# OOP LAB05

# 1.1. AWTAccumulator

## 1.1.1. Create class AWTAccumulator with the source code as below

|  |
| --- |
| package hust.soict.dsai.GUIProject;  import java.awt.\*;  import java.awt.event.\*;  public class AWTAccumulator extends Frame {  private TextField tfInput;  private TextField tfOutput;  private int sum = 0; // Accumulated sum, init to 0  // Constructor to setup the GUI components and event handlers  public AWTAccumulator() {  setLayout(new GridLayout(2, 2));  add(new Label("Enter an Integer: "));  tfInput = new TextField(10);  add(tfInput);  tfInput.addActionListener(new TFInputListener());  add(new Label("The Accumulated Sum is: "));  tfOutput = new TextField(10);  tfOutput.setEditable(false);  add(tfOutput);  setTitle("AWT Accumulator");  setSize(350, 120);  setVisible(true);  }  public static void main(String[] args) {  new AWTAccumulator();  }  private class TFInputListener implements ActionListener {  @Override  public void actionPerformed(ActionEvent evt) {  int numberIn = Integer.*parseInt*(tfInput.getText());  sum += numberIn;  tfInput.setText("");  tfOutput.setText(sum + "");  }  }  } |

## 1.2.1. Create class SwingAccumulator with the source code as below:

|  |
| --- |
| package hust.soict.dsai.GUIProject;  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  public class SwingAccumulator extends JFrame {  private JTextField tfInput;  private JTextField tfOutput;  private int sum = 0; // Accumulated sum, init to 0  // Constructor to setup the GUI components and event handlers  public SwingAccumulator() {  Container cp = getContentPane();  cp.setLayout(new GridLayout(2, 2));  cp.add(new JLabel("Enter an Integer: "));  tfInput = new JTextField(10);  cp.add(tfInput);  tfInput.addActionListener(new TFInputListener());  cp.add(new JLabel("The Accumulated Sum is: "));  tfOutput = new JTextField(10);  tfOutput.setEditable(false);  cp.add(tfOutput);  setTitle("Swing Accumulator");  setSize(350, 120);  setVisible(true);  }  public static void main(String[] args) {  new SwingAccumulator();  }  private class TFInputListener implements ActionListener {  @Override  public void actionPerformed(ActionEvent evt) {  int numberIn = Integer.*parseInt*(tfInput.getText());  sum += numberIn;  tfInput.setText("");  tfOutput.setText(sum + "");  }  }  } |

**1.3. Compare Swing and AWT elements:**

**AWT Top-Level Containers**:

* **Frame**: Represents the main application window.
* **Dialog**: Used for pop-up dialogs.
* **Panel**: A generic container for grouping components.

**Swing Top-Level Containers**:

* **JFrame**: Swing's equivalent of Frame, but with added flexibility, such as support for look-and-feel and double-buffering.
* **JDialog**: Swing's equivalent of Dialog.
* **JPanel**: Swing's replacement for Panel.

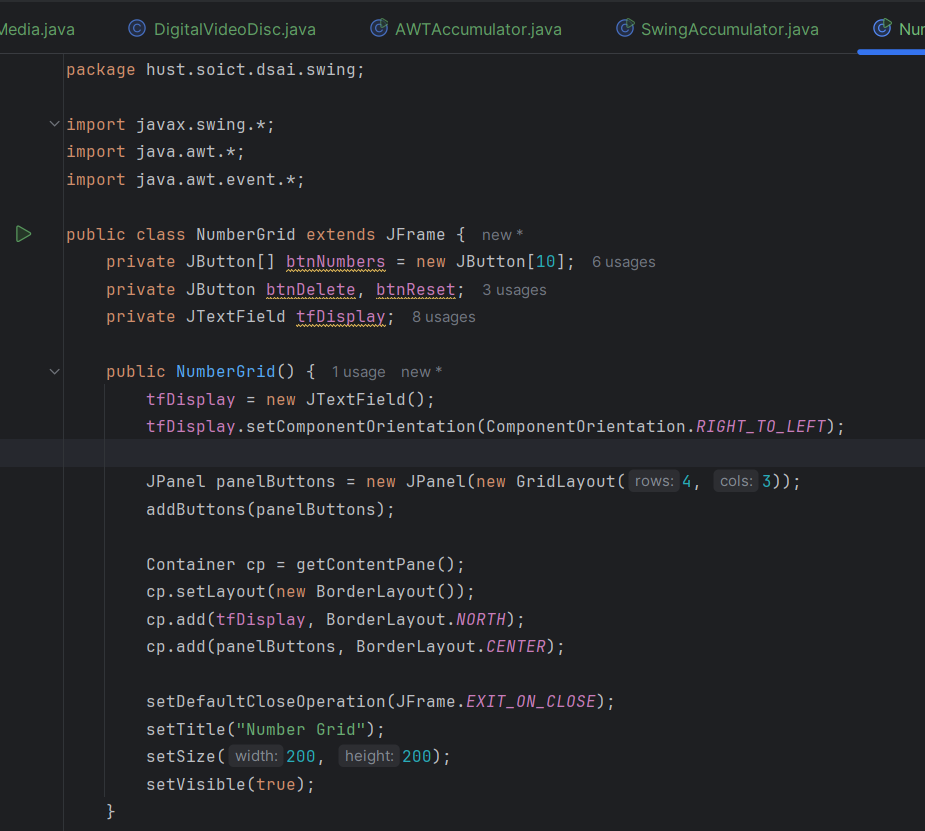
# 2. Organizing Swing components with Layout Managers



## 2.2. Using JPanel as secondary-level container to organize components

### 2.2.1. Create class NumberGrid

|  |
| --- |
| package hust.soict.dsai.swing;  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.\*;  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(4, 3));  addButtons(panelButtons);  Container cp = getContentPane();  cp.setLayout(new BorderLayout());  cp.add(tfDisplay, BorderLayout.*NORTH*);  cp.add(panelButtons, BorderLayout.*CENTER*);  setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  setTitle("Number Grid");  setSize(200, 200);  setVisible(true);  } |



### 2.2.2. Adding buttons

|  |
| --- |
| **void addButtons(JPanel panelButtons) {**  **ButtonListener btnListener = new ButtonListener();**  **// Add number buttons (1-9)**  **for (int i = 1; i <= 9; i++) {**  **btnNumbers[i] = new JButton("" + i);**  **panelButtons.add(btnNumbers[i]);**  **btnNumbers[i].addActionListener(btnListener);**  **}**  **// Add Delete button**  **btnDelete = new JButton("DEL");**  **panelButtons.add(btnDelete);**  **btnDelete.addActionListener(btnListener);**  **// Add 0 button**  **btnNumbers[0] = new JButton("0");**  **panelButtons.add(btnNumbers[0]);**  **btnNumbers[0].addActionListener(btnListener);**  **// Add Reset button**  **btnReset = new JButton("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(0) >= '0' && button.charAt(0) <= '9') {  // Append number to display  tfDisplay.setText(tfDisplay.getText() + button);  } else if (button.equals("DEL")) {  // Handle the "DEL" case  String text = tfDisplay.getText();  if (text.length() > 0) {  tfDisplay.setText(text.substring(0, text.length() - 1));  }  } else if (button.equals("C")) {  // Handle the "C" case  tfDisplay.setText("");  }  }  } |



# 

# 

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

## 3.1.1. Create the StoreScreen class:

## 3.1.2. The NORTH component:

|  |
| --- |
| package hust.soict.dsai.aims.screen;  import hust.soict.dsai.aims.store.Store;  import javax.swing.\*;  import java.awt.\*;  import java.util.ArrayList;  public class storeScreen extends JFrame{  private Store store;  public StoreScreen(Store store) {  this.store = store;  Container cp = getContentPane();  cp.setLayout(new BorderLayout());  cp.add(createNorth(), BorderLayout.*NORTH*);  cp.add(createCenter(), BorderLayout.*CENTER*);  setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  setTitle("AIMS Store");  setSize(1024, 768);  setVisible(true);  }  private JPanel createNorth() {  JPanel north = new JPanel();  north.setLayout(new BoxLayout(north, BoxLayout.*Y\_AXIS*));  north.add(createMenuBar());  north.add(createHeader());  return north;  }  private JMenuBar createMenuBar() {  JMenu menu = new JMenu("Options");  JMenu smUpdateStore = new JMenu("Update Store");  smUpdateStore.add(new JMenuItem("Add Book"));  smUpdateStore.add(new JMenuItem("Add CD"));  smUpdateStore.add(new JMenuItem("Add DVD"));  menu.add(smUpdateStore);  menu.add(new JMenuItem("View store"));  menu.add(new JMenuItem("View cart"));  JMenuBar menuBar = new JMenuBar();  menuBar.setLayout(new FlowLayout(FlowLayout.*LEFT*));  menuBar.add(menu);  return menuBar;  }  private JPanel createHeader() {  JPanel header = new JPanel();  header.setLayout(new BoxLayout(header, BoxLayout.*X\_AXIS*));  JLabel title = new JLabel("AIMS");  title.setFont(new Font(title.getFont().getName(), Font.*PLAIN*, 50));  title.setForeground(Color.*CYAN*);  JButton cart = new JButton("View cart");  cart.setPreferredSize(new Dimension(100, 50));  cart.setMaximumSize(new Dimension(100, 50));  header.add(Box.*createRigidArea*(new Dimension(10, 10)));  header.add(title);  header.add(Box.*createHorizontalGlue*());  header.add(cart);  header.add(Box.*createRigidArea*(new Dimension(10, 10)));  return header;  }  } |

## 3.1.3. The CENTER component

|  |
| --- |
| private JPanel createCenter() {  JPanel center = new JPanel();  center.setLayout(new GridLayout(3, 3, 2, 2));  ArrayList<Media> mediaInStore = store.getItemsInStore();  for (int i = 0; i < mediaInStore.size(); i++) {  MediaStore cell = new MediaStore(mediaInStore.get(i));  center.add(cell);  }  return center;  } |

**3.1.4. The MediaStore class**

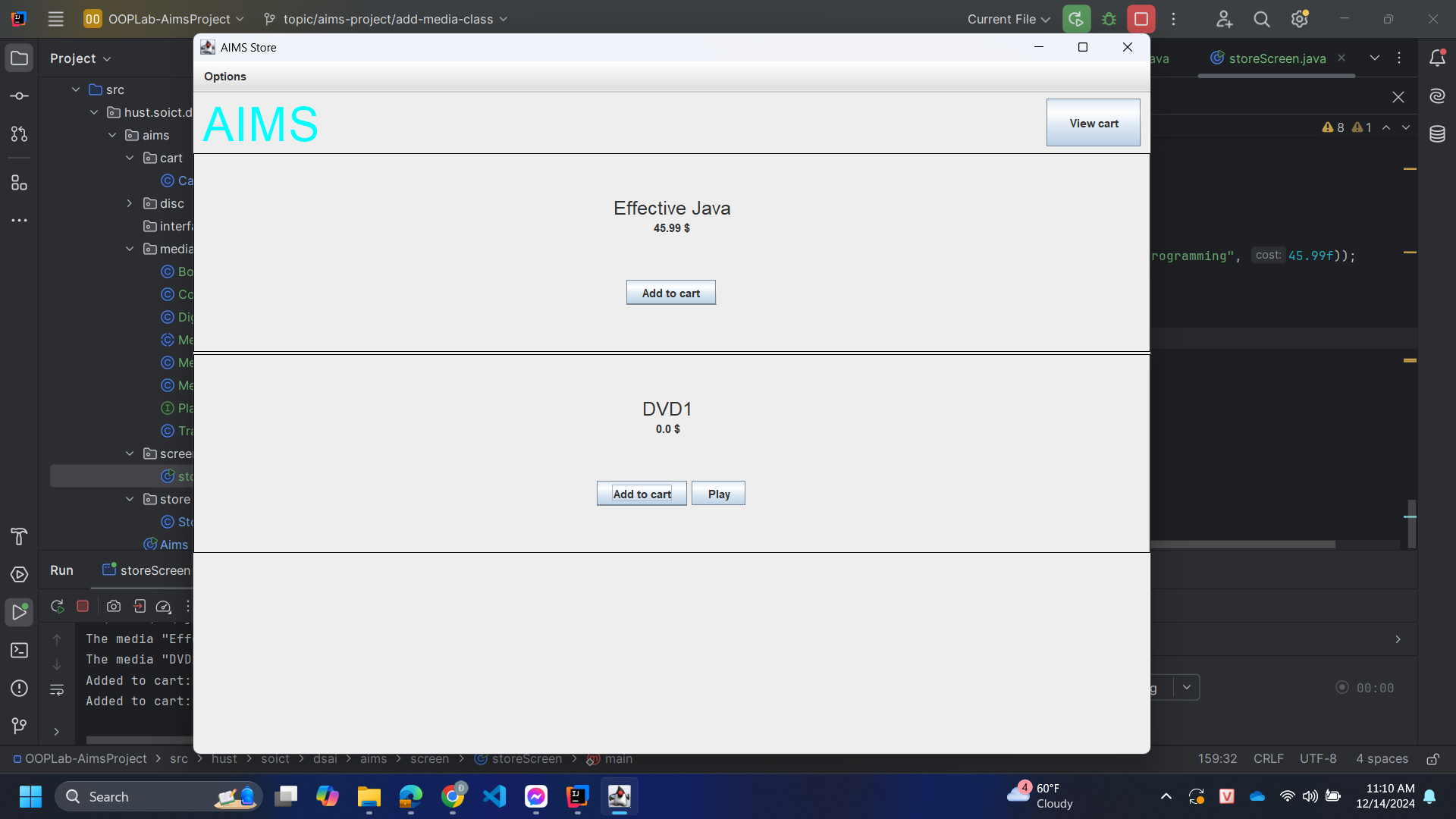
|  |
| --- |
| public class MediaStore extends JPanel {  private Media media;  public MediaStore(Media media) {  this.media = media;  this.setLayout(new BoxLayout(this, BoxLayout.*Y\_AXIS*));  // Title label  JLabel title = new JLabel(media.getTitle());  title.setFont(new Font(title.getFont().getName(), Font.*PLAIN*, 20));  title.setAlignmentX(*CENTER\_ALIGNMENT*);  // Cost label  JLabel cost = new JLabel(media.getCost() + " $");  cost.setAlignmentX(*CENTER\_ALIGNMENT*);  // Container for buttons  JPanel container = new JPanel();  container.setLayout(new FlowLayout(FlowLayout.*CENTER*));  container.add(new JButton("Add to cart"));  // Add "Play" button if the media is playable  if (media instanceof Playable) {  container.add(new JButton("Play"));  }  // Add components to the MediaStore panel  this.add(Box.*createVerticalGlue*());  this.add(title);  this.add(cost);  this.add(Box.*createVerticalGlue*());  this.add(container);  // Add a border around each media item  this.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*));  }  } |

**3.1.5. Putting it all together**

|  |
| --- |
| package hust.soict.dsai.aims.screen;  import hust.soict.dsai.aims.media.Media;  import hust.soict.dsai.aims.media.Playable;  import hust.soict.dsai.aims.store.Store;  import javax.swing.\*;  import java.awt.\*;  import java.util.ArrayList;  public class storeScreen extends JFrame{  private Store store;  private JPanel createNorth() {  JPanel north = new JPanel();  north.setLayout(new BoxLayout(north, BoxLayout.*Y\_AXIS*));  north.add(createMenuBar());  north.add(createHeader());  return north;  }  private JMenuBar createMenuBar() {  JMenu menu = new JMenu("Options");  JMenu smUpdateStore = new JMenu("Update Store");  smUpdateStore.add(new JMenuItem("Add Book"));  smUpdateStore.add(new JMenuItem("Add CD"));  smUpdateStore.add(new JMenuItem("Add DVD"));  menu.add(smUpdateStore);  menu.add(new JMenuItem("View store"));  menu.add(new JMenuItem("View cart"));  JMenuBar menuBar = new JMenuBar();  menuBar.setLayout(new FlowLayout(FlowLayout.*LEFT*));  menuBar.add(menu);  return menuBar;  }  private JPanel createCenter() {  JPanel center = new JPanel();  center.setLayout(new GridLayout(3, 3, 2, 2));  ArrayList<Media> mediaInStore = store.getItemsInStore();  for (int i = 0; i < mediaInStore.size(); i++) {  MediaStore cell = new MediaStore(mediaInStore.get(i));  center.add(cell);  }  return center;  }  private JPanel createHeader() {  JPanel header = new JPanel();  header.setLayout(new BoxLayout(header, BoxLayout.*X\_AXIS*));  JLabel title = new JLabel("AIMS");  title.setFont(new Font(title.getFont().getName(), Font.*PLAIN*, 50));  title.setForeground(Color.*CYAN*);  JButton cart = new JButton("View cart");  cart.setPreferredSize(new Dimension(100, 50));  cart.setMaximumSize(new Dimension(100, 50));  header.add(Box.*createRigidArea*(new Dimension(10, 10)));  header.add(title);  header.add(Box.*createHorizontalGlue*());  header.add(cart);  header.add(Box.*createRigidArea*(new Dimension(10, 10)));  return header;  }  public storeScreen(Store store) {  this.store = store;  Container cp = getContentPane();  cp.setLayout(new BorderLayout());  cp.add(createNorth(), BorderLayout.*NORTH*);  cp.add(createCenter(), BorderLayout.*CENTER*);  setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  setTitle("AIMS Store");  setSize(1024, 768);  setVisible(true);  }  public class MediaStore extends JPanel {  private Media media;  public MediaStore(Media media) {  this.media = media;  this.setLayout(new BoxLayout(this, BoxLayout.*Y\_AXIS*));  // Title label  JLabel title = new JLabel(media.getTitle());  title.setFont(new Font(title.getFont().getName(), Font.*PLAIN*, 20));  title.setAlignmentX(*CENTER\_ALIGNMENT*);  // Cost label  JLabel cost = new JLabel(media.getCost() + " $");  cost.setAlignmentX(*CENTER\_ALIGNMENT*);  // Container for buttons  JPanel container = new JPanel();  container.setLayout(new FlowLayout(FlowLayout.*CENTER*));  container.add(new JButton("Add to cart"));  // Add "Play" button if the media is playable  if (media instanceof Playable) {  container.add(new JButton("Play"));  }  // Add components to the MediaStore panel  this.add(Box.*createVerticalGlue*());  this.add(title);  this.add(cost);  this.add(Box.*createVerticalGlue*());  this.add(container);  // Add a border around each media item  this.setBorder(BorderFactory.*createLineBorder*(Color.*BLACK*));  }  }  } |

**3.2. Adding more user interaction**

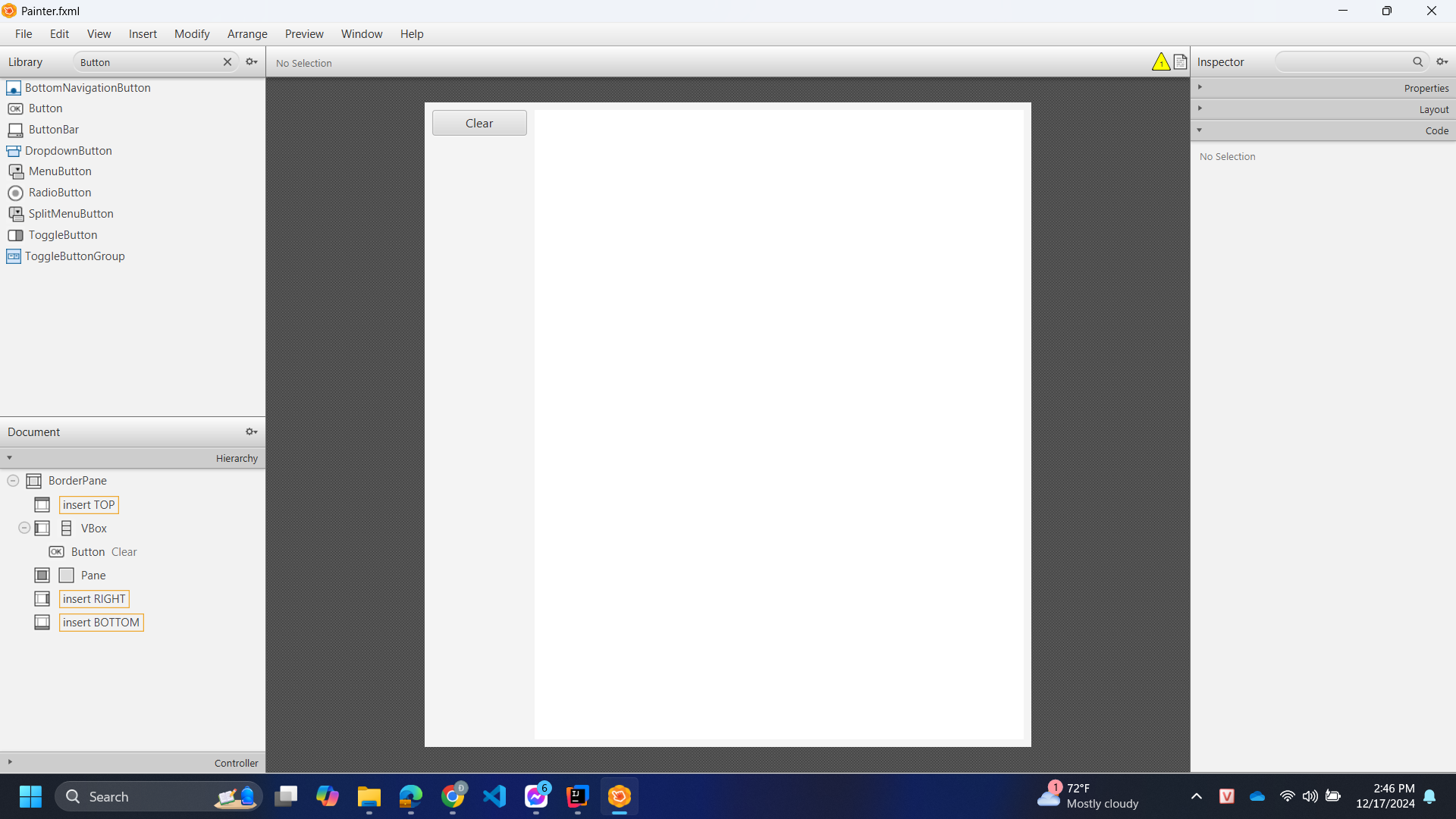
|  |
| --- |
| **public class MediaStore extends JPanel {**  **private Media media;**  **public MediaStore(Media media) {**  **this.media = media;**  **this.setLayout(new BoxLayout(this, BoxLayout.*Y\_AXIS*));**  **// Title label**  **JLabel title = new JLabel(media.getTitle());**  **title.setFont(new Font(title.getFont().getName(), Font.*PLAIN*, 20));**  **title.setAlignmentX(*CENTER\_ALIGNMENT*);**  **// Cost label**  **JLabel cost = new JLabel(media.getCost() + " $");**  **cost.setAlignmentX(*CENTER\_ALIGNMENT*);**  **// Container for buttons**  **JPanel container = new JPanel();**  **container.setLayout(new FlowLayout(FlowLayout.*CENTER*));**  **// Add to cart button**  **JButton btnAddToCart = new JButton("Add to cart");**  **btnAddToCart.addActionListener(new ActionListener() {**  **@Override**  **public void actionPerformed(ActionEvent e) {**  **System.*out*.println("Added to cart: " + media.getTitle());**  **}**  **});**  **container.add(btnAddToCart);**  **// Add "Play" button if the media is playable**  **if (media instanceof Playable) {**  **JButton btnPlay = new JButton("Play");**  **btnPlay.addActionListener(new ActionListener() {**  **@Override**  **public void actionPerformed(ActionEvent e) {**  **JDialog playDialog = new JDialog();**  **playDialog.setTitle("Playing Media");**  **playDialog.setSize(300, 100);**  **playDialog.add(new JLabel("Playing: " + media.getTitle(), SwingConstants.*CENTER*));**  **playDialog.setLocationRelativeTo(null);**  **playDialog.setVisible(true);**  **}**  **});**  **container.add(btnPlay);**  **}** |

****

# 4. JavaFX API:

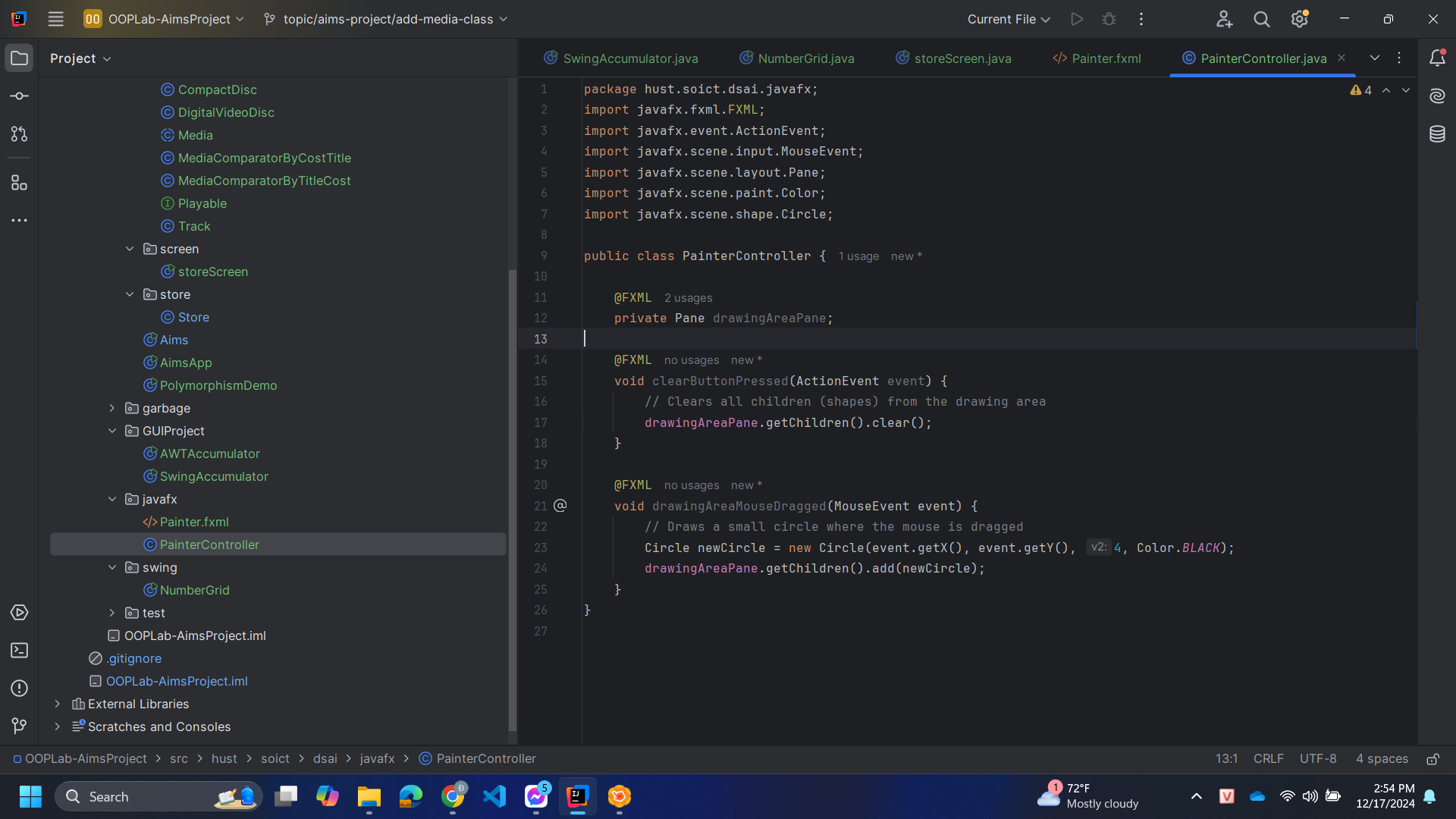
## 4.1. Create the FXML file

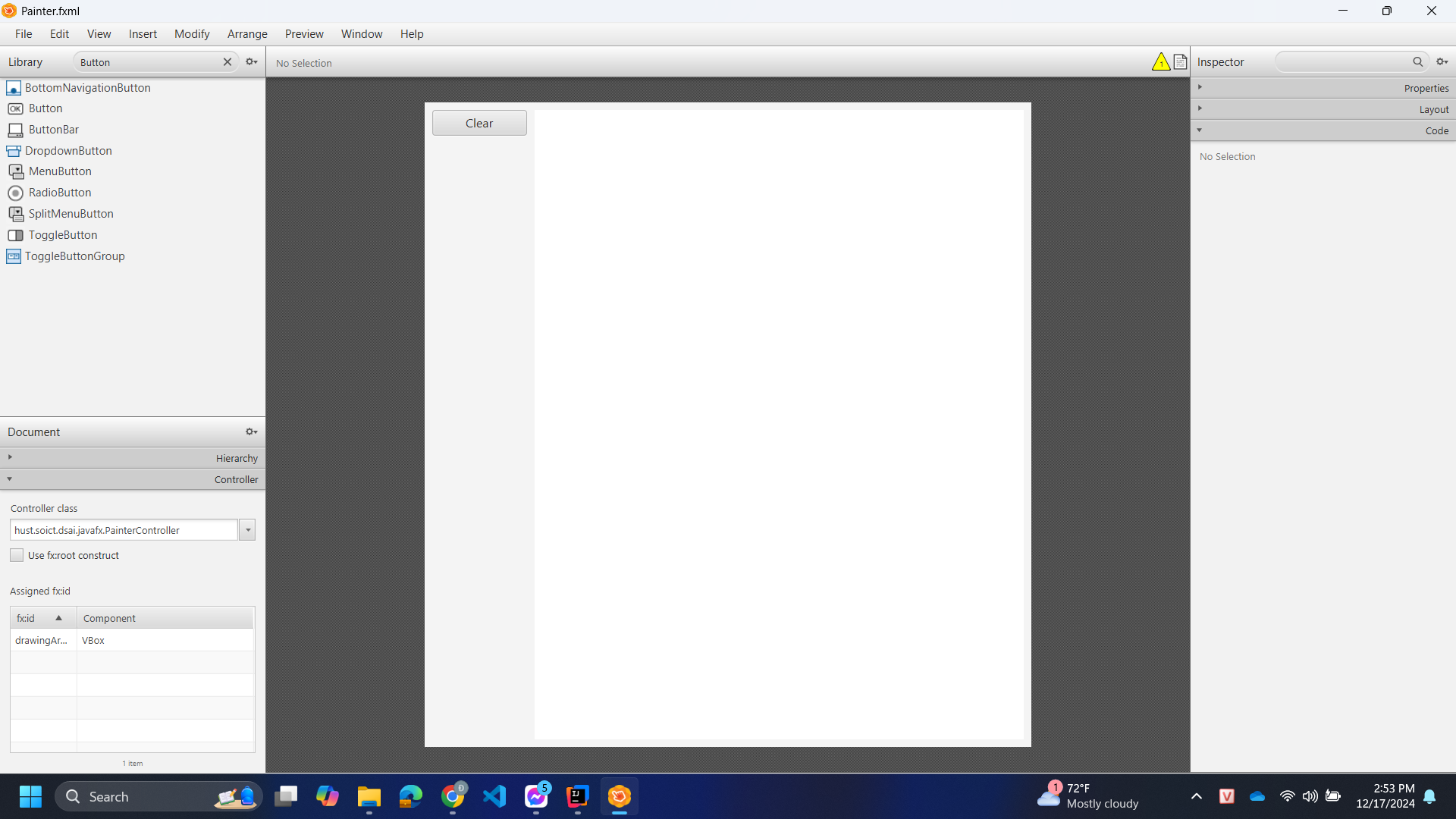
### 4.1.1. Create and open the FXML file in Scene Builder from Eclipse



|  |
| --- |
| **<?xml version="1.0" encoding="UTF-8"?>**  **<?import javafx.scene.layout.BorderPane?>**  **<?import java.lang.\*?>**  **<?import java.util.\*?>**  **<?import javafx.scene.\*?>**  **<?import javafx.scene.control.\*?>**  **<?import javafx.scene.layout.\*?>**  **<AnchorPane xmlns="http://javafx.com/javafx"**  **xmlns:fx="http://javafx.com/fxml"**  **fx:controller="hust.soict.dsai.javafx.Painter"**  **prefHeight="400.0" prefWidth="600.0" >**  **</AnchorPane>** |

## 4.2. Create the controller class



****

|  |
| --- |
| **package hust.soict.dsai.javafx;**  **import javafx.fxml.FXML;**  **import javafx.event.ActionEvent;**  **import javafx.scene.input.MouseEvent;**  **import javafx.scene.layout.Pane;**  **import javafx.scene.paint.Color;**  **import javafx.scene.shape.Circle;**  **public class PainterController {**  **@FXML**  **private Pane drawingAreaPane;**  **@FXML**  **void clearButtonPressed(ActionEvent event) {**  **// Clears all children (shapes) from the drawing area**  **drawingAreaPane.getChildren().clear();**  **}**  **@FXML**  **void drawingAreaMouseDragged(MouseEvent event) {**  **// Draws a small circle where the mouse is dragged**  **Circle newCircle = new Circle(event.getX(), event.getY(), 4, Color.*BLACK*);**  **drawingAreaPane.getChildren().add(newCircle);**  **}**  **}** |

## 4.3. Create the application

### 

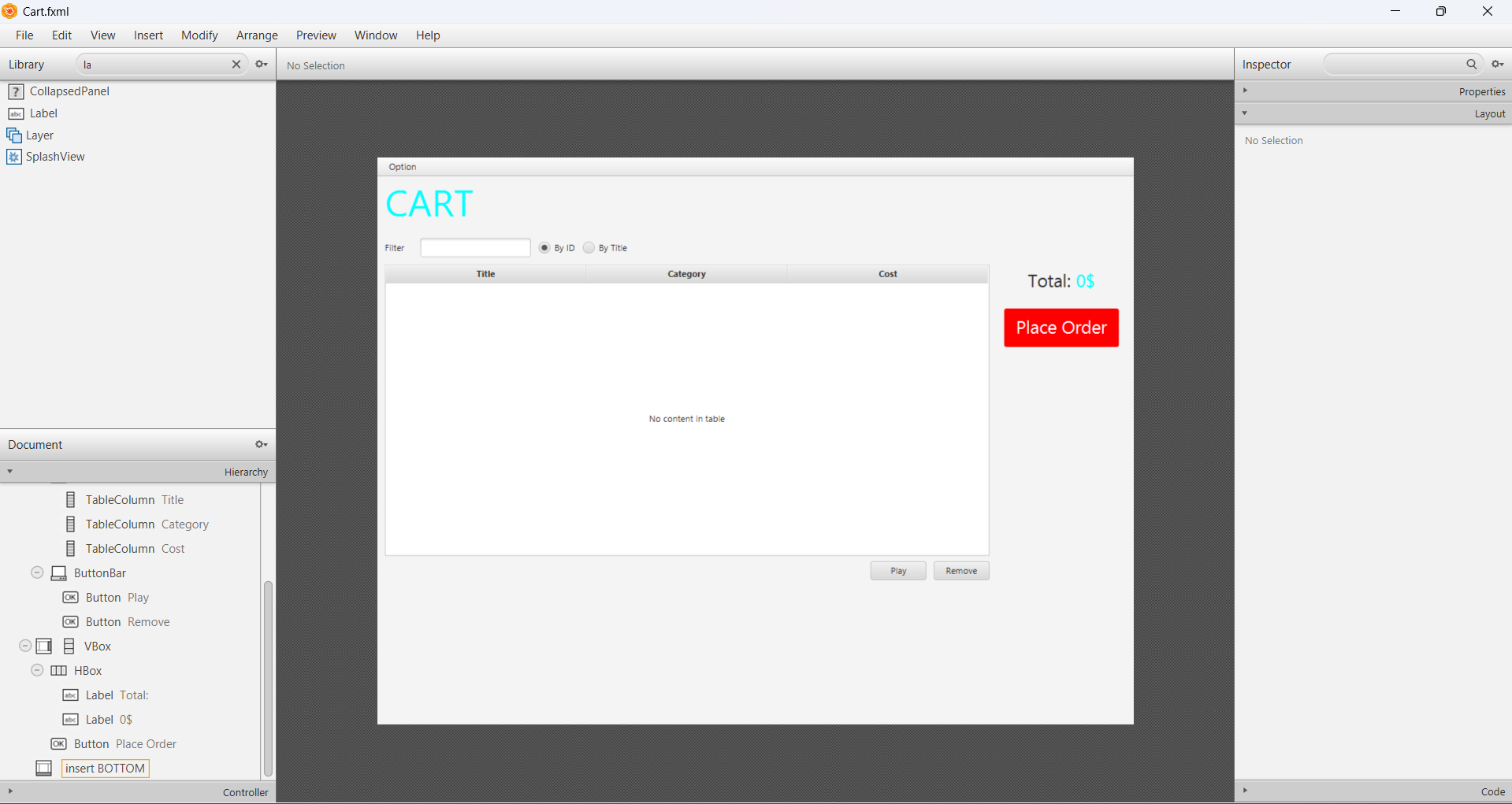
|  |
| --- |
| package hust.soict.dsai.javafx;  import javafx.application.Application;  import javafx.fxml.FXMLLoader;  import javafx.scene.Parent;  import javafx.scene.Scene;  import javafx.stage.Stage;  public class Painter extends Application {  @Override  public void start(Stage stage) throws Exception {  Parent root = FXMLLoader.*load*(getClass()  .getResource("/hust/soict/program/javafx/Painter.fxml"));  Scene scene = new Scene(root);  stage.setTitle("Painter");  stage.setScene(scene);  stage.show();  }  public static void main(String[] args) {  *launch*(args);  }  } |

## 4.4. Practice exercise

## 

# 

# 5. Setting up the View Cart Screen with ScreenBuilder

****

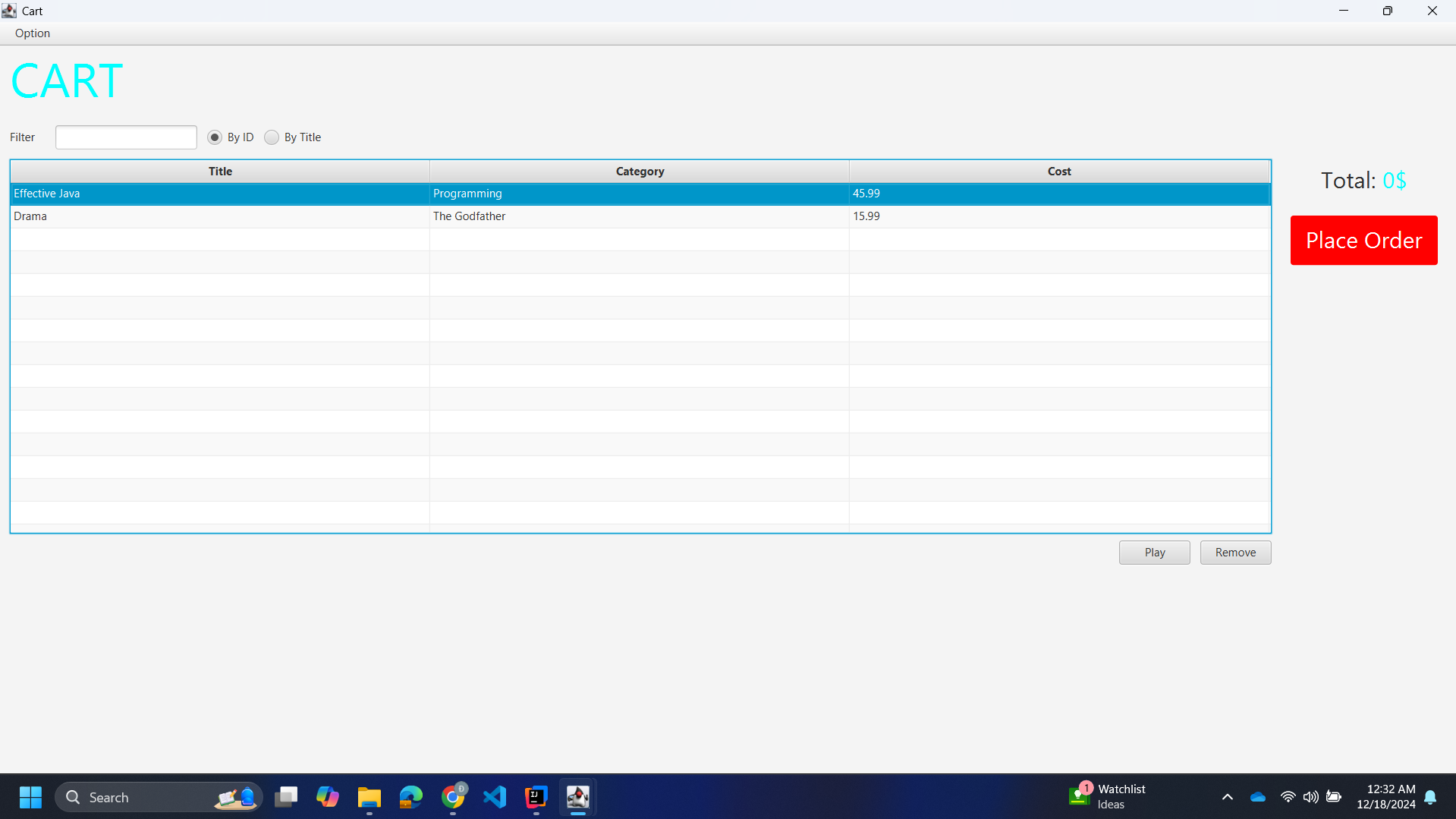
# 

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

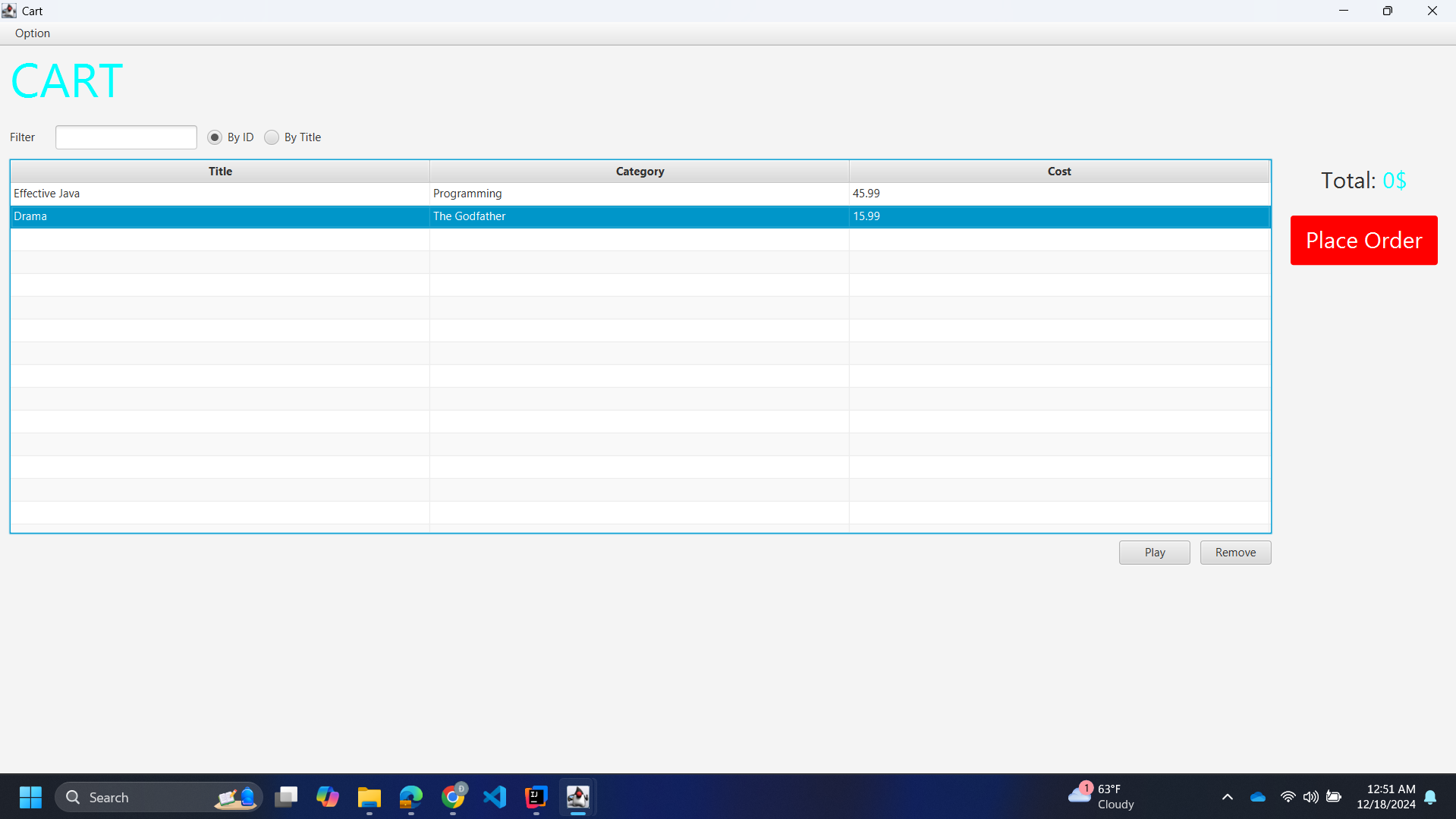
|  |
| --- |
| package hust.soict.dsai.aims.screen;  import hust.soict.dsai.aims.cart.Cart;  import javafx.application.Platform;  import javafx.embed.swing.JFXPanel;  import javafx.fxml.FXMLLoader;  import javafx.scene.Parent;  import javafx.scene.Scene;  import javax.swing.\*;  import java.io.IOException;  public class CartScreen extends JFrame {  private Cart cart;  public CartScreen(Cart cart) {  super();  this.cart = cart;  JFXPanel fxPanel = new JFXPanel();  this.add(fxPanel);  this.setTitle("Cart");  this.setVisible(true);  Platform.*runLater*(new Runnable() {  @Override  public void run() {  try {  FXMLLoader loader = new FXMLLoader(getClass()  .getResource("/screen/fxml/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

|  |
| --- |
| package hust.soict.dsai.javafx;  import hust.soict.dsai.aims.cart.Cart;  import hust.soict.dsai.aims.media.Media;  import javafx.fxml.FXML;  import javafx.scene.control.TableColumn;  import javafx.scene.control.TableView;  import javafx.collections.ObservableList;  import javafx.scene.control.cell.PropertyValueFactory;  public class CartScreenController {  private Cart cart;  @FXML  private TableView<Media> tblMedia;  @FXML  private TableColumn<Media, String> colMediaTitle;  @FXML  private TableColumn<Media, String> colMediacategory;  @FXML  private TableColumn<Media, Float> colMediaCost;  public CartScreenController(Cart cart) {  super();  this.cart = cart;  }  @FXML  private void initialize() {  colMediaTitle.setCellValueFactory(new PropertyValueFactory<>("title"));  colMediacategory.setCellValueFactory(new PropertyValueFactory<>("category"));  colMediaCost.setCellValueFactory(new PropertyValueFactory<>("cost"));  tblMedia.setItems(this.cart.getItemsOrdered());  }  } |



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



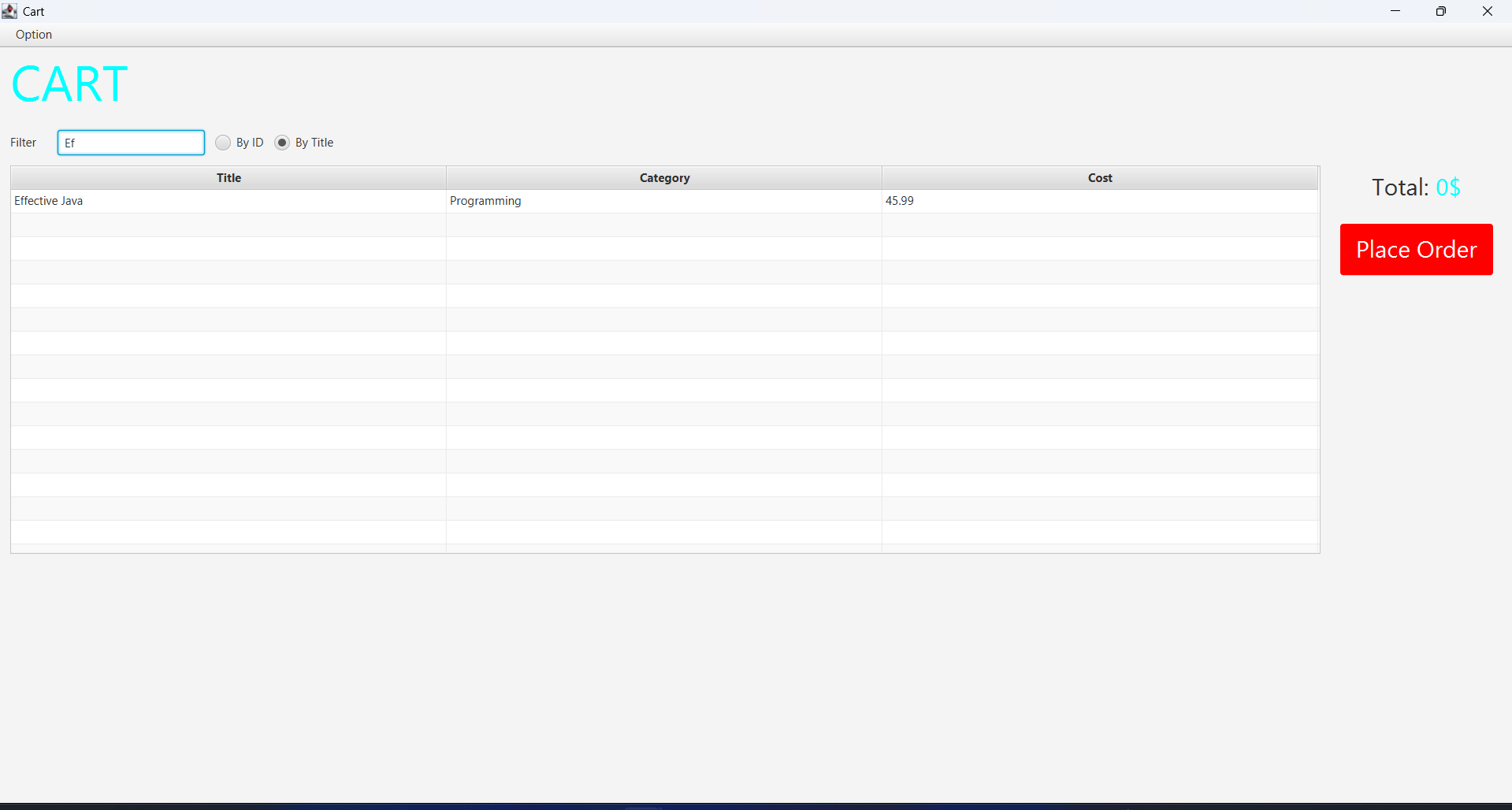
|  |
| --- |
| package hust.soict.dsai.javafx;  import hust.soict.dsai.aims.cart.Cart;  import javafx.beans.value.ChangeListener;  import javafx.beans.value.ObservableValue;  import javafx.collections.ObservableList;  import javafx.fxml.FXML;  import javafx.scene.control.Button;  import javafx.scene.control.TableColumn;  import javafx.scene.control.TableView;  import javafx.scene.control.cell.PropertyValueFactory;  import hust.soict.dsai.aims.media.Media;  import hust.soict.dsai.aims.media.Playable;  public class CartScreenController {  private Cart cart; // Cart object passed to the controller  @FXML  private TableView<Media> tblMedia;  @FXML  private TableColumn<Media, String> colMediaTitle;  @FXML  private TableColumn<Media, String> colMediacategory;  @FXML  private TableColumn<Media, Float> colMediaCost;  @FXML  private Button btnPlay;  @FXML  private Button btnRemove;  public CartScreenController(Cart cart) {  this.cart = cart;  }  @FXML  private void initialize() {  colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>("title"));  colMediacategory.setCellValueFactory(new PropertyValueFactory<Media, String>("category"));  colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, Float>("cost"));  tblMedia.setItems(this.cart.getItemsOrdered());  btnPlay.setVisible(false);  btnRemove.setVisible(false);  tblMedia.getSelectionModel().selectedItemProperty().addListener(  new ChangeListener<Media>() {  @Override  public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {  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

## 

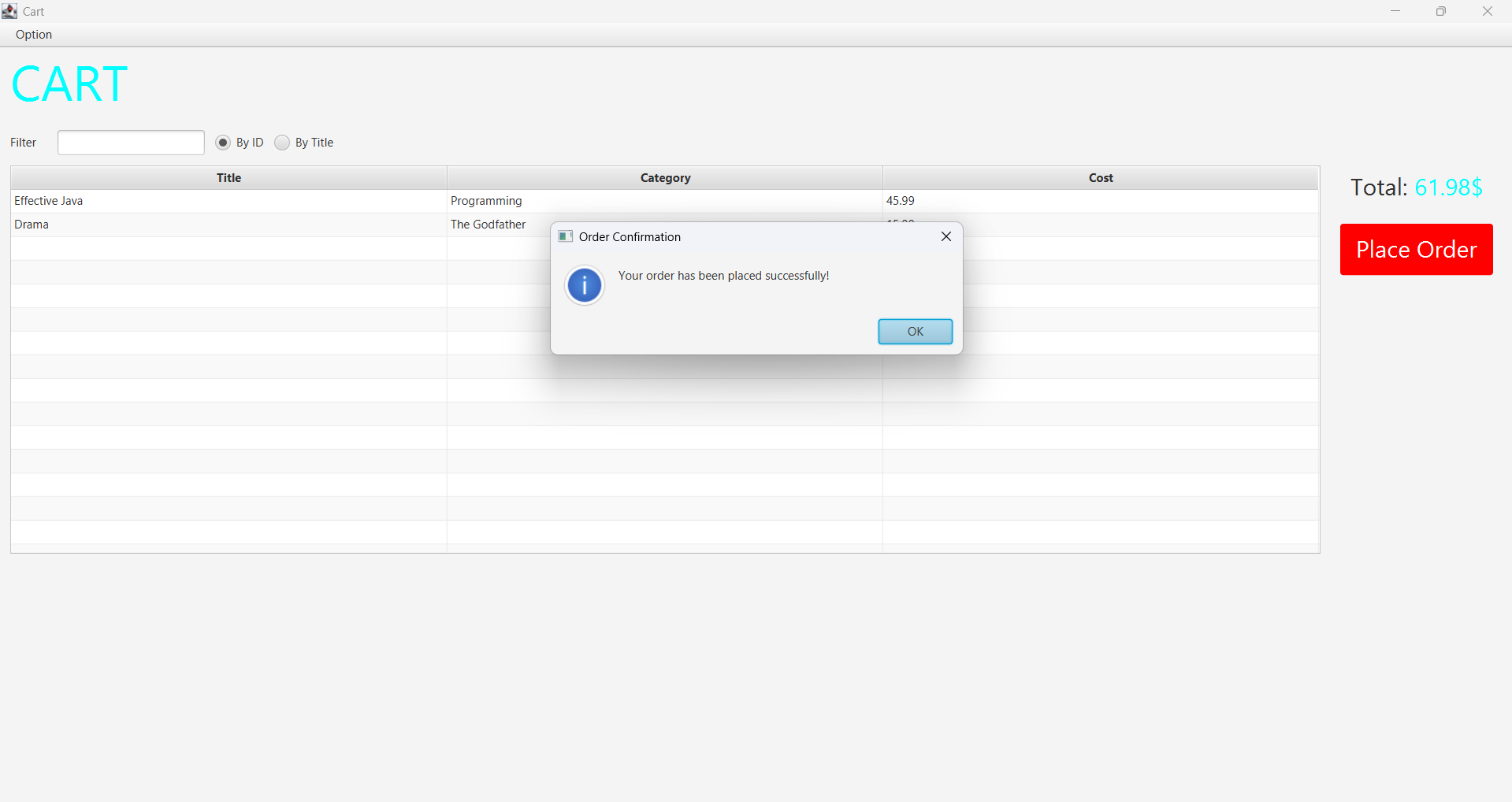
|  |
| --- |
| @FXML  void btnRemovePressed(ActionEvent event) {  Media media = tblMedia.getSelectionModel().getSelectedItem();  if (media != null) {  cart.removeMedia(media);  System.*out*.println("Removed media: " + media.getTitle());  }  } |

# 10. Filter items in cart – FilteredList



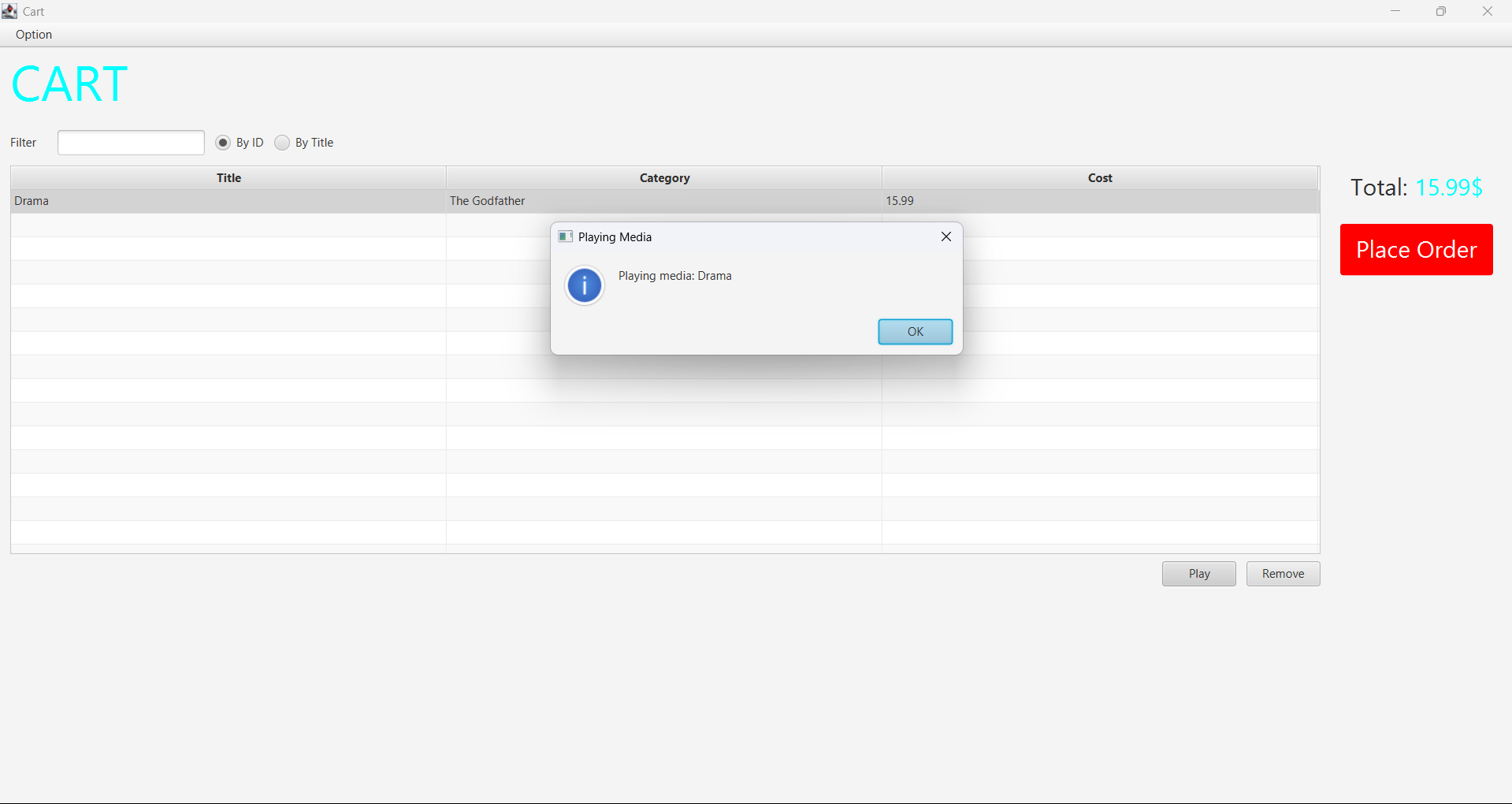
|  |
| --- |
| private void showFilteredMedia(String filterText) {  filteredList.setPredicate(media -> {  // If filter text is empty, display all items  if (filterText == null || filterText.isEmpty()) {  return true;  }  String lowerCaseFilter = filterText.toLowerCase();  if (radioBtnFilterId.isSelected()) {  return String.*valueOf*(media.getId()).contains(filterText);  } else if (radioBtnFilterTitle.isSelected()) {  return media.getTitle().toLowerCase().contains(lowerCaseFilter);  }  return true;  });  } |

# 11. Complete the Aims GUI application

* Cart screen:
* Place Order update:
* 

|  |
| --- |
| @FXML  private void btnPlaceOrderPressed(ActionEvent event) {  Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  alert.setTitle("Order Confirmation");  alert.setHeaderText(null);  alert.setContentText("Your order has been placed successfully!");  alert.showAndWait();  this.cart.clear();  updateTotalCost();  } |

* PlayButton update



@FXML

private void btnPlayPressed(ActionEvent event) {

Alert alert = new Alert(Alert.AlertType.*INFORMATION*);

alert.setTitle("Playing Media");

alert.setHeaderText(null);

alert.setContentText("Playing media");

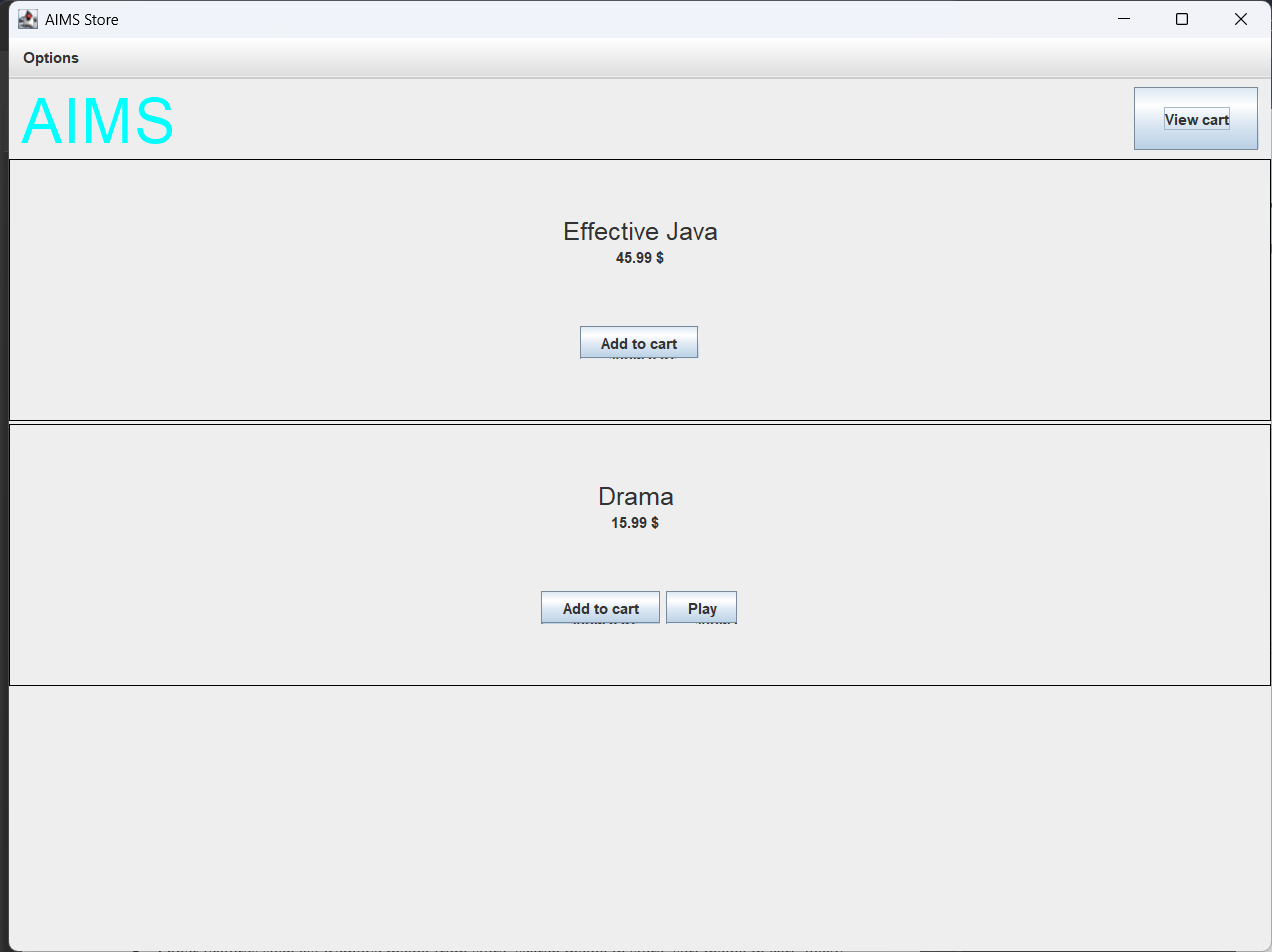
alert.showAndWait();

}

* Totalcost update
* 

|  |
| --- |
| @FXML  private void initialize() {  colMediaTitle.setCellValueFactory(new PropertyValueFactory<>("title"));  colMediacategory.setCellValueFactory(new PropertyValueFactory<>("category"));  colMediaCost.setCellValueFactory(new PropertyValueFactory<>("cost"));  filteredList = new FilteredList<>(this.cart.getItemsOrdered(), p -> true);  tblMedia.setItems(filteredList);  btnPlay.setVisible(false);  btnRemove.setVisible(false);  tblMedia.getSelectionModel().selectedItemProperty().addListener(  new ChangeListener<Media>() {  @Override  public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {  if (newValue != null) {  updateButtonBar(newValue);  }  }  });  // Listener for filter input  tfFilter.textProperty().addListener(new ChangeListener<String>() {  @Override  public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {  showFilteredMedia(newValue);  }  });  // Update total cost dynamically  cart.getItemsOrdered().addListener((ListChangeListener<Media>) c -> {  Platform.*runLater*(() -> updateTotalCost());  });  updateTotalCost();  }  private void updateTotalCost() {  float total = 0;  for (Media media : cart.getItemsOrdered()) {  total += media.getCost();  }  lblTotalCost.setText(String.*format*("Total: %.2f$", total));  } |

* Store Screen



// Add to cart button

JButton btnAddToCart = new JButton("Add to cart");

btnAddToCart.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

cart.addMedia(media); // Add media to cart

System.*out*.println("Added to cart: " + media.getTitle());

}

});

container.add(btnAddToCart);

* Update Store Screen  
    
  Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

  Mô tả được tạo tự động

public abstract class AddItemToStoreScreen extends JFrame {

protected Store store;

protected Cart cart;

protected JTextField titleField, costField;

public AddItemToStoreScreen(Store store, String itemType) {

this.store = store;

setTitle("Add " + itemType);

setSize(400, 300);

setLayout(new BorderLayout());

// Header

JLabel lblHeader = new JLabel("Add " + itemType, SwingConstants.*CENTER*);

lblHeader.setFont(new Font("Arial", Font.*BOLD*, 20));

add(lblHeader, BorderLayout.*NORTH*);

// Input Fields

JPanel inputPanel = new JPanel(new GridLayout(3, 2));

inputPanel.add(new JLabel("Title: "));

titleField = new JTextField();

inputPanel.add(titleField);

inputPanel.add(new JLabel("Cost: "));

costField = new JTextField();

inputPanel.add(costField);

add(inputPanel, BorderLayout.*CENTER*);

// Buttons

JButton btnAdd = new JButton("Add");

btnAdd.addActionListener(e -> handleAdd());

add(btnAdd, BorderLayout.*SOUTH*);

setVisible(true);

}

protected abstract void handleAdd();

}

package hust.soict.dsai.aims.screen;

import hust.soict.dsai.aims.media.CompactDisc;

import hust.soict.dsai.aims.store.Store;

import javax.swing.\*;

public class AddCompactDiscToStoreScreen extends AddItemToStoreScreen {

public AddCompactDiscToStoreScreen(Store store) {

super(store, "CD");

}

@Override

protected void handleAdd() {

try {

String title = titleField.getText();

float cost = Float.*parseFloat*(costField.getText());

CompactDisc cd = new CompactDisc(store.getItemsInStore().size() + 1, title, "CD Category", cost);

store.addMedia(cd);

JOptionPane.*showMessageDialog*(this, "CD added successfully!");

SwingUtilities.*invokeLater*(() -> new storeScreen(store, cart));

dispose();

} catch (Exception e) {

JOptionPane.*showMessageDialog*(this, "Invalid input. Please try again.");

}

}

}

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

public void addMedia(Media m) throws LimitExceededException {

final int MAX\_NUMBERS\_ORDERED = 20; // Example constraint

if (itemsOrdered.size() < MAX\_NUMBERS\_ORDERED) {

itemsOrdered.add(m);

System.*out*.println("Media added to the cart: " + m.getTitle());

} else {

throw new LimitExceededException("ERROR: The number of media items has reached its limit of " + MAX\_NUMBERS\_ORDERED);

}

}

public void removeMedia(Media m) throws IllegalArgumentException {

if (itemsOrdered.contains(m)) {

itemsOrdered.remove(m);

System.*out*.println("Media removed: " + m.getTitle());

} else {

throw new IllegalArgumentException("ERROR: Media not found in the cart.");

}

}

# 13. Create a class which inherits from Exception

package hust.soict.dsai.aims.exception;

public class PlayerException extends Exception {

public PlayerException(String message) {

super(message);

}

}

package hust.soict.dsai.aims.media;

import hust.soict.dsai.aims.exception.PlayerException;

public interface Playable {

void play() throws PlayerException;

}

package hust.soict.dsai.aims.media;

import hust.soict.dsai.aims.exception.PlayerException;

public class DigitalVideoDisc extends Media implements Playable {

private int length;

public DigitalVideoDisc(String title, String category, float cost, int length) {

super(title, category, cost);

this.length = length;

}

public int getLength() {

return length;

}

@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!");

}

}

}

# 14. Update the Aims class

package hust.soict.dsai.aims;

import hust.soict.dsai.aims.exception.PlayerException;

import hust.soict.dsai.aims.media.CompactDisc;

import hust.soict.dsai.aims.media.DigitalVideoDisc;

import hust.soict.dsai.aims.media.Track;

import javax.swing.\*;

public class Aims {

public static void main(String[] args) {

System.*out*.println("=== Testing DigitalVideoDisc ===");

DigitalVideoDisc dvd = new DigitalVideoDisc("The Godfather", "Drama", 19.99f, -1); // Invalid length

try {

dvd.play();

} catch (PlayerException e) {

System.*err*.println("Caught Exception: " + e.getMessage());

e.printStackTrace();

// Display a Swing dialog box for the error

JOptionPane.*showMessageDialog*(null,

"An error occurred while playing DVD:\n" + e.getMessage(),

"Player Error",

JOptionPane.*ERROR\_MESSAGE*);

}

System.*out*.println("\n=== Testing CompactDisc ===");

CompactDisc cd = new CompactDisc("Best Hits", "Music", 15.99f);

cd.addTrack(new Track("Song 1", 3));

cd.addTrack(new Track("Song 2", 0)); // Invalid track length

cd.addTrack(new Track("Song 3", 5));

try {

cd.play();

} catch (PlayerException e) {

System.*err*.println("Caught Exception: " + e.getMessage());

e.printStackTrace();

JOptionPane.*showMessageDialog*(null,

"An error occurred while playing CD:\n" + e.getMessage(),

"Player Error",

JOptionPane.*ERROR\_MESSAGE*);

}

System.*out*.println("\n=== Program Continues Gracefully ===");

}

}

# 

# 15. Modify the equals() method of Media class

package hust.soict.dsai.aims.media;

public class Media {

private String title;

private String category;

private float cost;

public Media(String title, String category, float cost) {

this.title = title;

this.category = category;

this.cost = cost;

}

public String getTitle() {

return title;

}

@Override

public boolean equals(Object obj) {

if (obj == null) {

return false;

}

if (this == obj) {

return true;

}

if (!(obj instanceof Media)) {

return false;

}

Media otherMedia = (Media) obj;

if (this.title == null || otherMedia.title == null) {

return false;

}

return this.title.equals(otherMedia.title);

}

@Override

public int hashCode() {

// Generate hash code based on the title

return (title == null) ? 0 : title.hashCode();

}

}

# 17. Update Aims class diagram

