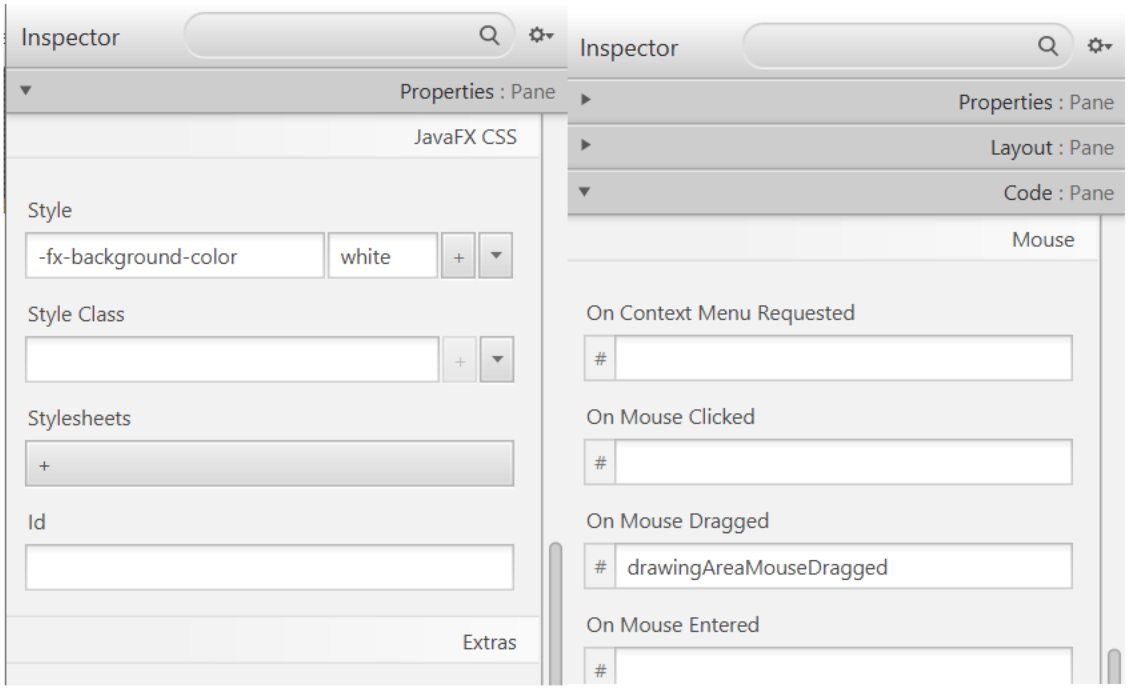

Figure 24: Configuring the Pane

Step 4. Adding the Button

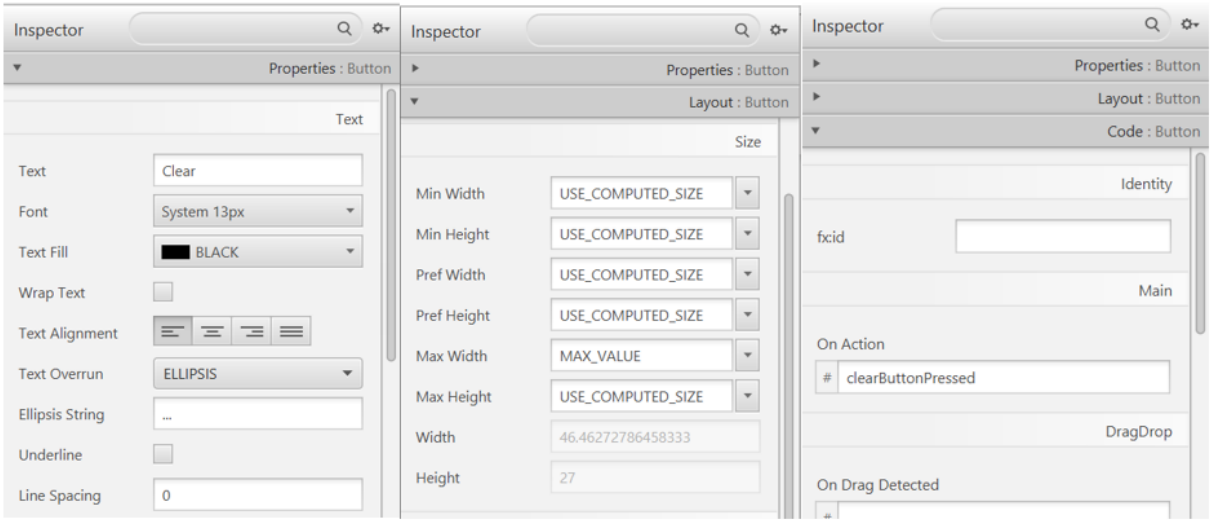

Figure 25: Configuring the Button

Preview:

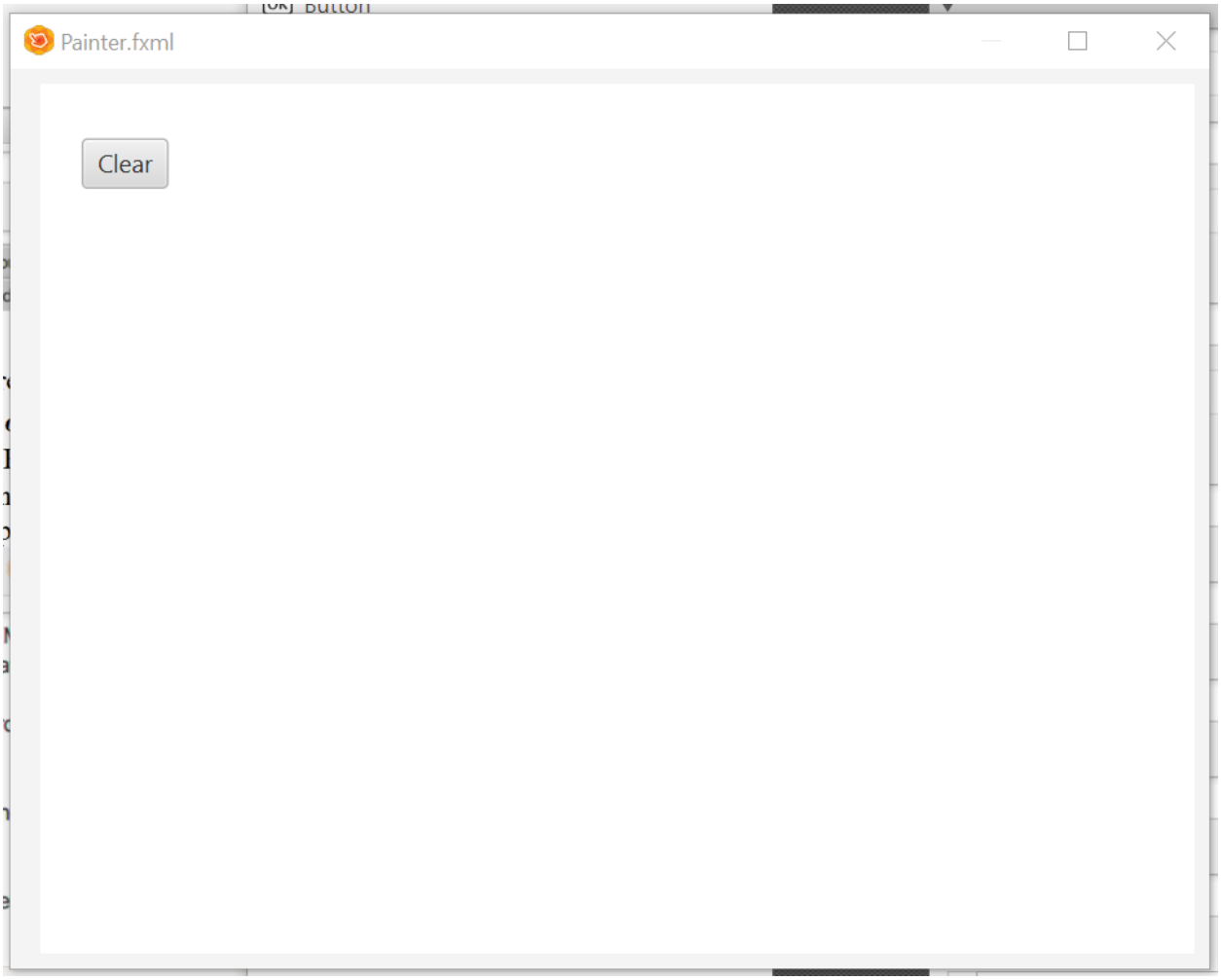

Figure 26: Preview in Window

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
            minWidth="-Infinity" prefHeight="480.0"
            prefWidth="640.0" xmlns="http://javafx.com/javafx/21"
            xmlns:fx="http://javafx.com/fxml/1" fx:controller="hust.soict.javafx.PainterController">
   <padding>
      <Insets bottom="8.0" left="8.0" right="8.0" top="8.0" />
   </padding>
   <left>
      <VBox maxHeight="1.7976931348623157E308" spacing="8.0" BorderPane.alignment="CENTER">
         <BorderPane.margin>
            <Insets right="8.0" />
         </BorderPane.margin>
         <children>
            <Button maxWidth="1.7976931348623157E308" mnemonicParsing="false"
```

Figure 27 : Source code of Painter.fxml (2)

```
                    onAction="#clearButtonPressed" text="Clear">
                  <font>
                     <Font size="13.0" />
                  </font>
               </Button>
            </children>
         </VBox>
      </left>
      <center>
         <Pane fx:id="drawingAreaPane" onMouseDragged="#drawingAreaMouseDragged"
               prefHeight="200.0" prefWidth="200.0" style="-fx-background-color: white;"
               BorderPane.alignment="CENTER" />
      </center>
   </BorderPane>
```

Figure 28: Source code of Painter.fxml (3)

4.2. Create the controller class

```java
package hust.soict.javafx;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.VBox;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.Pane;

1 usage  new *
public class PainterController {

    @FXML
    private Pane drawingAreaPane;
    new *
    @FXML
    void clearButtonPressed(ActionEvent event) {}
    new *
    @FXML
    void drawingAreaMouseDragged(MouseEvent event) {}
}
```

Figure 29: Source code of PainterController Class

4.3. Create the application

```
1  package hust.soict.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
   new *
9  public class Painter extends Application {
       new *
10     public void start(Stage stage) throws Exception {
11         Parent root = FXMLLoader.load(getClass()
12                 .getResource( name: "hust/soict/javafx/Painter.fxml"));
13         Scene scene = new Scene(root);
14         stage.setTitle("Painter");
15         stage.setScene(scene);
16         stage.show();
17     }
18
       new *
19     public static void main(String[] args) {
20         launch(args);
21     }
22 }
```

Figure 30: Painter source code

4.4. Practice exercise

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.RadioButton?>
<?import javafx.scene.control.TitledPane?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="480.0"
            prefWidth="640.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="hust.soict.javafx.PainterController">
   <padding>
      <Insets bottom="8.0" left="8.0" right="8.0" top="8.0" />
   </padding>
   <left>
      <VBox maxHeight="1.7976931348623157E308" spacing="8.0" BorderPane.alignment="CENTER">
         <BorderPane.margin>
            <Insets right="8.0" />
         </BorderPane.margin>
```

Figure 31: Source code of Painter.fxml update (1)

```
         <children>
            <TitledPane alignment="CENTER" animated="false" text="Tools">
              <content>
                <AnchorPane>
                     <children>
                        <RadioButton alignment="CENTER_LEFT" layoutX="17.0" layoutY="14.0"
                                     mnemonicParsing="false" text="Pen">
                           <padding>
                              <Insets bottom="8.0" left="8.0" right="8.0" top="8.0" />
                           </padding>
                        </RadioButton>
                        <RadioButton layoutX="17.0" layoutY="47.0" mnemonicParsing="false" text="Eraser">
                           <padding>
                              <Insets bottom="8.0" left="8.0" right="8.0" top="8.0" />
                           </padding>
                        </RadioButton>
                     </children>
                  </AnchorPane>
              </content>
               <padding>
                  <Insets bottom="4.0" left="4.0" right="4.0" top="4.0" />
               </padding>
            </TitledPane>
```

Figure 32: Source code of Painter.fxml update (2)

```
47              <Button maxWidth="1.7976931348623157E308" mnemonicParsing="false" onAction="#clearButtonPressed" text="Clear">
48                 <font>
49                    <Font size="13.0" />
50                 </font>
51              </Button>
52           </children>
53        </VBox>
54     </left>
55     <center>
56        <Pane fx:id="drawingAreaPane" onMouseDragged="#drawingAreaMouseDragged" prefHeight="200.0" prefWidth="200.0"
57              style="-fx-background-color: white;" BorderPane.alignment="CENTER" />
58     </center>
59  </BorderPane>
```

Figure 33: Source code of Painter.fxml update (3)

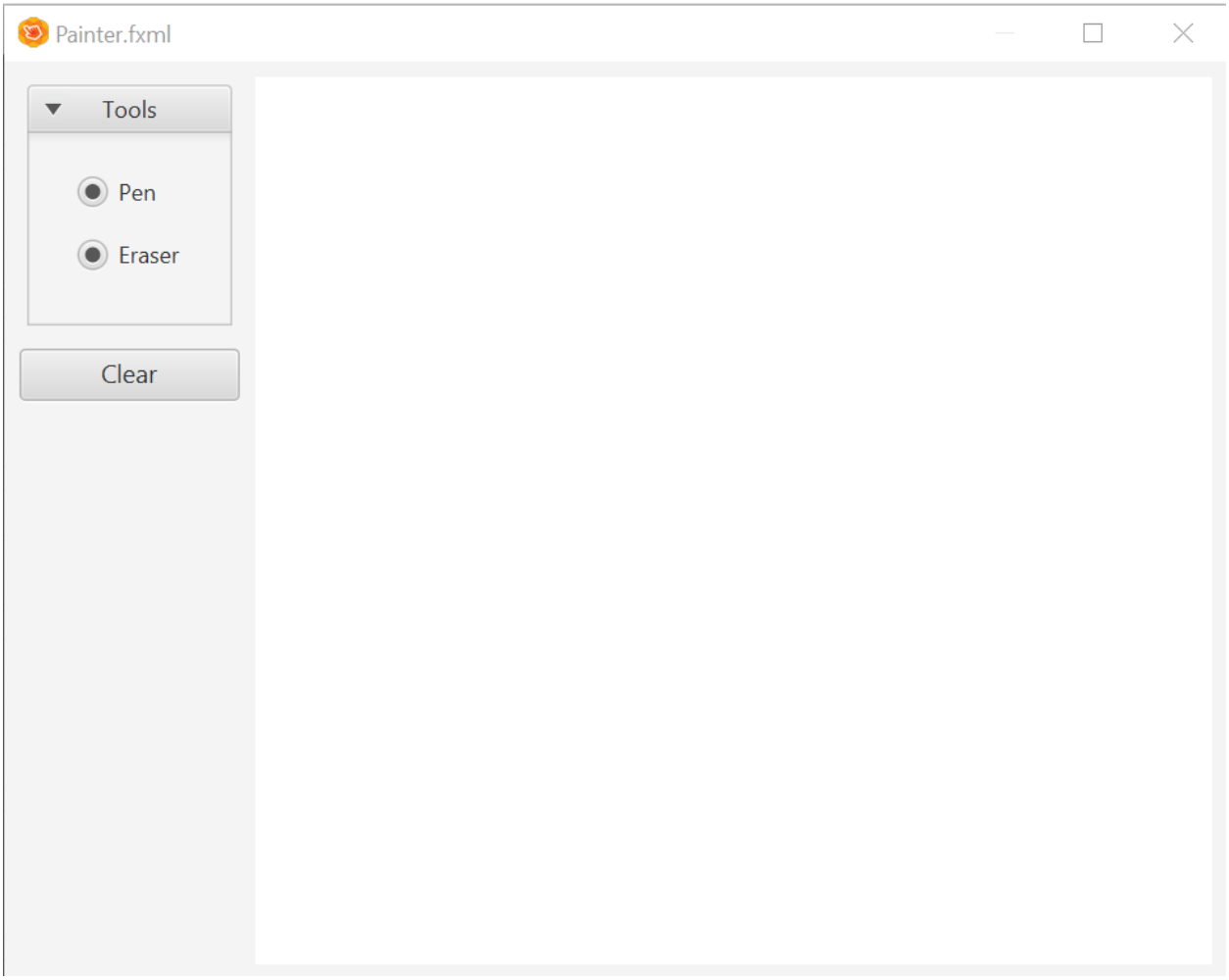

Figure 34: Painter with Eraser

5. Setting up the View Cart Screen with ScreenBuilder

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ButtonBar?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.RadioButton?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>


<AnchorPane prefHeight="768.0" prefWidth="1024.0" xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1
   <children>
      <BorderPane layoutX="3.0" layoutY="3.0" prefHeight="768.0" prefWidth="1024.0">
         <top>
            <VBox prefWidth="100.0" BorderPane.alignment="CENTER">
               <children>
                  <MenuBar>
                     <menus>
                        <Menu mnemonicParsing="false" text="Options">
```

Figure 35: Source code of Cart.fxml (1)

```
                        <items>
                            <Menu mnemonicParsing="false" text="Update Store">
                              <items>
                                <MenuItem mnemonicParsing="false" text="Add Book" />
                                    <MenuItem mnemonicParsing="false" text="Add  CD" />
                                    <MenuItem mnemonicParsing="false" text="Add DVD" />
                              </items>
                            </Menu>
                              <MenuItem mnemonicParsing="false" text="View Store" />
                              <MenuItem mnemonicParsing="false" text="View Cart" />
                        </items>
                      </Menu>
                    </menus>
                  </MenuBar>
                  <Label text="CART" textFill="AQUA">
                    <font>
                      <Font size="50.0" />
                    </font>
                    <padding>
                      <Insets left="10.0" />
                    </padding>
                  </Label>
                </children>
              </VBox>
            </top>
            <center>
              <VBox prefHeight="200.0" prefWidth="100.0" BorderPane.alignment="CENTER">
                <padding>
                  <Insets left="10.0" />
                </padding>
```

Figure 36: Source code of Cart.fxml (2)

```
<children>
   <HBox alignment="CENTER_LEFT" prefWidth="200.0" spacing="10.0">
      <VBox.margin>
         <Insets />
      </VBox.margin>
      <padding>
         <Insets bottom="10.0" top="10.0" />
      </padding>
      <children>
         <Label text="Filter:" />
         <TextField />
         <RadioButton mnemonicParsing="false" selected="true" text="By ID">
            <toggleGroup>
               <ToggleGroup fx:id="filterCategory" />
            </toggleGroup>
         </RadioButton>
         <RadioButton mnemonicParsing="false" text="By Title" toggleGroup="$filterCategory" />
      </children>
   </HBox>
   <TableView>
     <columns>
       <TableColumn prefWidth="75.0" text="Text" />
       <TableColumn prefWidth="75.0" text="Category" />
         <TableColumn prefWidth="75.0" text="Cost" />
     </columns>
      <columnResizePolicy>
         <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
      </columnResizePolicy>
   </TableView>
   <ButtonBar prefHeight="40.0" prefWidth="200.0">
```

Figure 37: Source code of Cart.fxml (3)

```
                    <buttons>
                      <Button mnemonicParsing="false" text="Play" />
                        <Button mnemonicParsing="false" text="Remove" />
                    </buttons>
                  </ButtonBar>
               </children>
            </VBox>
         </center>
         <right>
            <VBox alignment="TOP_CENTER" prefHeight="200.0" BorderPane.alignment="CENTER">
               <padding>
                  <Insets top="50.0" />
               </padding>
               <children>
                  <HBox alignment="CENTER">
                     <children>
                        <Label lineSpacing="10.0" text="Total:">
                           <font>
                              <Font size="24.0" />
                           </font>
                        </Label>
                        <Label text="0 $" textFill="AQUA">
                           <font>
                              <Font size="24.0" />
                           </font>
                        </Label>
                     </children>
                  </HBox>
                  <Button mnemonicParsing="false" style="-fx-background-color: red;"
                          text="Place Order" textFill="WHITE">
```

Figure 38: Source code of Cart.fxml (4)

```
                     <font>
                        <Font size="24.0" />
                     </font>
                  </Button>
               </children>
            </VBox>
         </right>
      </BorderPane>
   </children>
</AnchorPane>
```

Figure 39: Source code of Cart.fxml (5)

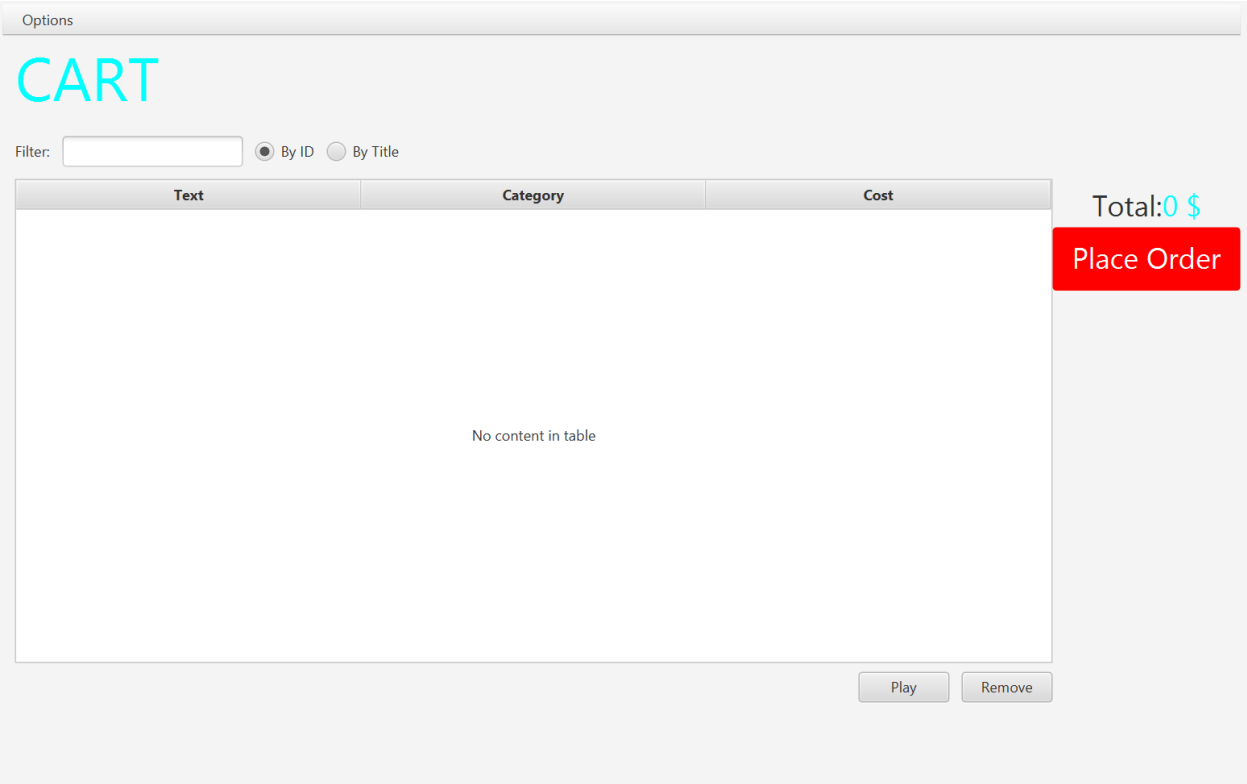

Figure 40: View Cart Screen

6. Integrating JavaFX into Swing application – The JFXPanel class

```java
package hust.soict.aims.screen;

import hust.soict.aims.cart.Cart;
import hust.soict.javafx.CartContronller;
import javafx.application.Platform;
import javafx.embed.swing.JFXPanel;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;

import javax.swing.*;
import java.io.IOException;

no usages  new *
public class CartScreen extends JFrame {
    1 usage
    private Cart cart;

    no usages  new *
    public  CartScreen(Cart cart){
        super();
        this.cart = cart;
        JFXPanel jfxPanel = new JFXPanel();
        this.add(jfxPanel);
        this.setTitle("Cart");
        this.setVisible(true);
```

Figure 41: Source code for CartScreen(1)

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

Figure 42: Source code for CartScreen(2)

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

```java
package hust.soict.javafx;
import hust.soict.aims.media.Media;
import hust.soict.aims.cart.Cart;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;

new *
public class CartScreenContronller {
    2 usages
    private Cart cart;
    no usages
    private FXMLLoader loader;
    1 usage
    @FXML
    private TableView<Media> tbMedia;
    1 usage
    @FXML
    private TableColumn<Media, String> colMediaTitle;
    1 usage
    @FXML
    private TableColumn<Media, String> colMediaCategory;
```

Figure 43: Source code for CartScreenController (1)

```
1 usage
    @FXML
    private TableColumn<Media, Float> colMediaCost;

    1 usage  new *
    public CartScreenContronller(Cart cart){
        super();
        this.cart = cart;
    }

    no usages  new *
    @FXML
    private void initialize(){
        colMediaTitle.setCellValueFactory(
                new PropertyValueFactory<Media, String>( s: "title"));
        colMediaCategory.setCellValueFactory(
                new PropertyValueFactory<Media, String>( s: "catagory")
        );
        colMediaCost.setCellValueFactory(
                new PropertyValueFactory<Media, Float>( s: "cost")
        );
        tbMedia.setItems(this.cart.getItemsOrdered());
    }
}
```

Figure 44: Source code for CartScreenController (3)

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

```java
@FXML
private void initialize(){
    colMediaTitle.setCellValueFactory(
            new PropertyValueFactory<Media, String>( s: "title"));
    colMediaCategory.setCellValueFactory(
            new PropertyValueFactory<Media, String>( s: "catagory"));
    colMediaCost.setCellValueFactory(
            new PropertyValueFactory<Media, Float>( s: "cost"));
    tbMedia.setItems(this.cart.getItemsOrdered());

    btnPlay.setVisible(false);
    btnRemove.setVisible(false);
    tbMedia.getSelectionModel().selectedItemProperty().addListener(
            new *
            new ChangeListener<Media>() {
                new *
                @Override
                public void changed(ObservableValue<? extends Media> observableValue, Media oldValue, Media newValue) {
                    if(newValue!=null){
                        updateButtonBar(newValue);
                    }
                }
            }
    );
}
```

Figure 45: Modified initialize() method

```java
1 usage  new *
void updateButtonBar(Media media){
    btnRemove.setVisible(true);
    if (media instanceof IPlayable){
        btnPlay.setVisible(true);
    }
    else {
        btnPlay.setVisible(false);
    }
}
```

Figure 46: Source code of updateButtonBar()

9. Deleting a media

```
75      @FXML
76      void btnRemovePressed (ActionEvent event) throws Exception {
77          Media media = tbMedia.getSelectionModel().getSelectedItem();
78          cart.removeMedia(media);
79          lbTotalCost.setText(Double.toString(cart.totalCost()) + " $");
80      }
81
```

Figure 47: Handle remove media

10. Filter items in cart – FilteredList

```
        no usages
20      @FXML
21      private RadioButton radioBtnFilterId;
22
        no usages
23      @FXML
24      private RadioButton radioBtnFilterTitle;
25
        no usages
26      @FXML
27      private TextField tfFilter;
```

Figure 48: Creat corresponding attributes in the controller

```
75      tfFilter.textProperty().addListener(new ChangeListener<String>() {
            new *
76          @Override
77          public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
78              showFilteredMedia(newValue);
79          }
80      });
81  }
82
```

Figure 49: Adding ChangListener for tfFilter in initialize()

```
83      void showFilteredMedia(String filterText) {
84          ObservableList<Media> mediaList = cart.getItemsOrdered();
85          FilteredList<Media> filteredMediaList = new FilteredList<>(mediaList);
86
87          Predicate<Media> filterPredicate = media -> {
88              return media.getTitle().toLowerCase().contains(filterText.toLowerCase());
89          };
90          filteredMediaList.setPredicate(filterPredicate);
91          tbMedia.setItems(filteredMediaList);
92
93          if (tbMedia.getSelectionModel().getSelectedItem() != null) {
94              updateButtonBar(tbMedia.getSelectionModel().getSelectedItem());
95          } else {
96              btnPlay.setVisible(false);
97              btnRemove.setVisible(false);
98          }
99      }
```

Figure 50: Source code of showFilteredMedia()

11. Complete the Aims GUI application

```
1   package hust.soict.aims.screen;
2
3   import hust.soict.aims.cart.Cart;
4   import hust.soict.aims.media.AddBookToStore;
5   import hust.soict.aims.media.AddCDToStore;
6   import hust.soict.aims.media.Media;
7   import hust.soict.aims.store.Store;
8
9   import javax.swing.*;
10  import java.awt.*;
11  import java.awt.event.ActionEvent;
12  import java.awt.event.ActionListener;
13
    5 usages  3 inheritors  new *
14  public class AddItemToStoreScreen extends JFrame {
15      // Attribute
        3 usages
16      protected Store store;
17      protected Cart cart;
18      protected String title = "Add media to store";
        2 usages
19      protected String label = "ADD MEDIA TO STORE";
        3 usages
20      protected JTextField tfId = new JTextField( columns: 10);
        3 usages
21      protected JTextField tfTitle = new JTextField( columns: 10);
```

```
21      protected JTextField tfTitle = new JTextField( columns: 10);
        3 usages
22      protected JTextField tfCategory = new JTextField( columns: 10);
        3 usages
23      protected JTextField tfCost = new JTextField( columns: 10);
24
        3 usages   new *
25      public AddItemToStoreScreen(Store store, Cart cart, String title, String label) {
26          this.store = store;
27          this.cart = cart;
28          this.title = title;
29          this.label = label;
30
31          Container cp = getContentPane();
32          cp.setLayout(new BorderLayout());
33
34          cp.add(createCenter(), BorderLayout.CENTER);
35          cp.add(createNorth(), BorderLayout.NORTH);
36
37          setVisible(true);
38          setTitle(title);
39          setSize( width: 1024,  height: 768);
40      }
```

```
43      JPanel createNorth() {
44          JPanel north = new JPanel();
45          north.setLayout(new BoxLayout(north, BoxLayout.Y_AXIS));
46          north.add(createMenuBar());
47          north.add(createHeader());
48          return north;
49      }
50
51
        1 usage   new *
52      JMenuBar createMenuBar() {
53          JMenu menu = new JMenu( s: "Options");
54
55          JMenu smUpdateStore = new JMenu( s: "Update Store");
56          MenuListener menuListener = new MenuListener(store, cart);
57          JMenuItem addBook = new JMenuItem( text: "Add Book");
58          addBook.addActionListener(menuListener);
59          smUpdateStore.add(addBook);
60          JMenuItem addCD = new JMenuItem( text: "Add CD");
61          addCD.addActionListener(menuListener);
62          smUpdateStore.add(addCD);
63          JMenuItem addDVD = new JMenuItem( text: "Add DVD");
64          addDVD.addActionListener(menuListener);
65          smUpdateStore.add(addDVD);
```

```
67          menu.add(smUpdateStore);
            JMenuItem viewStore = new JMenuItem( text: "View store");
69          viewStore.addActionListener(menuListener);
70          menu.add(viewStore);
71          JMenuItem viewCart = new JMenuItem( text: "View cart");
72          viewCart.addActionListener(menuListener);
73          menu.add(viewCart);
74
75          JMenuBar menuBar = new JMenuBar();
76          menuBar.setLayout(new FlowLayout(FlowLayout.LEFT));
77          menuBar.add(menu);
78
79          return menuBar;
80      }
81
            1 usage  new *
82      JPanel createHeader() {
83          JPanel header = new JPanel();
84          header.setLayout(new BoxLayout(header, BoxLayout.X_AXIS));
85
86          JLabel title = new JLabel(label);
87          title.setFont(new Font(title.getFont().getName(), Font.PLAIN,  size: 50));
88          title.setForeground(Color.CYAN);
89
```

```
86          JLabel title = new JLabel(label);
87          title.setFont(new Font(title.getFont().getName(), Font.PLAIN,  size: 50));
88          title.setForeground(Color.CYAN);
89
90          ButtonListener buttonListener = new ButtonListener(store, cart);
91          JButton add = new JButton( text: "Add");
92          add.setPreferredSize(new Dimension( width: 100,  height: 50));
93          add.setMaximumSize(new Dimension( width: 100,  height: 50));
94          add.addActionListener(buttonListener);
95
96          header.add(Box.createRigidArea(new Dimension( width: 10,  height: 10)));
97          header.add(title);
98          header.add(Box.createHorizontalGlue());
99          header.add(add);
100         header.add(Box.createRigidArea(new Dimension( width: 10,  height: 10)));
101
102         return header;
103     }
104
            1 usage  1 override  new *
105     JPanel createCenter() {
106         return null;
107     }
108
```

```
109     public void addInputRow(JPanel center, String att, JTextField tf) {
110         JLabel l = new JLabel(att, JLabel.LEFT);
111         l.setLabelFor(tf);
112         center.add(tf);
113         center.add(l);
114     }
115
            1 usage  3 overrides  new *
116     public Media createMedia() {
117         return null;
118     }
119
            2 usages  new *
120     private class ButtonListener implements ActionListener {
                3 usages
121         private Store store;
                2 usages
122         private Cart cart;
123
                1 usage  new *
124         public ButtonListener(Store store, Cart cart) {
125             super();
126             this.store = store;
127             this.cart = cart;
128         }
```

```
                new *
130         @Override
131         public void actionPerformed(ActionEvent e) {
132             String button = e.getActionCommand();
133             if (button.equals("Add")) {
134                 Media media = createMedia();
135                 try {
136                     store.addMedia(media);
137                 } catch (Exception ex) {
138                     throw new RuntimeException(ex);
139                 }
140                 new StoreScreen(store, cart);
141                 setVisible(false);
142                 dispose();
143             }
144         }
145     }
146
            2 usages  new *
147     private class MenuListener implements ActionListener {
                5 usages
148         private Store store;
                6 usages
149         private Cart cart;
150
```

```
1 usage  new *
    public MenuListener(Store store, Cart cart) {
        super();
        this.store = store;
        this.cart = cart;
    }

    new *
    @Override
    public void actionPerformed(ActionEvent e) {
        String menu = e.getActionCommand();
        if (menu.equals("Add Book")) {
            new AddBookToStore(store, cart);
            setVisible(false);
            dispose();
        } else if (menu.equals("Add CD")) {
            new AddCDToStore(store, cart);
            setVisible(false);
            dispose();
        } else if (menu.equals("Add DVD")) {
            new AddDVDToStoreScreen(store, cart);
            setVisible(false);
            dispose();
        } else if (menu.equals("View store")) {
            new StoreScreen(store, cart);
            setVisible(false);
            dispose();
        } else if (menu.equals("View cart")) {
            new CartScreen(cart);
            setVisible(false);
            dispose();
        }

    }
}

```

```java
public class AddDVDToStoreScreen extends  AddItemToStoreScreen{
    2 usages
    protected JTextField tfDirector;
    2 usages
    protected JTextField tfLength;

    1 usage  new *
    public AddDVDToStoreScreen(Store store, Cart cart) {
        super(store, cart, title: "Add DVD to store", label: "ADD DVD TO STORE");
    }

    1 usage  new *
    JPanel createCenter() {
        JPanel center = new JPanel();
        tfDirector = new JTextField( columns: 10);
        tfLength = new JTextField( columns: 10);
        center.setLayout (new GridLayout( rows: 6, cols: 2, hgap: 2, vgap: 2));
        addInputRow(center, att: "Id", tfId);
        addInputRow(center, att: "Title", tfTitle);
        addInputRow(center, att: "Category", tfCategory);
        addInputRow(center, att: "Cost", tfCost);
        addInputRow(center, att: "Director", tfDirector);
        addInputRow(center, att: "Length", tfLength);
        return center;
    }
```

12. Use case Diagram

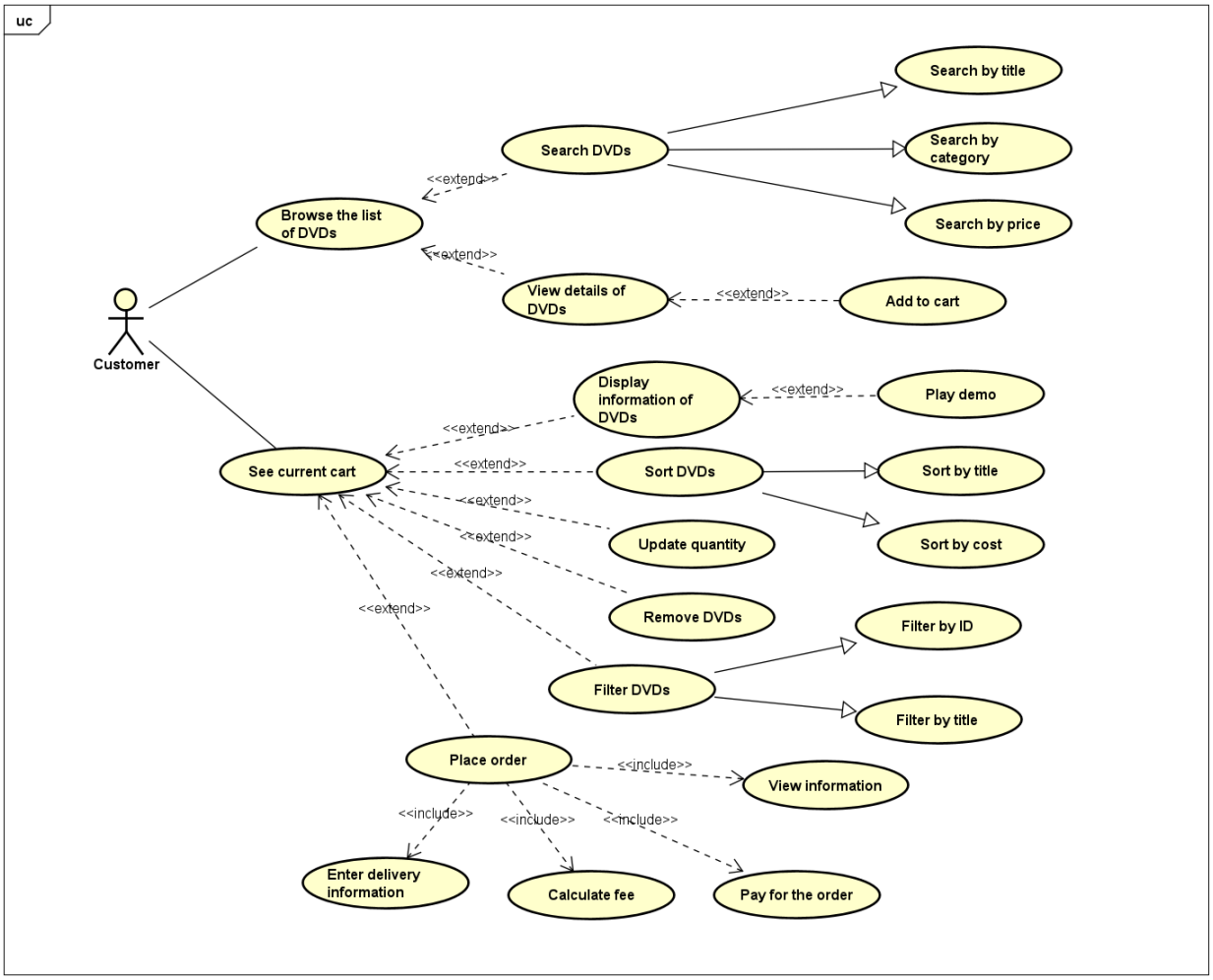

Figure 51: UseCase Diagram Customer

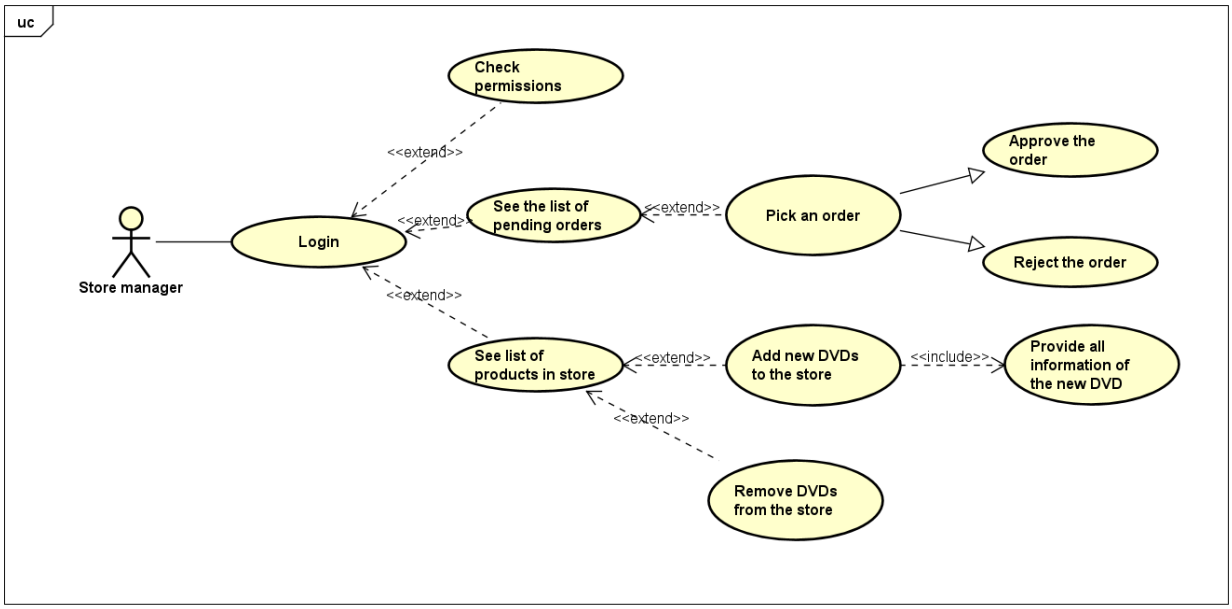

Figure 52: UseCase Diagram Store manager

13. Class Diagram

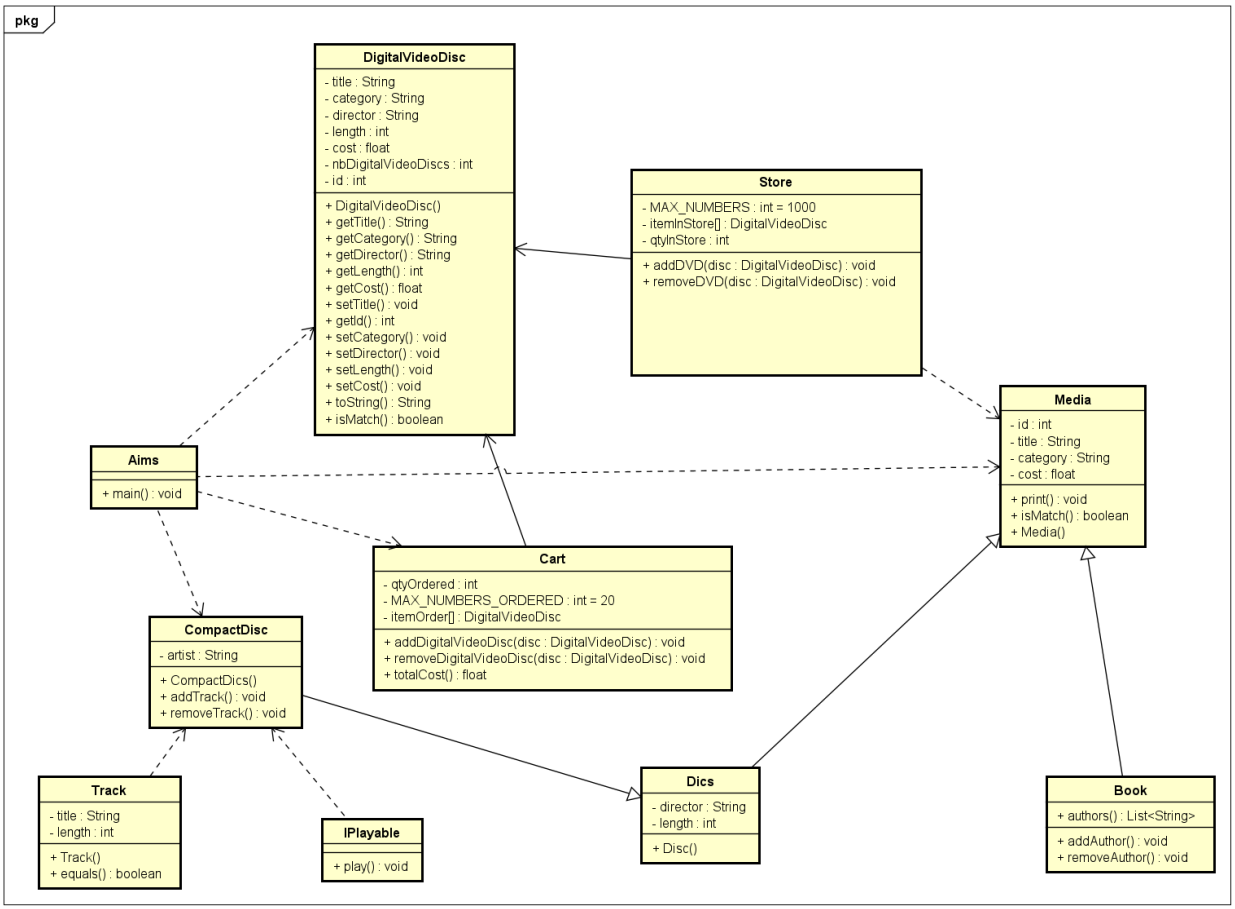

Figure 53: Class Diagram update