

ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO THỰC HÀNH
IT1130-744528-2024.1

BÀI THỰC HÀNH 05

Họ và tên sv: Phạm Quốc Cường
Lớp: **K67-Việt Nhật 01**

GVHD: Lê Thị Hoa

TA: **Phạm Mạnh Cường**

Hà Nội 12/2024

BÁO CÁO THỰC HÀNH LAB

5 LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Contents

1.	Swing components.....	4
1.1	AWTAccumulator	4
1.2	SwingAccumulator	5
2	Organizing Swing components with Layout Managers.....	6
2.1	Code	6
2.2	Demo	8
3	Create a graphical user interface for AIMS with Swing.....	9
3.1	Create class StoreScreen.....	9
3.2	Create class MediaStore	13
3.3	Demo	14
4	JavaFX API	16
4.1	Create class Painter	16
4.2	Create Painter.fxml.....	16
4.3	Create class PainterController	17
5	View Cart Screen	19
5.1	Create cart.fxml.....	19
5.2	Create class CartScreen	20
5.3	Create class CartScreenController	21
5.4	Demo	22
6	Updating buttons based on selected item in TableView – ChangeListener	22
6.1	Edit class CartScreenController	22
6.2	Demo	23
7	Deleting a media.....	24
7.1	Code	24
7.2	Demo	25
8	Complete the Aims GUI application	26
9	Use case Diagram.....	30
10	Class Diagram	31

Figure 1.1: Source code of AWTAccumulator	4
Figure 1.2: Demo of AWTAccumulator.....	5
Figure 1.3: Source code of SwingAccumulator.....	5
Figure 1.4: Demo of SwingAccumulator.....	6
Figure 2.1: Source code of NumberGrid 1	6
Figure 2.2: Source code of NumberGrid 2	7
Figure 2.3: Demo buttons 0-9.....	8
Figure 2.4: Demo DEL button.....	8
Figure 2.5: Demo C button.....	8
Figure 3.1: Class StoreScreen 1.....	9
Figure 3.2: Class StoreScreen 2.....	10
Figure 3.3: Class StoreScreen 3.....	10
Figure 3.4: Class StoreScreen 4.....	11
Figure 3.5: Class StoreScreen 5.....	11
Figure 3.6: Class StoreScreen 6.....	12
Figure 3.7: Class MediaStore 1	13
Figure 3.8: Class MediaStore 2	13
Figure 3.9: Class MediaStore 3	14
Figure 3.10: StoreScreen	14
Figure 3.11 Demo Add to cart button.....	15
Figure 3.12 Demo Play button.....	15
Figure 3.13 Demo View cart button	15
Figure 4.1: Class Painter.....	16
Figure 4.2: Painter.fxml 1.....	16
Figure 4.3: Painter.fxml 2.....	17
Figure 4.4: PainterController	17
Figure 4.5: Use Pen	18
Figure 4.6: Use Eraser	18
Figure 4.7: Clear button	18
Figure 5.1: Cart.fxml 1	19
Figure 5.2: Cart.fxml 2	19
Figure 5.3: Cart.fxml 3	20
Figure 5.4: CartScreen class	20
Figure 5.5: CartScreenController 1	21
Figure 5.6: CartScreenController 2	21
Figure 5.7: Demo CartScreen	22
Figure 6.1: CartScreenController 1	22
Figure 6.2: CartScreenController 2	23
Figure 6.3: Demo media playable.....	23
Figure 6.4: Demo media unplayable.....	24
Figure 7.1: btnRemovePressed Method.....	24
Figure 7.2: button Remove	25
Figure 7.3: button Remove	25
Figure 8.1: Store before add book	26

Figure 8.2: Add book.....	26
Figure 8.3: Store after add book	27
Figure 8.4: Add CD.....	27
Figure 8.5: Store after add CD	28
Figure 8.6 Add DVD.....	28
Figure 8.7: Store after add DVD	29
Figure 8.8: Cart	29
Figure 8.9: Exception	30

1. Swing components

1.1 AWTAccumulator



The screenshot shows a Java code editor window with the following code:

```
1 //Pham Quoc Cuong - 20225604
2 package hust.soict.hedspi.swing;
3 import java.awt.Label;
4 import java.awt.GridLayout;
5 import java.awt.TextField;
6 import java.awt.Frame;
7 import java.awt.event.ActionListener;
8 import java.awt.event.ActionEvent;
9 public class AWTAccumulator extends Frame {
10     private TextField tfInput;
11     private TextField tfOutput;
12     private int sum = 0;
13
14     public AWTAccumulator(){
15         setLayout(new GridLayout(2,2));
16         add(new Label("CuongPQ 5604 - Enter an Integer: "));
17
18         tfInput = new TextField(10);
19         add(tfInput);
20         tfInput.addActionListener(new TFInputListener());
21
22         add(new Label("CuongPQ 5604 - The Accumulated Sum is: "));
23
24         tfOutput = new TextField(10);
25         tfOutput.setEditable(false);
26         add(tfOutput);
27
28         setTitle("CuongPQ 5604 - AWT Accumulator");
29         setSize(350, 120);
30         setVisible(true);
31     }
32
33     public static void main(String[] args){
34         new AWTAccumulator();
35     }
36     private class TFInputListener implements ActionListener {
37         @Override
38         public void actionPerformed(ActionEvent ewt){
39             int numberIn = Integer.parseInt(tfInput.getText());
40             sum += numberIn;
41             tfInput.setText("");
42             tfOutput.setText(sum + "");
43         }
44     }
45 }
```

Figure 1.1: Source code of AWTAccumulator

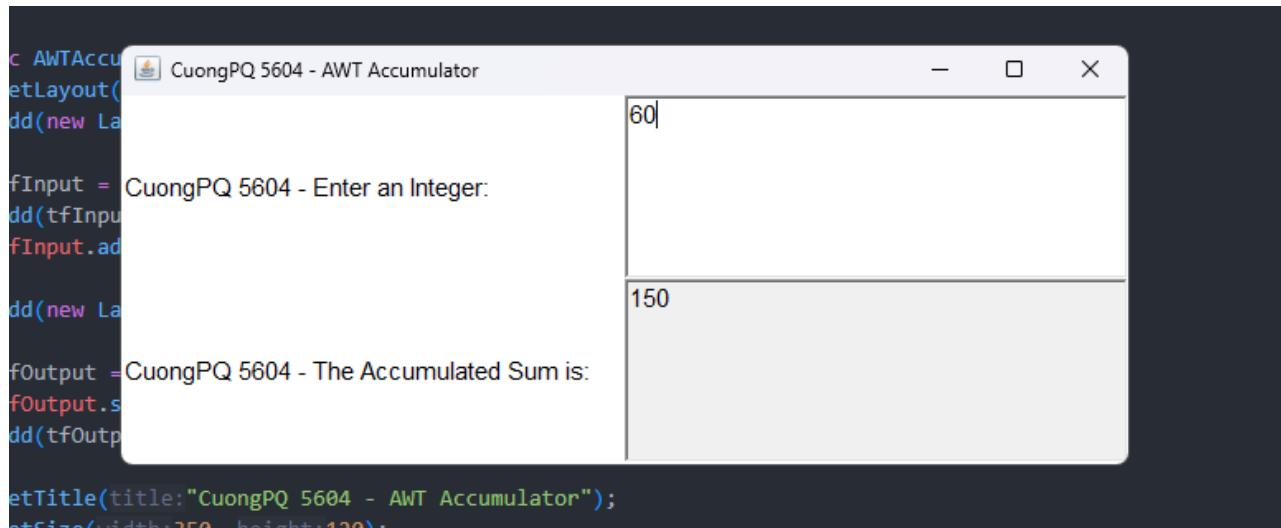


Figure 1.2: Demo of AWTAccumulator

1.2 SwingAccumulator



```
1 // Pham Quoc Cuong - 20225604
2 package hust.soict.hedspi.swing;
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.ActionListener;
7 public class SwingAccumulator extends JFrame {
8     private JTextField tfInput;
9     private JTextField tfOutput;
10    private int sum = 0;
11
12    public SwingAccumulator() {
13        Container cp = getContentPane();
14        cp.setLayout(new GridLayout(2,2));
15
16        cp.add(new Label("CuongPQ 5604 - Enter an Integer: "));
17        tfInput = new JTextField(10);
18        cp.add(tfInput);
19        tfInput.addActionListener(new TFInputListener());
20
21        cp.add(new Label("CuongPQ 5604 - The Accumulated Sum is: "));
22        tfOutput = new JTextField(10);
23        tfOutput.setEditable(false);
24        cp.add(tfOutput);
25
26        setTitle("CuongPQ 5604 - Swing Accumulator");
27        setSize(350, 120);
28        setVisible(true);
29    }
30
31    public static void main(String[] args){
32        new SwingAccumulator();
33    }
34
35    private class TFInputListener implements ActionListener {
36        @Override
37        public void actionPerformed(ActionEvent ewt){
38            int numberIn = Integer.parseInt(tfInput.getText());
39            sum += numberIn;
40            tfInput.setText("");
41            tfOutput.setText(sum + "");
42        }
43    }
44 }
```

Figure 1.3: Source code of SwingAccumulator

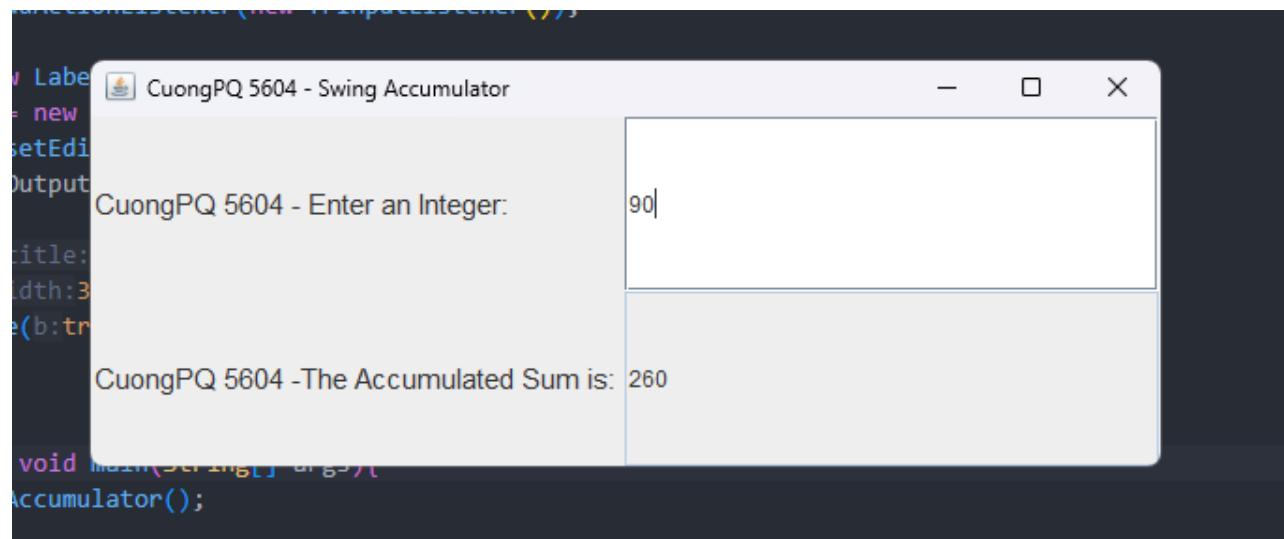
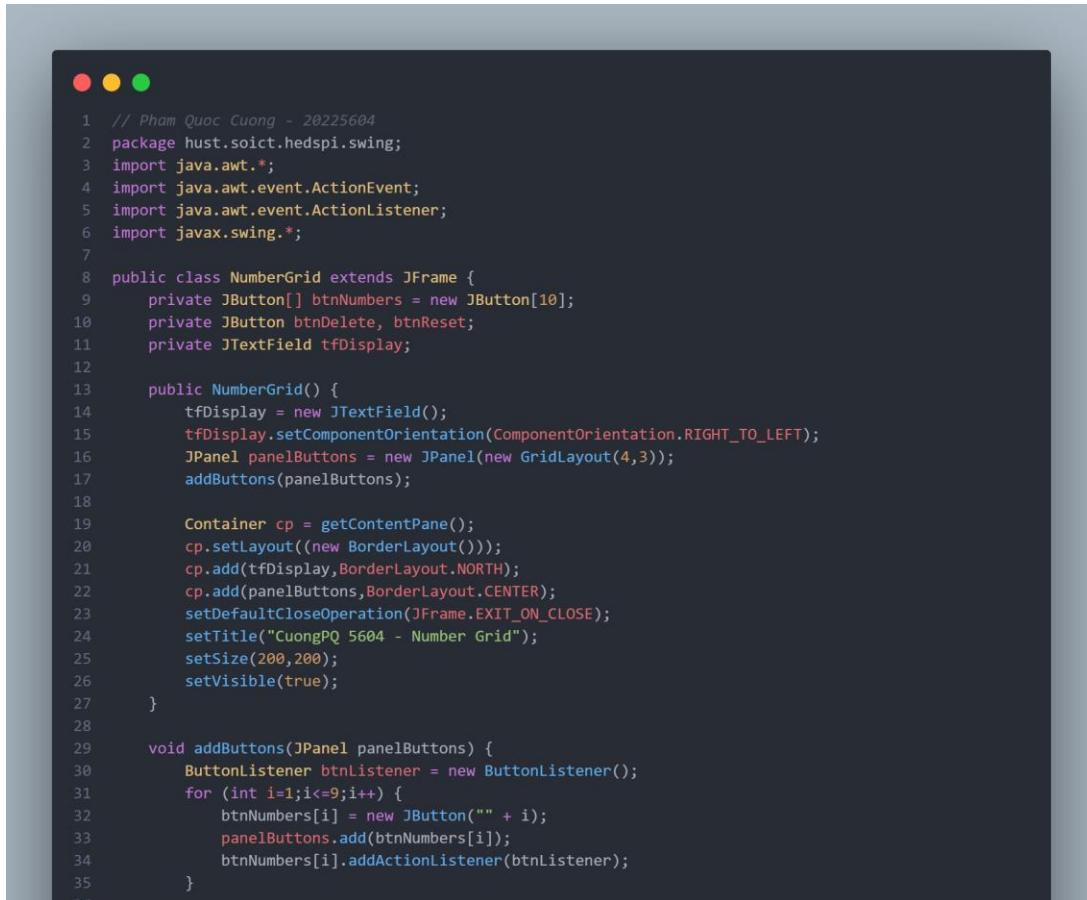


Figure 1.4: Demo of SwingAccumulator

2 Organizing Swing components with Layout Managers

2.1 Code



```
1 // Pham Quoc Cuong - 20225604
2 package hust.soict.hedspi.swing;
3 import java.awt.*;
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6 import javax.swing.*;
7
8 public class NumberGrid extends JFrame {
9     private JButton[] btnNumbers = new JButton[10];
10    private JButton btnDelete, btnReset;
11    private JTextField tfDisplay;
12
13    public NumberGrid() {
14        tfDisplay = new JTextField();
15        tfDisplay.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
16        JPanel panelButtons = new JPanel(new GridLayout(4,3));
17        addButtons(panelButtons);
18
19        Container cp = getContentPane();
20        cp.setLayout((new BorderLayout()));
21        cp.add(tfDisplay,BorderLayout.NORTH);
22        cp.add(panelButtons,BorderLayout.CENTER);
23        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24        setTitle("CuongPQ 5604 - Number Grid");
25        setSize(200,200);
26        setVisible(true);
27    }
28
29    void addButtons(JPanel panelButtons) {
30        ButtonListener btnListener = new ButtonListener();
31        for (int i=1;i<=9;i++) {
32            btnNumbers[i] = new JButton("" + i);
33            panelButtons.add(btnNumbers[i]);
34            btnNumbers[i].addActionListener(btnListener);
35        }
36    }
37}
```

Figure 2.1: Source code of NumberGrid I

```
36
37     btnDelete = new JButton("DEL");
38     panelButtons.add(btnDelete);
39     btnDelete.addActionListener(btnListener);
40
41     btnNumbers[0] = new JButton("0");
42     panelButtons.add(btnNumbers[0]);
43     btnNumbers[0].addActionListener(btnListener);
44
45     btnReset = new JButton("C");
46     panelButtons.add(btnReset);
47     btnReset.addActionListener(btnListener);
48
49 }
50
51 private class ButtonListener implements ActionListener {
52     @Override
53     public void actionPerformed(ActionEvent e) {
54         String button = e.getActionCommand();
55         if(button.charAt(0) >= '0' && button.charAt(0) <= '9') {
56             tfDisplay.setText(tfDisplay.getText() + button);
57         }
58         else if (button.equals("DEL")) {
59
60             tfDisplay.setText(tfDisplay.getText().substring(0,tfDisplay.getText().length() - 1));
61         }
62         else {
63             tfDisplay.setText("");
64         }
65     }
66 }
67
68 public static void main(String[] args) {
69     new NumberGrid();
70 }
71 }
```

Figure 2.2: Source code of NumberGrid 2

2.2 Demo

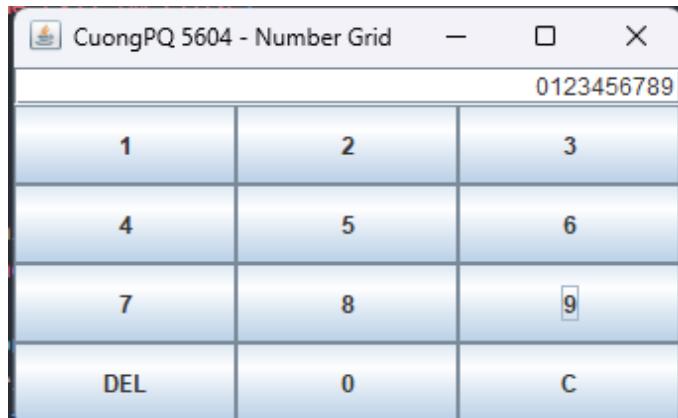


Figure 2.3: Demo buttons 0-9

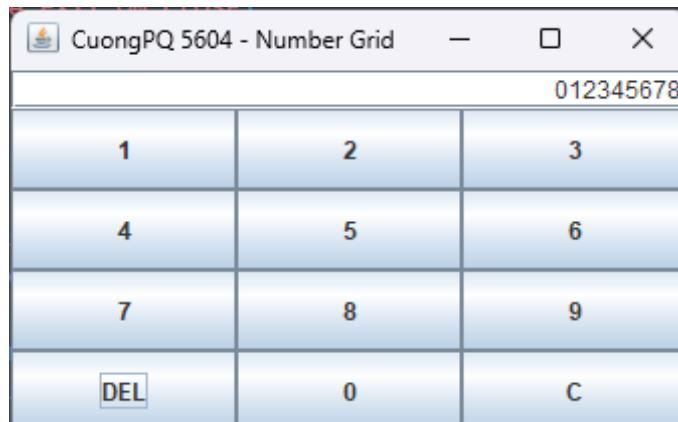


Figure 2.4: Demo DEL button

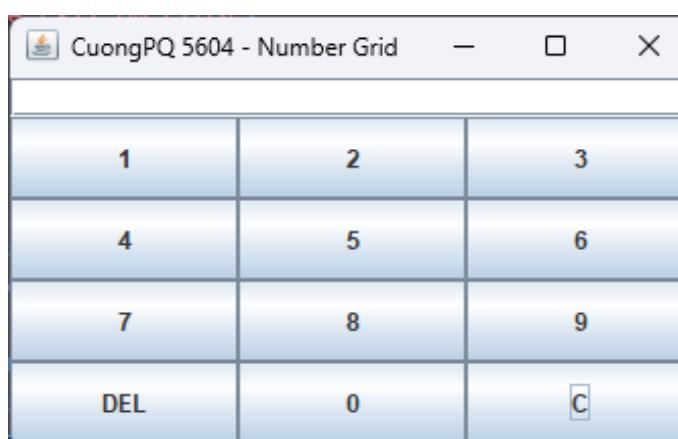


Figure 2.5: Demo C button

3 Create a graphical user interface for AIMS with Swing

3.1 Create class StoreScreen



The screenshot shows a Java code editor with the following code:

```
1 package hust.soict.dsai.aims.screen;
2 import javax.swing.*;
3 import hust.soict.dsai.aims.cart.Cart;
4 import hust.soict.dsai.aims.media.*;
5 import hust.soict.dsai.aims.store.Store;
6 import java.awt.*;
7 import java.awt.event.ActionEvent;
8 import java.awt.event.ActionListener;
9 import java.util.ArrayList;
10
11 public class StoreScreen extends JFrame {
12     private Store store;
13     private Container cp;
14     private Cart cart;
15     JPanel createNorth() {
16         JPanel north = new JPanel();
17         north.setLayout(new BoxLayout(north,BoxLayout.Y_AXIS));
18         north.add(createMenuBar());
19         north.add(createHeader());
20         return north;
21     }
22
23     JMenuBar createMenuBar() {
24         JMenu menu = new JMenu("Options");
25         JMenu smUpdateStore = new JMenu("CuongPQ 225604 - Update Store");
26         JMenuItem addBook = new JMenuItem("CuongPQ 225604 - Add Book");
27         smUpdateStore.add(addBook);
28         addBook.addActionListener(e -> {
29             new AddBookStoreScreen(store);
30         });
31         JMenuItem addCD = new JMenuItem("CuongPQ 225604 - Add CD");
32         smUpdateStore.add(addCD);
33         addCD.addActionListener(e -> {
34             new AddCDStoreScreen(store);
35         });
36         JMenuItem addDVD = new JMenuItem("CuongPQ 225604 - Add DVD");
37         smUpdateStore.add(addDVD);
38         addDVD.addActionListener(e -> {
39             new AddDVDStoreScreen(store);
40         });
41         menu.add(smUpdateStore);
42         JMenuItem viewCart = new JMenuItem("View cart");
43         viewCart.addActionListener(e -> {
44             new CartScreen(cart);
45         });
46         menu.add(viewCart);
47         JMenuBar menuBar = new JMenuBar();
48         menuBar.setLayout(new FlowLayout(FlowLayout.LEFT));
49         menuBar.add(menu);
50         return menuBar;
51     }
}
```

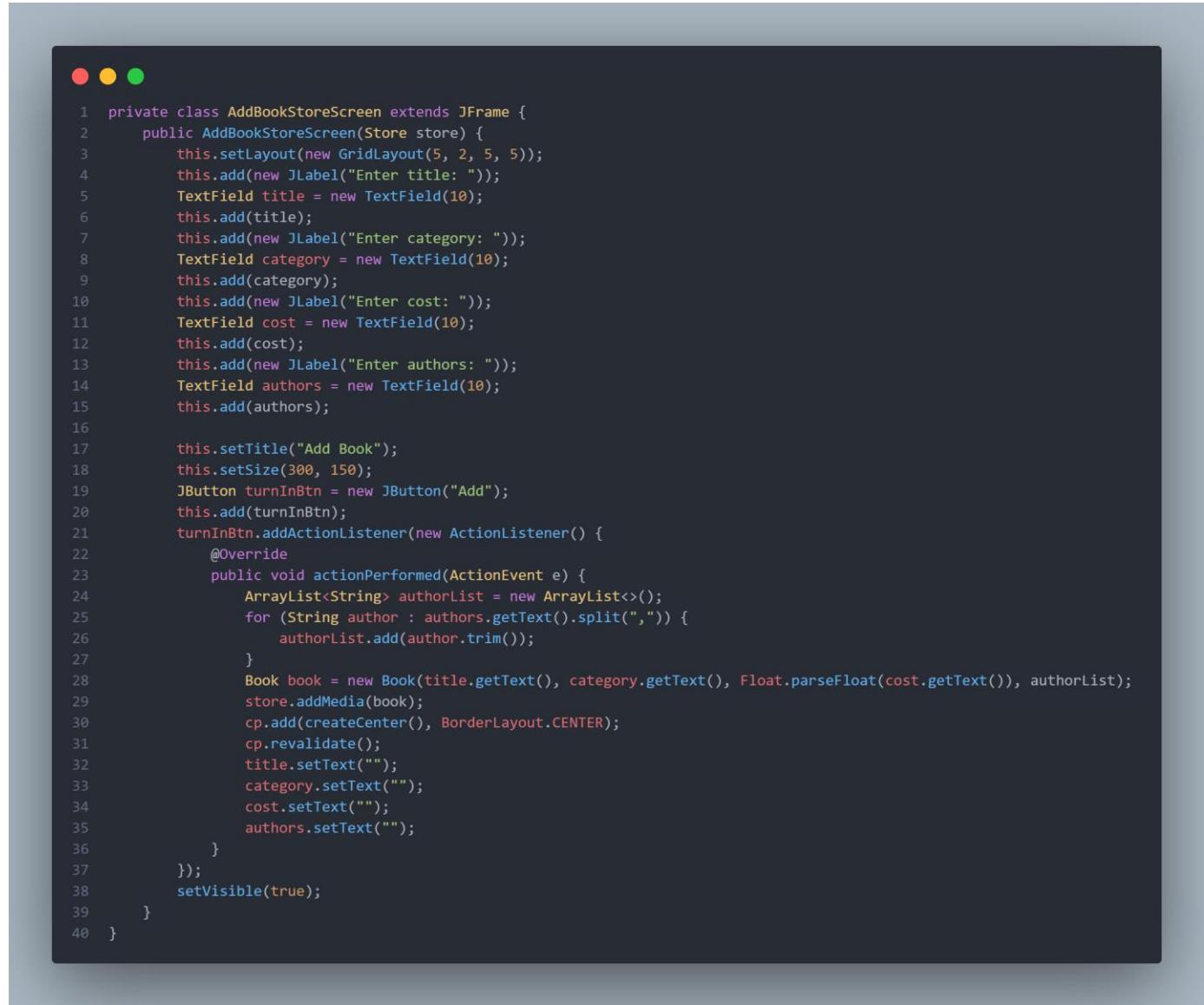
Figure 3.1: Class StoreScreen 1



The screenshot shows a Java code editor with a dark theme. At the top, there are three colored circular icons (red, yellow, green). The code itself is a Java class named `StoreScreen`. It contains methods for creating a header panel, a center panel, and a constructor for the `StoreScreen` class. The code uses various Java Swing components like `JPanel`, `JLabel`, and `JButton`, along with layout managers like `BoxLayout` and `GridLayout`.

```
1 JPanel createHeader() {
2     JPanel header = new JPanel();
3     header.setLayout(new BoxLayout(header,BoxLayout.X_AXIS));
4
5     JLabel title = new JLabel("CuongPQ 225604 - AIMS");
6     title.setFont(new Font(title.getFont().getName(),Font.PLAIN,50));
7     title.setForeground(Color.BLACK);
8
9     JButton cart1 = new JButton("View cart");
10    cart1.setPreferredSize(new Dimension(100,50));
11    cart1.setMaximumSize(new Dimension(100,50));
12    cart1.addActionListener(e -> {
13        new CartScreen(cart);
14    });
15
16    header.add(Box.createRigidArea(new Dimension(10,10)));
17    header.add(title);
18    header.add(Box.createHorizontalGlue());
19    header.add(cart1);
20    header.add(Box.createRigidArea(new Dimension(10,10)));
21    return header;
22 }
23
24 JPanel createCenter() {
25     JPanel center = new JPanel();
26     center.setLayout(new GridLayout(3,3,2,2));
27     ArrayList<Media> mediaStore = store.getItemsInStore();
28     for(Media media : mediaStore) {
29         MediaStore cell = new MediaStore(media,cart);
30         center.add(cell);
31     }
32     return center;
33 }
34
35 public StoreScreen(Store store, Cart myCart) {
36     this.store = store;
37     this.cart = myCart;
38     cp = getContentPane();
39     cp.setLayout(new BorderLayout());
40     cp.add(createNorth(),BorderLayout.NORTH);
41     cp.add(createCenter(),BorderLayout.CENTER);
42     setVisible(true);
43     setTitle("20225604 - Store");
44     setSize(1024,768);
45 }
46 }
```

Figure 3.2: Class `StoreScreen` 2



The screenshot shows a Java code editor with a dark theme. The code is a Java class named `AddBookStoreScreen` which extends `JFrame`. It contains methods for setting up a grid layout with five rows and two columns, adding labels and text fields for title, category, cost, and authors, and a button to add a book. The code also includes an overridden `actionPerformed` method that processes the input from the text fields, creates a `Book` object, adds it to a `ArrayList`, and then revalidates the frame.

```
1 private class AddBookStoreScreen extends JFrame {
2     public AddBookStoreScreen(Store store) {
3         this.setLayout(new GridLayout(5, 2, 5, 5));
4         this.add(new JLabel("Enter title: "));
5         TextField title = new TextField(10);
6         this.add(title);
7         this.add(new JLabel("Enter category: "));
8         TextField category = new TextField(10);
9         this.add(category);
10        this.add(new JLabel("Enter cost: "));
11        TextField cost = new TextField(10);
12        this.add(cost);
13        this.add(new JLabel("Enter authors: "));
14        TextField authors = new TextField(10);
15        this.add(authors);
16
17        this.setTitle("Add Book");
18        this.setSize(300, 150);
19        JButton turnInBtn = new JButton("Add");
20        this.add(turnInBtn);
21        turnInBtn.addActionListener(new ActionListener() {
22             @Override
23             public void actionPerformed(ActionEvent e) {
24                 ArrayList<String> authorList = new ArrayList<>();
25                 for (String author : authors.getText().split(",")) {
26                     authorList.add(author.trim());
27                 }
28                 Book book = new Book(title.getText(), category.getText(), Float.parseFloat(cost.getText()), authorList);
29                 store.addMedia(book);
30                 cp.add(createCenter(), BorderLayout.CENTER);
31                 cp.revalidate();
32                 title.setText("");
33                 category.setText("");
34                 cost.setText("");
35                 authors.setText("");
36             }
37         });
38         setVisible(true);
39     }
40 }
```

Figure 3.3: Class StoreScreen 3

```

1 private class AddDVDStoreScreen extends JFrame {
2     public AddDVDStoreScreen(Store store) {
3         this.setLayout(new GridLayout(4, 2, 5, 5));
4         this.add(new JLabel("Enter title: "));
5         JTextField title = new JTextField(10);
6         this.add(title);
7         this.add(new JLabel("Enter category: "));
8         JTextField category = new JTextField(10);
9         this.add(category);
10        this.add(new JLabel("Enter cost: "));
11        JTextField cost = new JTextField(10);
12        this.add(cost);
13
14        this.setTitle("Add DVD");
15        this.setSize(300, 100);
16        JButton turnInBtn = new JButton("Add");
17        this.add(turnInBtn);
18        turnInBtn.addActionListener(new ActionListener() {
19            @Override
20            public void actionPerformed(ActionEvent e) {
21                DigitalVideoDisc dvd = new DigitalVideoDisc(title.getText(), category.getText(), Float.parseFloat(cost.getText()));
22                store.addMedia(dvd);
23                cp.add(createCenter(), BorderLayout.CENTER);
24                cp.revalidate();
25                title.setText("");
26                category.setText("");
27                cost.setText("");
28            }
29        });
30        setVisible(true);
31    }
32}

```

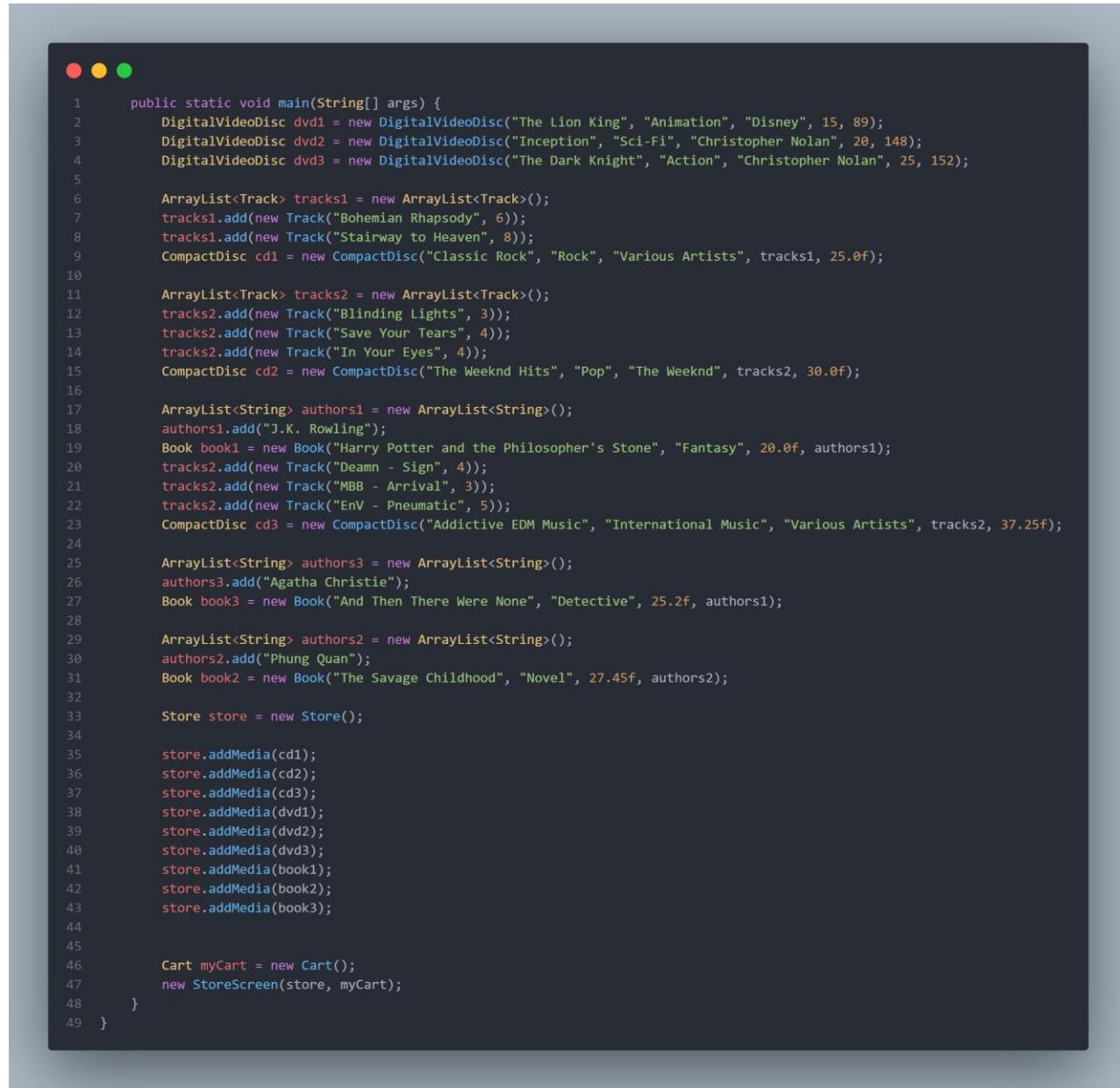
Figure 3.4: Class StoreScreen 4

```

1 private class AddCDStoreScreen extends JFrame {
2     public AddCDStoreScreen(Store store) {
3         this.setLayout(new GridLayout(7, 2, 5, 5));
4         this.add(new JLabel("Enter title: "));
5         JTextField title = new JTextField(10);
6         this.add(title);
7         this.add(new JLabel("Enter category: "));
8         JTextField category = new JTextField(10);
9         this.add(category);
10        this.add(new JLabel("Enter cost: "));
11        JTextField cost = new JTextField(10);
12        this.add(cost);
13        this.add(new JLabel("Enter artist: "));
14        JTextField artist = new JTextField(10);
15        this.add(artist);
16        this.setTitle("Add CD");
17        this.add(new JLabel("Number of tracks: "));
18        JTextField numberOfTracks = new JTextField(10);
19        this.add(numberOfTracks);
20        this.pack();
21        JButton turnInBtn = new JButton("Add");
22        this.add(turnInBtn);
23        turnInBtn.addActionListener(new ActionListener() {
24            @Override
25            public void actionPerformed(ActionEvent e) {
26                CompactDisc cd = new CompactDisc(title.getText(), category.getText(), artist.getText(), new ArrayList<Track>(), Float.parseFloat(cost.getText()));
27                store.addMedia(cd);
28                cp.add(createCenter(), BorderLayout.CENTER);
29                cp.revalidate();
30                title.setText("");
31                category.setText("");
32                cost.setText("");
33            }
34        });
35        setVisible(true);
36    }
37}

```

Figure 3.5: Class StoreScreen 5



The screenshot shows a Java code editor with a dark theme. The code is a class named `StoreScreen` with a main method. It creates several media objects (DigitalVideoDisc, CompactDisc, Book) and adds them to a `Store` object. Finally, it creates a `Cart` and initializes a `StoreScreen` with the store and cart.

```
1 public static void main(String[] args) {
2     DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King", "Animation", "Disney", 15, 89);
3     DigitalVideoDisc dvd2 = new DigitalVideoDisc("Inception", "Sci-Fi", "Christopher Nolan", 20, 148);
4     DigitalVideoDisc dvd3 = new DigitalVideoDisc("The Dark Knight", "Action", "Christopher Nolan", 25, 152);
5
6     ArrayList<Track> tracks1 = new ArrayList<Track>();
7     tracks1.add(new Track("Bohemian Rhapsody", 6));
8     tracks1.add(new Track("Stairway to Heaven", 8));
9     CompactDisc cd1 = new CompactDisc("Classic Rock", "Rock", "Various Artists", tracks1, 25.0f);
10
11    ArrayList<Track> tracks2 = new ArrayList<Track>();
12    tracks2.add(new Track("Blinding Lights", 3));
13    tracks2.add(new Track("Save Your Tears", 4));
14    tracks2.add(new Track("In Your Eyes", 4));
15    CompactDisc cd2 = new CompactDisc("The Weeknd Hits", "Pop", "The Weeknd", tracks2, 30.0f);
16
17    ArrayList<String> authors1 = new ArrayList<String>();
18    authors1.add("J.K. Rowling");
19    Book book1 = new Book("Harry Potter and the Philosopher's Stone", "Fantasy", 20.0f, authors1);
20    tracks2.add(new Track("Deamn - Sign", 4));
21    tracks2.add(new Track("MBB - Arrival", 3));
22    tracks2.add(new Track("EnV - Pneumatic", 5));
23    CompactDisc cd3 = new CompactDisc("Addictive EDM Music", "International Music", "Various Artists", tracks2, 37.25f);
24
25    ArrayList<String> authors3 = new ArrayList<String>();
26    authors3.add("Agatha Christie");
27    Book book3 = new Book("And Then There Were None", "Detective", 25.2f, authors1);
28
29    ArrayList<String> authors2 = new ArrayList<String>();
30    authors2.add("Phung Quan");
31    Book book2 = new Book("The Savage Childhood", "Novel", 27.45f, authors2);
32
33    Store store = new Store();
34
35    store.addMedia(cd1);
36    store.addMedia(cd2);
37    store.addMedia(cd3);
38    store.addMedia(dvd1);
39    store.addMedia(dvd2);
40    store.addMedia(dvd3);
41    store.addMedia(book1);
42    store.addMedia(book2);
43    store.addMedia(book3);
44
45
46    Cart myCart = new Cart();
47    new StoreScreen(store, myCart);
48 }
49 }
```

Figure 3.6: Class `StoreScreen` 6

3.2 Create class MediaStore



```
1 // Pham Quoc Cuong - 20225604
2 package hust.soict.dsai.aims.screen;
3 import hust.soict.dsai.aims.cart.Cart;
4 import hust.soict.dsai.aims.media.*;
5 import javax.swing.*;
6 import java.awt.*;
7
8 public class MediaStore extends JPanel {
9     private Media media;
10
11     public MediaStore(Media media, Cart myCart) {
12         this.media = media;
13         this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
14
15         JLabel title = new JLabel(media.getTitle());
16         title.setFont(new Font(title.getFont().getName(), Font.PLAIN, 20));
17         title.setAlignmentX(CENTER_ALIGNMENT);
18
19         JLabel cost = new JLabel(" " + media.getCost() + "$");
20         cost.setAlignmentX(CENTER_ALIGNMENT);
21
22         JPanel container = new JPanel();
23         container.setLayout(new FlowLayout(FlowLayout.CENTER));
24         JButton btnAdd = new JButton("Add to cart");
25         container.add(btnAdd);
26         btnAdd.addActionListener(e -> myCart.addMedia(media));
27
28         if(media instanceof Playable) {
29             JButton playBtn= new JButton("Play");
30             playBtn.addActionListener(e -> {
31                 JDialog playDialog = createPlayDialog(media);
32                 playDialog.setVisible(true);
33                 playDialog.setSize(300,200);
34                 playDialog.pack();
35             });
36             container.add(playBtn);
37         }
38         this.add(Box.createVerticalGlue());
39         this.add(title);
40         this.add(cost);
41         this.add(Box.createVerticalGlue());
42         this.add(container);
43         this.setBorder(BorderFactory.createLineBorder(Color.BLACK));
44     }
}
```

Figure 3.7: Class MediaStore 1



A screenshot of a Java code editor window. The title bar shows three colored circles (red, yellow, green) at the top left. The main area contains the following Java code:

```
1 static JDialog createPlayDialog(Media media) {
2     JDialog playDialog = new JDialog();
3     Container container = playDialog.getContentPane();
4     playDialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
5     container.setLayout(new BoxLayout(container,BoxLayout.Y_AXIS));
6     container.add(Box.createRigidArea(new Dimension(10,10)));
7     if(media instanceof DigitalVideoDisc dvd) {
8         container.add(new JLabel("Playing DVD:" + dvd.getTitle()));
9         container.add(new JLabel("DVD length:" + dvd.getLength() +" min"));
10    } else if (media instanceof CompactDisc cd) {
11        container.add(new JLabel("Title: " + cd.getTitle()));
12        container.add(new JLabel("Artist: " + cd.getArtist()));
13        for (Track track : cd.getTracks()) {
14            container.add(new JLabel("Play: " + track.getTitle() + ". Length: " + track.getLength() + " min"));
15        }
16    }
17    playDialog.setTitle("Play " + media.getTitle());
18    return playDialog;
19 }
20 }
21 }
```

Figure 3.8: Class MediaStore 2

3.3 Demo

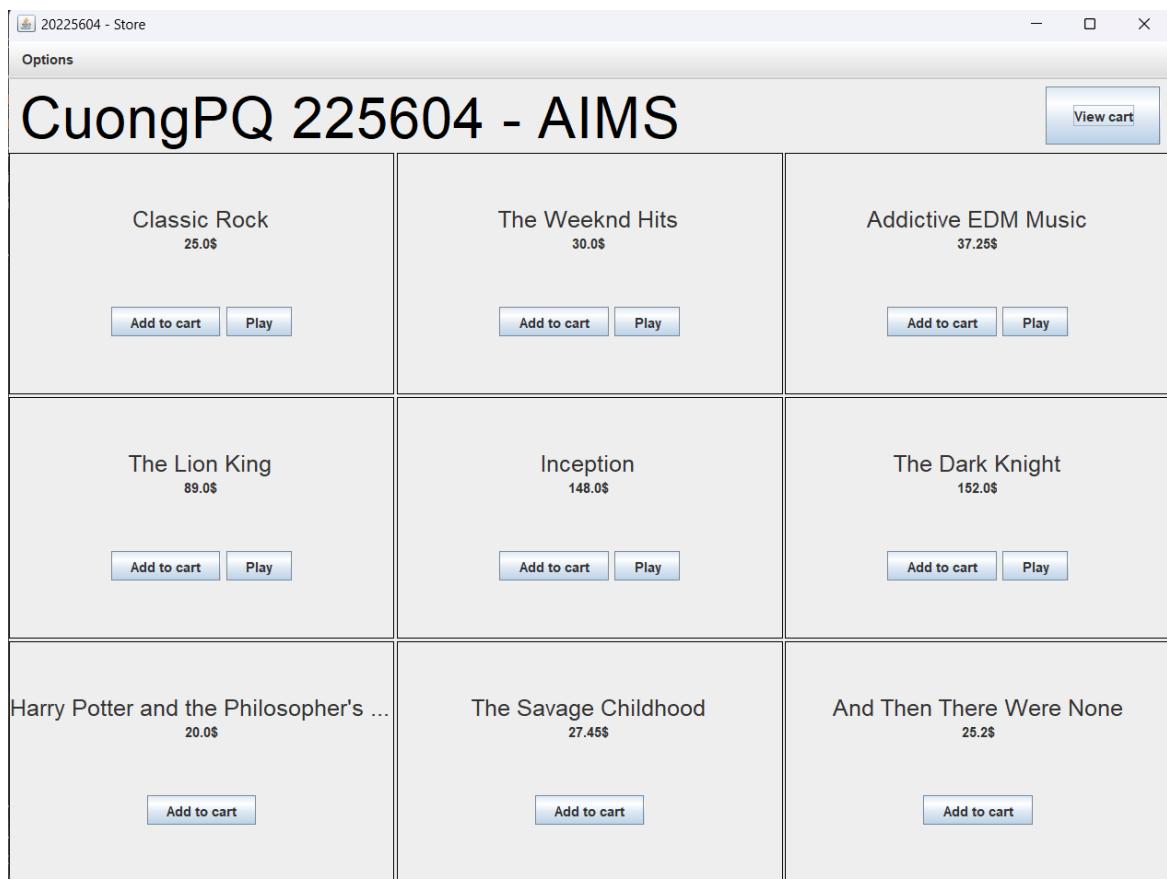


Figure 3.10: StoreScreen

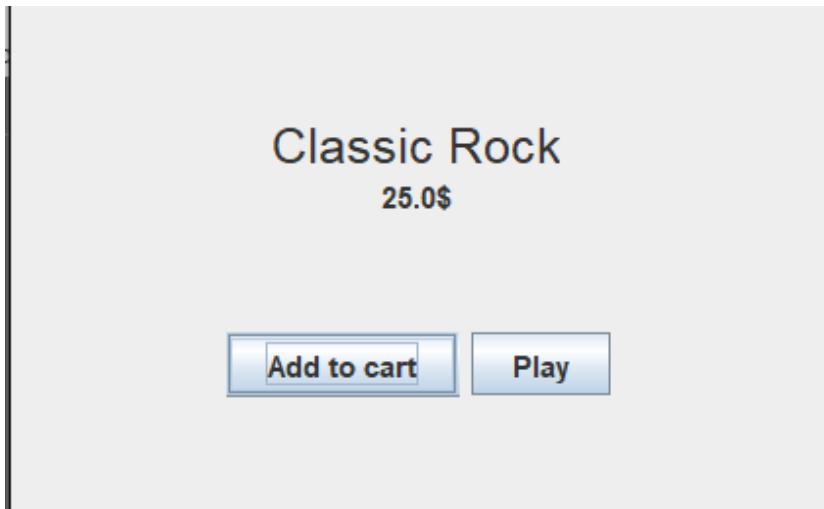


Figure 3.11 Demo Add to cart button

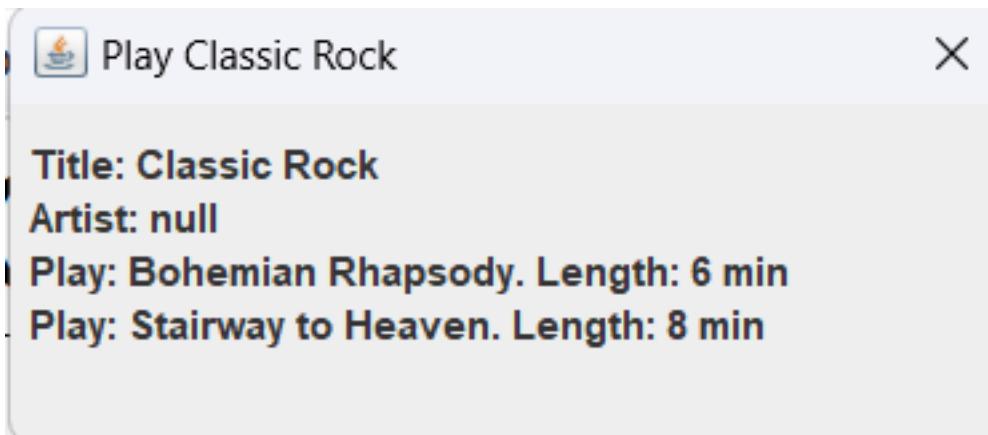


Figure 3.12 Demo Play button

The screenshot shows a "CART" page. At the top, there is a header with a logo, the text "20225604 - Cart", and "Options" buttons. Below the header is a search bar with a placeholder "Filter" and two radio buttons: "By ID" and "By Title". The main area is titled "CART" in large blue letters. Below the title is a table with three columns: "Title", "Category", and "Cost". The table has one visible row for "Classic Rock" under "Rock" category with a cost of "25.0". To the right of the table, the total cost is displayed as "Total: 25.0\$" and a large blue "Place Order" button is shown.

Title	Category	Cost
Classic Rock	Rock	25.0

Figure 3.13 Demo View cart button

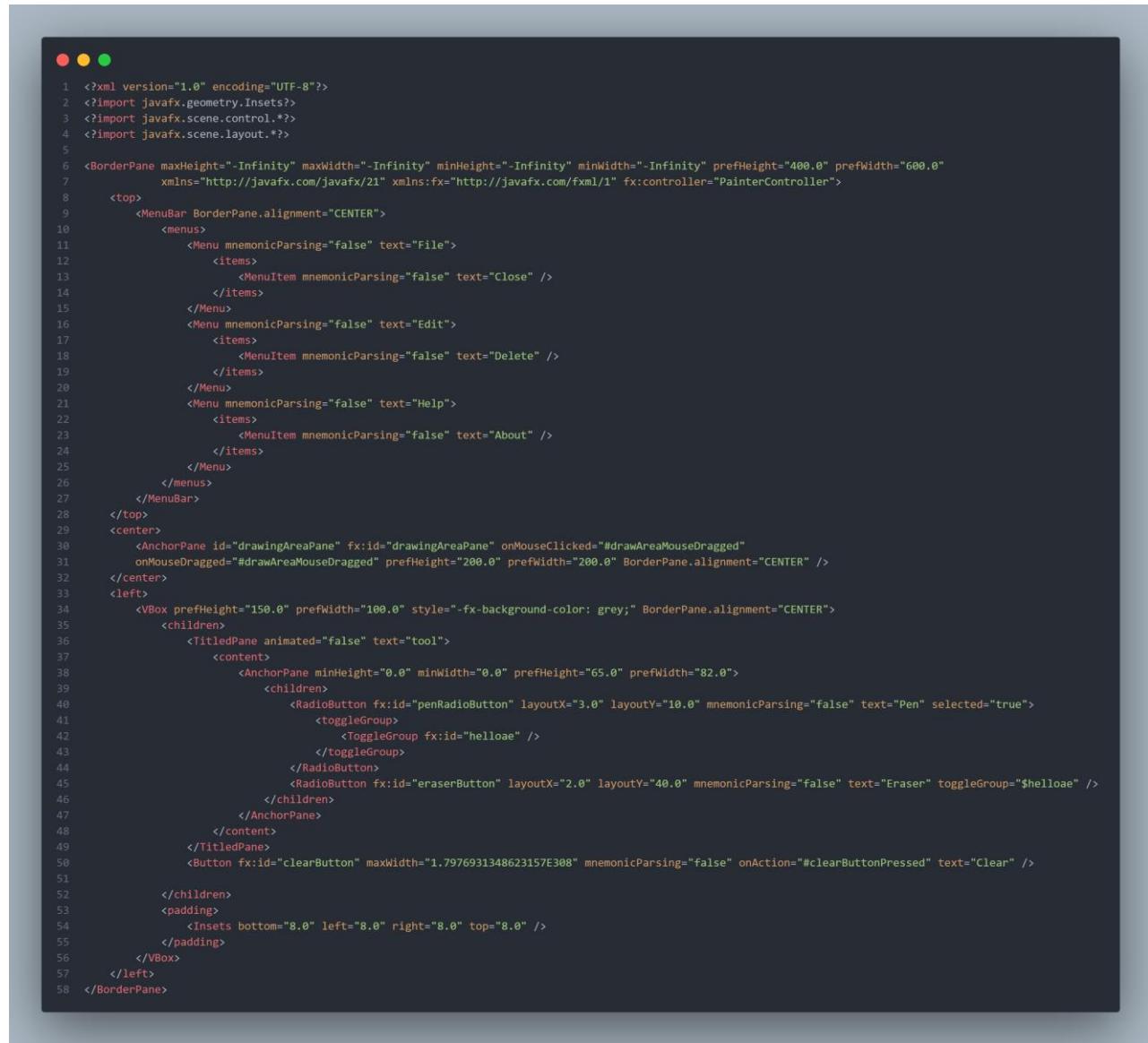
4 JavaFX API

4.1 Create class Painter

```
1 // PhamQuocCuong - 20225604
2 package hust.soict.hedspi.javafx;
3
4 import javafx.application.*;
5 import javafx.fxml.*;
6 import javafx.scene.Parent;
7 import javafx.scene.Scene;
8 import javafx.stage.Stage;
9
10 import java.util.Objects;
11
12 public class Painter extends Application {
13     @Override
14     public void start(Stage stage) throws Exception {
15         try {
16             System.out.println("Loading FXML...");
17             Parent root = FXMLLoader.load(Objects.requireNonNull(getClass().getResource("Painter.fxml")));
18             System.out.println("FXML loaded successfully.");
19             Scene scene = new Scene(root);
20             stage.setTitle("Painter");
21             stage.setScene(scene);
22             stage.show();
23         } catch (Exception e) {
24             System.out.println("Exception: " + e.toString());
25             e.printStackTrace();
26         }
27     }
28
29     public static void main(String[] args) {
30         launch(args);
31     }
32 }
```

Figure 4.1. Painter Class 1

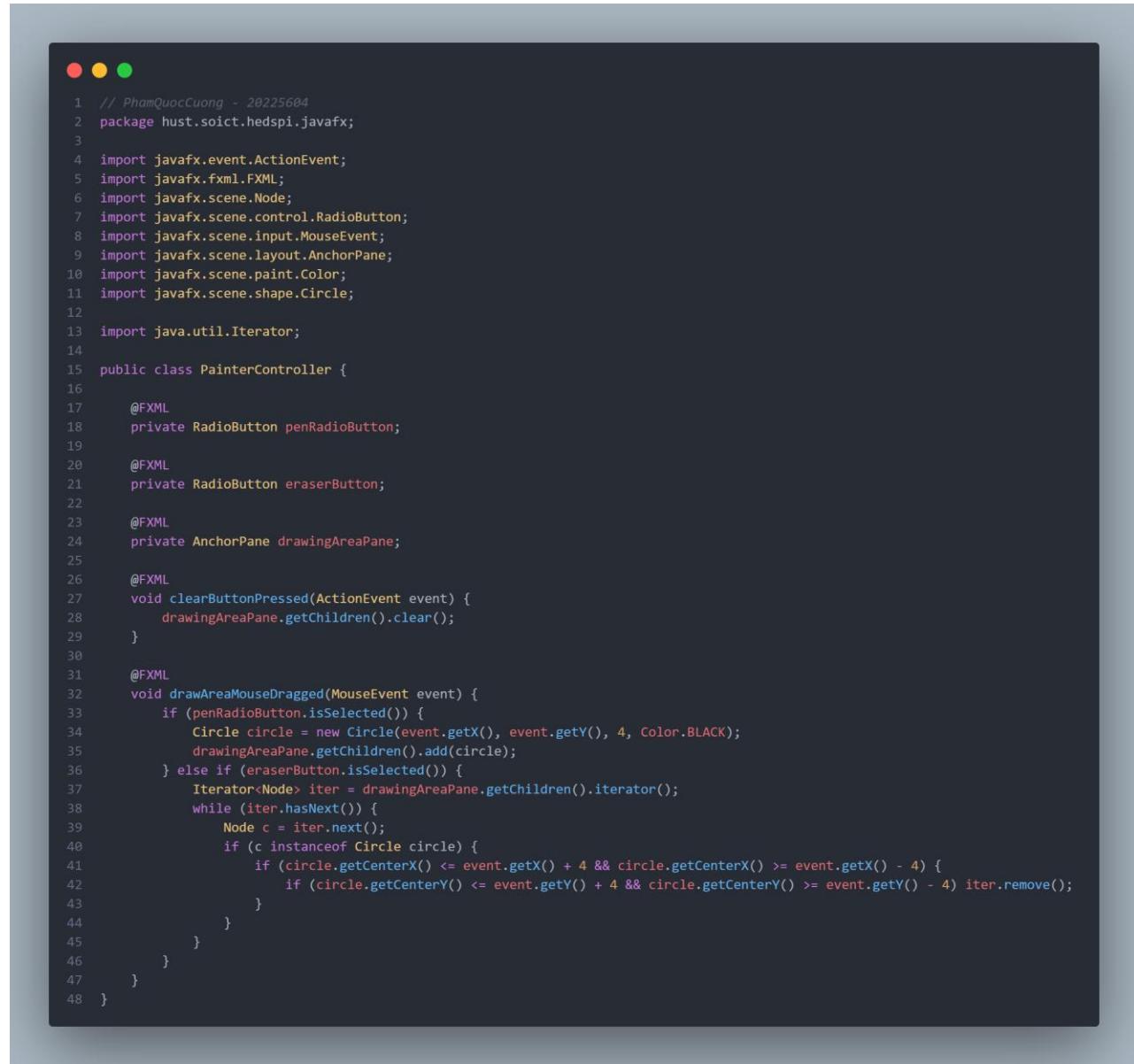
4.2 Create Painter.fxml



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?import javafx.geometry.Insets?>
3 <?import javafx.scene.control.*?>
4 <?import javafx.scene.layout.*?>
5
6 <BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0"
7         xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" fx:controller="PainterController">
8     <top>
9         <MenuBar BorderPane.alignment="CENTER">
10            <menus>
11                <Menu mnemonicParsing="false" text="File">
12                    <items>
13                        <MenuItem mnemonicParsing="false" text="Close" />
14                    </items>
15                </Menu>
16                <Menu mnemonicParsing="false" text="Edit">
17                    <items>
18                        <MenuItem mnemonicParsing="false" text="Delete" />
19                    </items>
20                </Menu>
21                <Menu mnemonicParsing="false" text="Help">
22                    <items>
23                        <MenuItem mnemonicParsing="false" text="About" />
24                    </items>
25                </Menu>
26            </menus>
27        </MenuBar>
28    </top>
29    <center>
30        <AnchorPane id="drawingAreaPane" fx:id="drawingAreaPane" onMouseClicked="#drawAreaMouseDragged"
31             onMouseDragged="#drawAreaMouseDragged" prefHeight="200.0" prefWidth="200.0" BorderPane.alignment="CENTER" />
32    </center>
33    <left>
34        <VBox prefHeight="150.0" prefWidth="100.0" style="-fx-background-color: grey;" BorderPane.alignment="CENTER">
35            <children>
36                <TitledPane animated="false" text="tool">
37                    <content>
38                        <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="65.0" prefWidth="82.0">
39                            <children>
40                                <RadioButton fx:id="penRadioButton" layoutX="3.0" layoutY="10.0" mnemonicParsing="false" text="Pen" selected="true">
41                                    <toggleGroup>
42                                        <ToggleGroup fx:id="helloae" />
43                                    </toggleGroup>
44                                </RadioButton>
45                                <RadioButton fx:id="eraserButton" layoutX="2.0" layoutY="40.0" mnemonicParsing="false" text="Eraser" toggleGroup="$helloae" />
46                            </children>
47                        </AnchorPane>
48                    </content>
49                </TitledPane>
50                <Button fx:id="clearButton" maxWidth="1.7976931348623157E308" mnemonicParsing="false" onAction="#clearButtonPressed" text="Clear" />
51            <children>
52                <padding>
53                    <Insets bottom="8.0" left="8.0" right="8.0" top="8.0" />
54                </padding>
55            </children>
56        </VBox>
57    </left>
58 </BorderPane>
```

Figure 4.2: Painter.fxml 1

4.3 Create class PainterController



The screenshot shows a Java code editor with a dark theme. The code is written in Java and defines a class named PainterController. The code includes imports for various JavaFX classes and methods, and it contains two methods: clearButtonPressed and drawAreaMouseDragged. The drawAreaMouseDragged method uses an iterator to remove nodes from a drawing area pane if they are within a certain distance from the mouse event.

```
1 // PhamQuocCuong - 20225604
2 package hust.soict.hedspi.javafx;
3
4 import javafx.event.ActionEvent;
5 import javafx.fxml.FXML;
6 import javafx.scene.Node;
7 import javafx.scene.control.RadioButton;
8 import javafx.scene.input.MouseEvent;
9 import javafx.scene.layout.AnchorPane;
10 import javafx.scene.paint.Color;
11 import javafx.scene.shape.Circle;
12
13 import java.util.Iterator;
14
15 public class PainterController {
16
17     @FXML
18     private RadioButton penRadioButton;
19
20     @FXML
21     private RadioButton eraserButton;
22
23     @FXML
24     private AnchorPane drawingAreaPane;
25
26     @FXML
27     void clearButtonPressed(ActionEvent event) {
28         drawingAreaPane.getChildren().clear();
29     }
30
31     @FXML
32     void drawAreaMouseDragged(MouseEvent event) {
33         if (penRadioButton.isSelected()) {
34             Circle circle = new Circle(event.getX(), event.getY(), 4, Color.BLACK);
35             drawingAreaPane.getChildren().add(circle);
36         } else if (eraserButton.isSelected()) {
37             Iterator<Node> iter = drawingAreaPane.getChildren().iterator();
38             while (iter.hasNext()) {
39                 Node c = iter.next();
40                 if (c instanceof Circle circle) {
41                     if (circle.getCenterX() <= event.getX() + 4 && circle.getCenterX() >= event.getX() - 4) {
42                         if (circle.getCenterY() <= event.getY() + 4 && circle.getCenterY() >= event.getY() - 4) iter.remove();
43                     }
44                 }
45             }
46         }
47     }
48 }
```

Figure 4.4: PainterControl

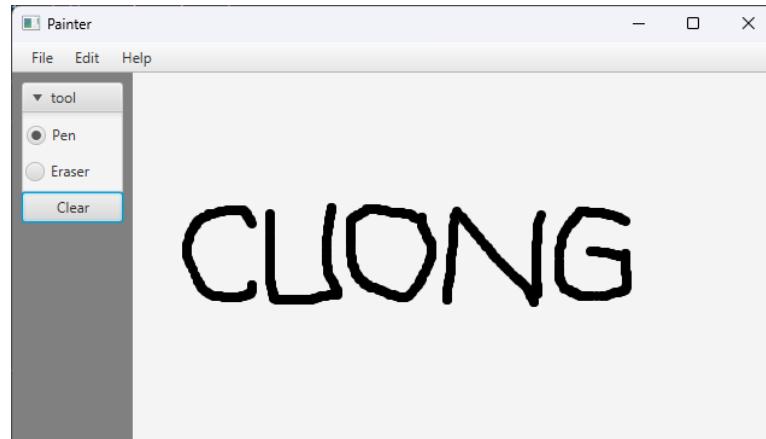


Figure 4.5: Use Pen

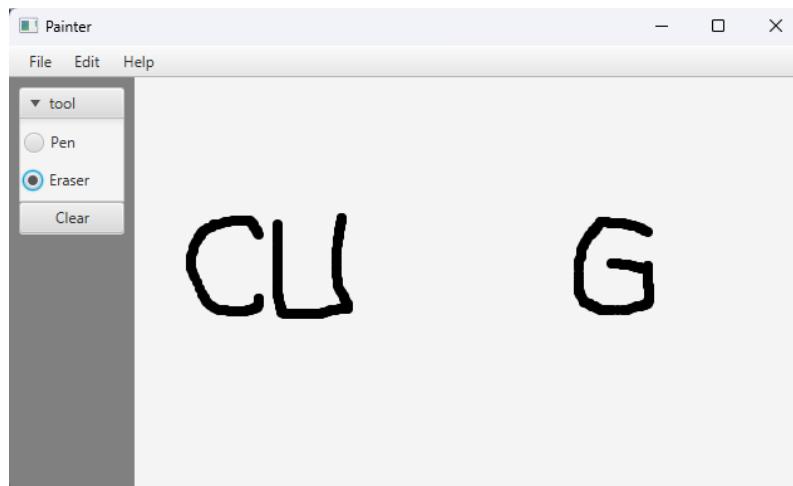


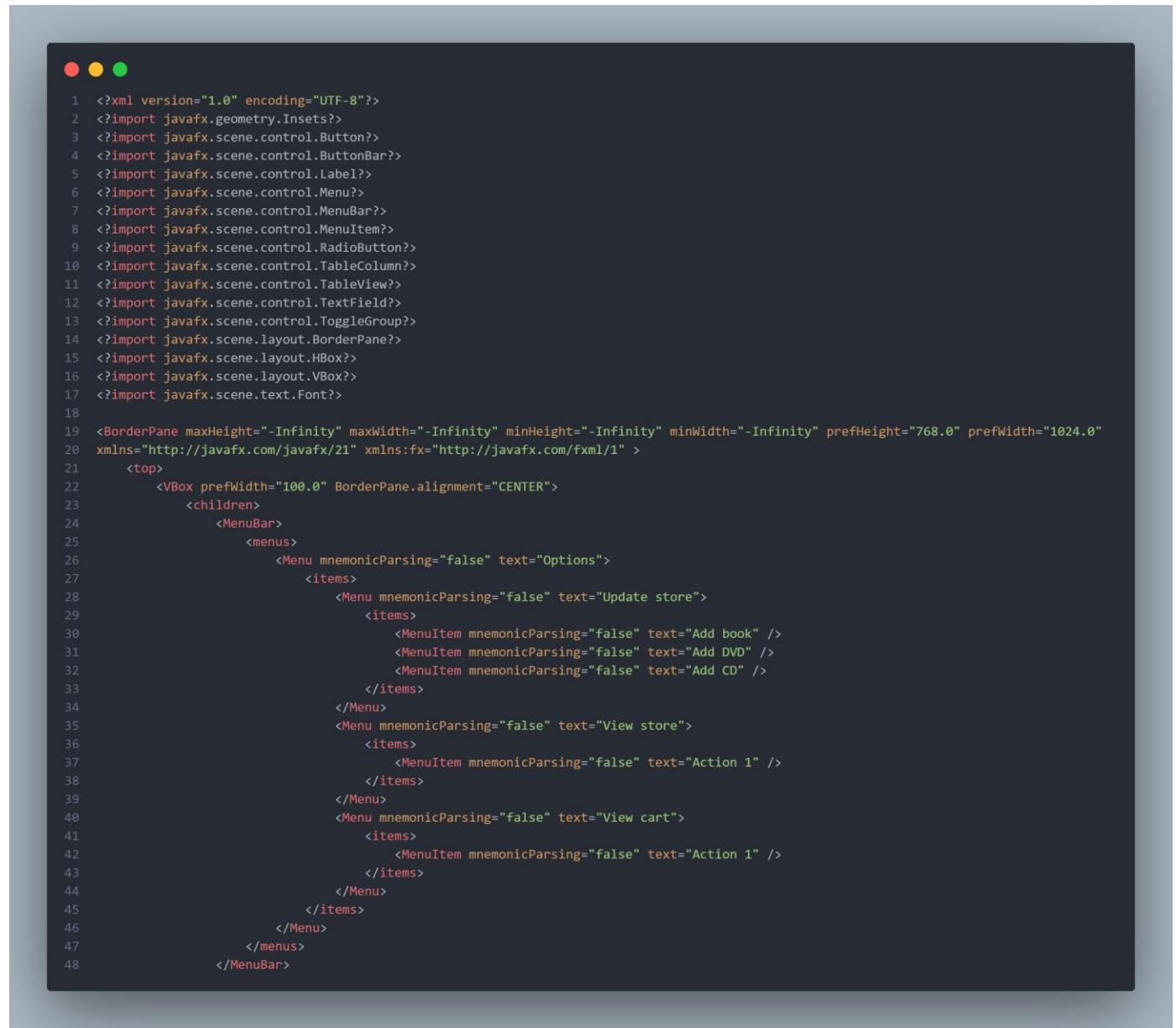
Figure 4.6: Use Eraser



Figure 4.7: Clear button

5 View Cart Screen

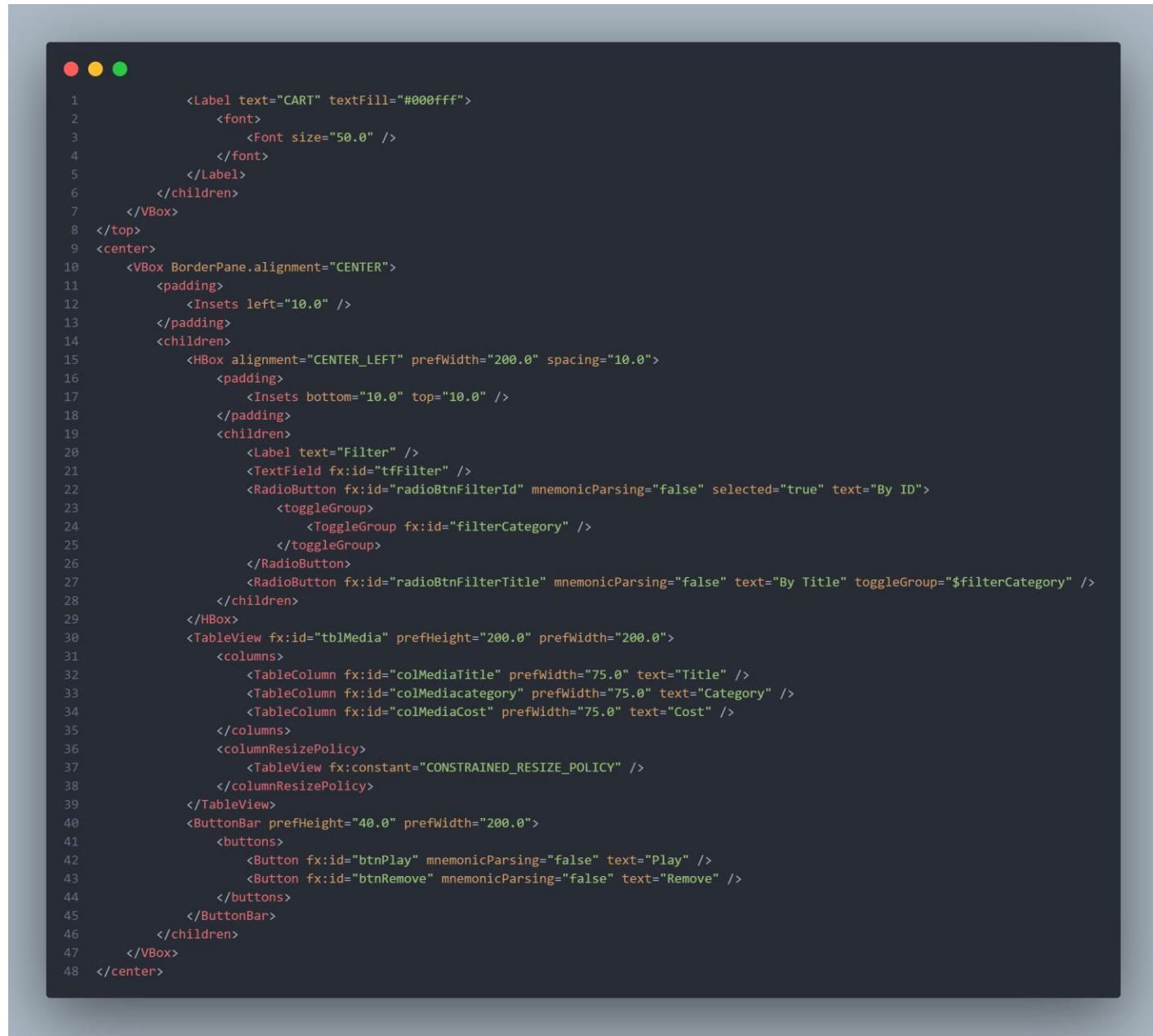
5.1 Create cart.fxml



The screenshot shows a JavaFX application window with three colored window control buttons (red, yellow, green) at the top left. The main content area displays the XML code for the `cart.fxml` file. The code defines a `BorderPane` with a `MenuBar` at the top containing three `Menu` items: "Options", "Update store", and "View cart". The "Update store" menu has three `MenuItem` children: "Add book", "Add DVD", and "Add CD". The "View cart" menu has one `MenuItem`: "Action 1". The `BorderPane` has a `VBox` with a `BorderPane.alignment="CENTER"` set to its center.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?import javafx.geometry.Insets?>
3 <?import javafx.scene.control.Button?>
4 <?import javafx.scene.control.ButtonBar?>
5 <?import javafx.scene.control.Label?>
6 <?import javafx.scene.control.Menu?>
7 <?import javafx.scene.control.MenuBar?>
8 <?import javafx.scene.control.MenuItem?>
9 <?import javafx.scene.control.RadioButton?>
10 <?import javafx.scene.control.TableColumn?>
11 <?import javafx.scene.control.TableView?>
12 <?import javafx.scene.control.TextField?>
13 <?import javafx.scene.control.ToggleGroup?>
14 <?import javafx.scene.layout.BorderPane?>
15 <?import javafx.scene.layout.HBox?>
16 <?import javafx.scene.layout.VBox?>
17 <?import javafx.scene.text.Font?>
18
19 <BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="768.0" prefWidth="1024.0"
20 xmlns="http://javafx.com/javafx/21" xmlns:fx="http://javafx.com/fxml/1" >
21     <top>
22         <VBox prefWidth="100.0" BorderPane.alignment="CENTER">
23             <children>
24                 <MenuBar>
25                     <menus>
26                         <Menu mnemonicParsing="false" text="Options">
27                             <items>
28                                 <Menu mnemonicParsing="false" text="Update store">
29                                     <items>
30                                         <MenuItem mnemonicParsing="false" text="Add book" />
31                                         <MenuItem mnemonicParsing="false" text="Add DVD" />
32                                         <MenuItem mnemonicParsing="false" text="Add CD" />
33                                     </items>
34                                 </Menu>
35                         <Menu mnemonicParsing="false" text="View store">
36                             <items>
37                                 <MenuItem mnemonicParsing="false" text="Action 1" />
38                             </items>
39                         </Menu>
40                         <Menu mnemonicParsing="false" text="View cart">
41                             <items>
42                                 <MenuItem mnemonicParsing="false" text="Action 1" />
43                             </items>
44                         </Menu>
45                     </items>
46                 </MenuBar>
47             </menus>
48         </MenuBar>
```

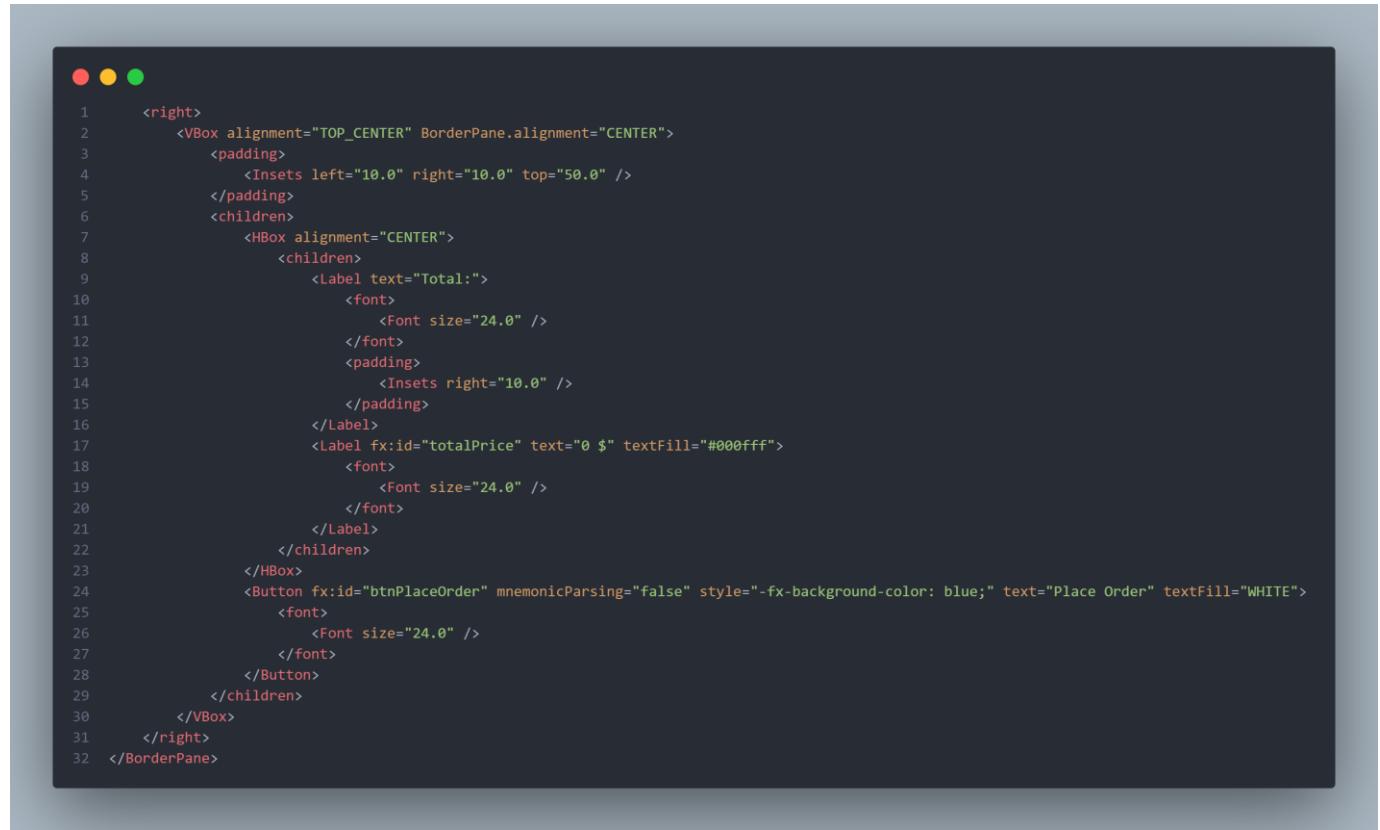
Figure 5.1: Cart.fxml I



The screenshot shows a JavaFX FXML code editor window. The title bar has three colored circles (red, yellow, green) at the top left. The main area displays the XML code for the `Cart.fxml` file. The code defines a user interface with a central `Label` for "CART", a `Table View` for media items, and a `Button Bar` with "Play" and "Remove" buttons.

```
1      <Label text="CART" textFill="#000fff">
2          <font>
3              <Font size="50.0" />
4          </font>
5      </Label>
6  </children>
7 </VBox>
8 </top>
9 <center>
10     <VBox BorderPane.alignment="CENTER">
11         <padding>
12             <Insets left="10.0" />
13         </padding>
14         <children>
15             <HBox alignment="CENTER_LEFT" prefWidth="200.0" spacing="10.0">
16                 <padding>
17                     <Insets bottom="10.0" top="10.0" />
18                 </padding>
19                 <children>
20                     <Label text="Filter" />
21                     <TextField fx:id="tfFilter" />
22                     <RadioButton fx:id="radioBtnFilterId" mnemonicParsing="false" selected="true" text="By ID">
23                         <toggleGroup>
24                             <ToggleGroup fx:id="filterCategory" />
25                         </toggleGroup>
26                     </RadioButton>
27                     <RadioButton fx:id="radioBtnFilterTitle" mnemonicParsing="false" text="By Title" toggleGroup="$filterCategory" />
28                 </children>
29             </HBox>
30             <TableView fx:id="tblMedia" prefHeight="200.0" prefWidth="200.0">
31                 <columns>
32                     <TableColumn fx:id="colMediaTitle" prefWidth="75.0" text="Title" />
33                     <TableColumn fx:id="colMediaCategory" prefWidth="75.0" text="Category" />
34                     <TableColumn fx:id="colMediaCost" prefWidth="75.0" text="Cost" />
35                 </columns>
36                 <columnResizePolicy>
37                     <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
38                 </columnResizePolicy>
39             </TableView>
40             <ButtonBar prefHeight="40.0" prefWidth="200.0">
41                 <buttons>
42                     <Button fx:id="btnPlay" mnemonicParsing="false" text="Play" />
43                     <Button fx:id="btnRemove" mnemonicParsing="false" text="Remove" />
44                 </buttons>
45             </ButtonBar>
46         </children>
47     </VBox>
48 </center>
```

Figure 5.2: Cart.fxml 2

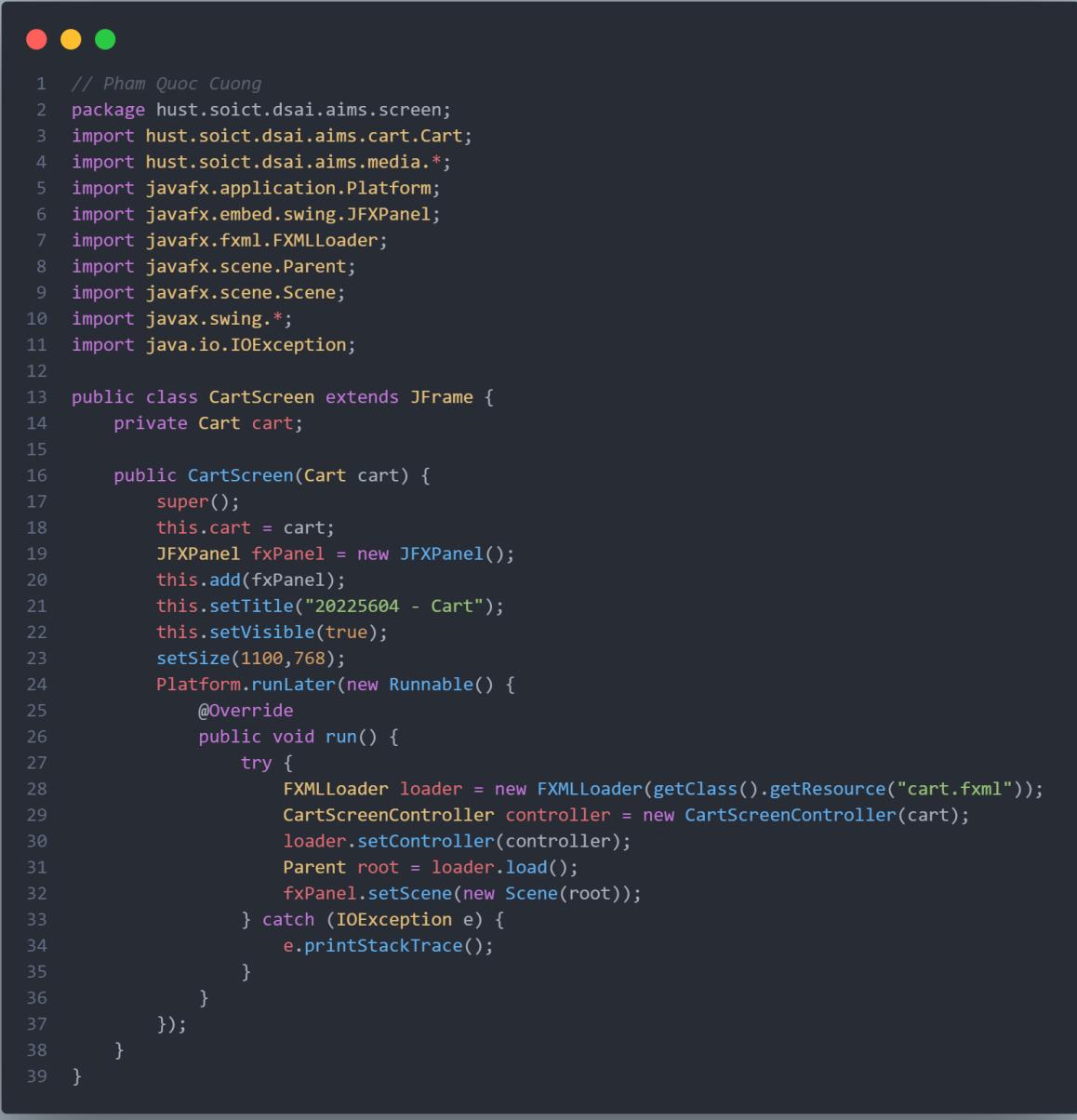


The screenshot shows a JavaFX FXML code editor window. The code is a snippet of FXML for a 'Cart' screen. It defines a 'BorderPane' with a 'right' side containing a 'VBox'. This 'VBox' has a 'padding' of 10.0 for left, right, and top, and 50.0 for bottom. It contains an 'HBox' with a 'Label' for 'Total:' and a 'Label' with 'fx:id="totalPrice"' containing '\$0'. Below these is a 'Button' with 'fx:id="btnPlaceOrder"' labeled 'Place Order'. The 'BorderPane' also has a 'padding' of 10.0 for all sides.

```
<right>
    <VBox alignment="TOP_CENTER" BorderPane.alignment="CENTER">
        <padding>
            <Insets left="10.0" right="10.0" top="50.0" />
        </padding>
        <children>
            <HBox alignment="CENTER">
                <children>
                    <Label text="Total:">
                        <font>
                            <Font size="24.0" />
                        </font>
                    </Label>
                    <Label fx:id="totalPrice" text="0 $" textFill="#000fff">
                        <font>
                            <Font size="24.0" />
                        </font>
                    </Label>
                </children>
            </HBox>
            <Button fx:id="btnPlaceOrder" mnemonicParsing="false" style="-fx-background-color: blue;" text="Place Order" textFill="WHITE">
                <font>
                    <Font size="24.0" />
                </font>
            </Button>
        </children>
    </VBox>
</right>
</BorderPane>
```

Figure 5.3: Cart.fxml 3

5.2 Create class CartScreen



The screenshot shows a Java code editor with a dark theme. The window title bar has three colored circles (red, yellow, green) in the top-left corner. The code itself is a Java class named `CartScreen` which extends `JFrame`. It imports various JavaFX and Swing components. The constructor initializes a `JFXPanel`, sets the title to "20225604 - Cart", and makes the frame visible. It then runs a `Runnable` task on the JavaFX thread to load an FXML file named `cart.fxml` using an `FXMLLoader`. A controller object is created and set to the loader. The root node of the FXML is loaded into the panel, and the scene is set to the frame. If an `IOException` occurs during loading, it is caught and its stack trace is printed.

```
1 // Pham Quoc Cuong
2 package hust.soict.dsai.aims.screen;
3 import hust.soict.dsai.aims.cart.Cart;
4 import hust.soict.dsai.aims.media.*;
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 javax.swing.*;
11 import java.io.IOException;
12
13 public class CartScreen extends JFrame {
14     private Cart cart;
15
16     public CartScreen(Cart cart) {
17         super();
18         this.cart = cart;
19         JFXPanel fxPanel = new JFXPanel();
20         this.add(fxPanel);
21         this.setTitle("20225604 - Cart");
22         this.setVisible(true);
23         setSize(1100,768);
24         Platform.runLater(new Runnable() {
25             @Override
26             public void run() {
27                 try {
28                     FXMLLoader loader = new FXMLLoader(getClass().getResource("cart.fxml"));
29                     CartScreenController controller = new CartScreenController(cart);
30                     loader.setController(controller);
31                     Parent root = loader.load();
32                     fxPanel.setScene(new Scene(root));
33                 } catch (IOException e) {
34                     e.printStackTrace();
35                 }
36             }
37         });
38     }
39 }
```

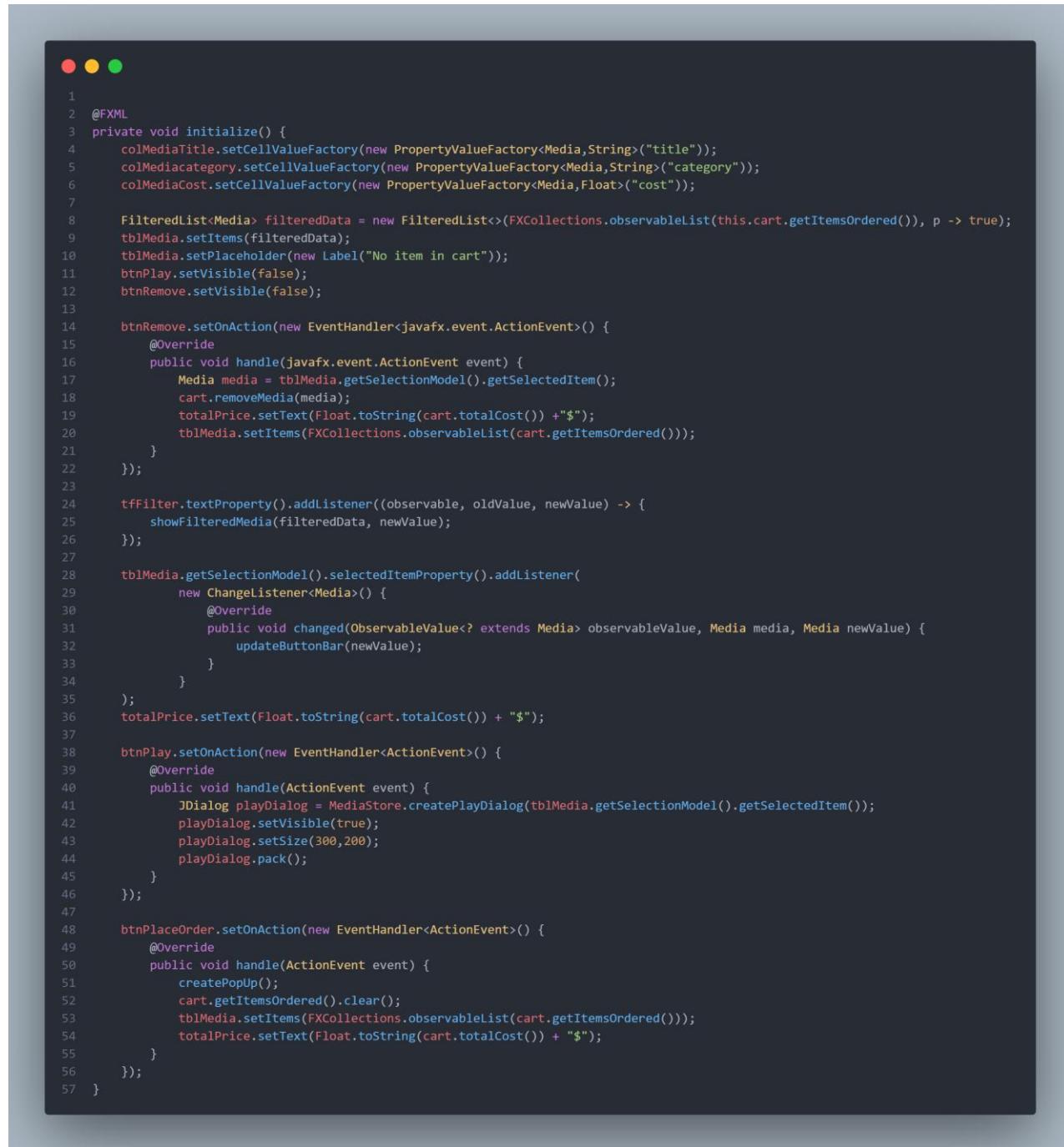
Figure 5.4: `CartScreen` class

5.3 Create class CartScreenController



```
1 package hust.soict.dsai.aims.screen;
2 import hust.soict.dsai.aims.cart.Cart;
3 import hust.soict.dsai.aims.media.*;
4 import javafx.beans.value.ChangeListener;
5 import javafx.beans.value.ObservableValue;
6 import javafx.collections.FXCollections;
7 import javafx.collections.transformation.FilteredList;
8 import javafx.event.ActionEvent;
9 import javafx.event.EventHandler;
10 import javafx.fxml.FXML;
11 import javafx.fxml.FXMLLoader;
12 import javafx.geometry.Pos;
13 import javafx.scene.Parent;
14 import javafx.scene.Scene;
15 import javafx.scene.control.*;
16 import javafx.scene.control.Button;
17 import javafx.scene.control.Label;
18 import javafx.scene.control.TextField;
19 import javafx.scene.control.cell.PropertyValueFactory;
20 import javafx.scene.layout.VBox;
21 import javafx.scene.paint.Color;
22 import javafx.scene.paint.*;
23 import javafx.scene.text.Font;
24 import javafx.scene.text.FontWeight;
25 import javafx.stage.Modality;
26 import javafx.stage.Stage;
27 import javax.swing.*;
28 import java.awt.*;
29 import java.util.ArrayList;
30
31 public class CartScreenController {
32     private Cart cart;
33     @FXML
34     private Button btnPlaceOrder;
35     @FXML
36     private TextField tfFilter;
37     @FXML
38     private ToggleGroup filterCategory;
39     @FXML
40     private RadioButton radioBtnFilterId;
41     @FXML
42     private RadioButton radioBtnFilterTitle;
43     @FXML
44     private Button btnPlay;
45     @FXML
46     private Button btnRemove;
47     @FXML
48     private TableView<Media> tblMedia;
49     @FXML
50     private TableColumn<Media, String> colMediaTitle;
51     @FXML
52     private TableColumn<Media, String> colMediacategory;
53     @FXML
54     private TableColumn<Media,Float> colMediaCost;
55     @FXML
56     private Label totalPrice;
57
58     public CartScreenController(Cart cart) {
59         super();
60         this.cart = cart;
61     }
}
```

Figure 5.5: CartScreenController I



The screenshot shows a JavaFX application window with a dark theme. The title bar has three colored window control buttons (red, yellow, green) at the top left. The main area contains the Java code for `CartScreenController`. The code is annotated with line numbers from 1 to 57. It includes imports for `javafx.event.ActionEvent`, `javafx.collections.FXCollections`, `javafx.scene.control.Button`, `javafx.scene.control.TableColumn`, `javafx.scene.control.TableView`, `javafx.scene.control.TextField`, `javafx.scene.layout.HBox`, `javafx.scene.layout.VBox`, and `javafx.util.Pair`. The code initializes a `TableView` (`tblMedia`) with columns for title, category, and cost. It sets up a `FilteredList` (`filteredData`) to filter items based on a search term (`tfFilter`). It handles item selection changes in the table and removes selected items from a shopping cart (`cart`). It also handles button events for playing media and placing orders. The total price is displayed in a `TextField` (`totalPrice`).

```
1
2  @FXML
3  private void initialize() {
4      colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>("title"));
5      colMediaCategory.setCellValueFactory(new PropertyValueFactory<Media, String>("category"));
6      colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, Float>("cost"));
7
8      FilteredList<Media> filteredData = new FilteredList<>(FXCollections.observableList(this.cart.getItemsOrdered()), p -> true);
9      tblMedia.setItems(filteredData);
10     tblMedia.setPlaceholder(new Label("No item in cart"));
11     btnPlay.setVisible(false);
12     btnRemove.setVisible(false);
13
14     btnRemove.setOnAction(new EventHandler<javafx.event.ActionEvent>() {
15         @Override
16         public void handle(javafx.event.ActionEvent event) {
17             Media media = tblMedia.getSelectionModel().getSelectedItem();
18             cart.removeMedia(media);
19             totalPrice.setText(Float.toString(cart.totalCost()) + "$");
20             tblMedia.setItems(FXCollections.observableList(cart.getItemsOrdered()));
21         }
22     });
23
24     tfFilter.textProperty().addListener((observable, oldValue, newValue) -> {
25         showFilteredMedia(filteredData, newValue);
26     });
27
28     tblMedia.getSelectionModel().selectedItemProperty().addListener(
29         new ChangeListener<Media>() {
30             @Override
31             public void changed(ObservableValue<? extends Media> observableValue, Media media, Media newValue) {
32                 updateButtonBar(newValue);
33             }
34         });
35     totalPrice.setText(Float.toString(cart.totalCost()) + "$");
36
37     btnPlay.setOnAction(new EventHandler<ActionEvent>() {
38         @Override
39         public void handle(ActionEvent event) {
40             JDialog playDialog = MediaStore.createPlayDialog(tblMedia.getSelectionModel().getSelectedItem());
41             playDialog.setVisible(true);
42             playDialog.setSize(300,200);
43             playDialog.pack();
44         }
45     });
46
47     btnPlaceOrder.setOnAction(new EventHandler<ActionEvent>() {
48         @Override
49         public void handle(ActionEvent event) {
50             createPopUp();
51             cart.getItemsOrdered().clear();
52             tblMedia.setItems(FXCollections.observableList(cart.getItemsOrdered()));
53             totalPrice.setText(Float.toString(cart.totalCost()) + "$");
54         }
55     });
56 };
57 }
```

Figure 5.6: CartScreenController 2

```
1      @FXML
2      void updateButtonBar(Media media) {
3          btnRemove.setVisible(true);
4          if(media instanceof Playable) {
5              btnPlay.setVisible(true);
6          } else {
7              btnPlay.setVisible(false);
8          }
9      }
10
11     @FXML
12     void showFilteredMedia(FilteredList<Media> filteredData, String filter) {
13         filteredData.setPredicate(media -> {
14             if (filter == null || filter.isEmpty()) {
15                 return true;
16             }
17             String lowerCaseFilter = filter.toLowerCase();
18             if (filterCategory.getSelectedToggle() == radioBtnFilterTitle) {
19                 return media.getTitle().toLowerCase().contains(lowerCaseFilter);
20             } else if (filterCategory.getSelectedToggle() == radioBtnFilterId) {
21                 try {
22                     return Integer.toString(media.getId()).contains(lowerCaseFilter);
23                 } catch (NumberFormatException e) {
24                     return false;
25                 }
26             }
27             return false;
28         });
29     }
30
31     @FXML
32     void createPopUp() {
33         Stage popupwindow =new Stage();
34         popupwindow.initModality(Modality.APPLICATION_MODAL);
35         popupwindow.setTitle("Place order");
36
37         Label label1 = new Label("You have placed your order!");
38         label1.setFont(Font.font("Arial", FontWeight.BOLD,14));
39         Label label2 = new Label("Your bill total is " + Float.toString(cart.totalCost()) + "$");
40         Button button1= new Button("OK!");
41         label2.setTextFill(Color.RED);
42         button1.setOnAction(e -> popupwindow.close());
43         VBox layout= new VBox(10);
44         layout.getChildren().addAll(label1, label2,button1);
45         layout.setAlignment(Pos.CENTER);
46         Scene scene1= new Scene(layout, 300, 200);
47         popupwindow.setScene(scene1);
48         popupwindow.show();
49     }
50
51     @FXML
52     void btnRemovePressed(ActionEvent event) {
53         Media media = tblMedia.getSelectionModel().getSelectedItem();
54         cart.removeMedia(media);
55         totalPrice.setText(Float.toString(cart.totalCost()) + "$");
56     }
57 }
```

5.4 Demo

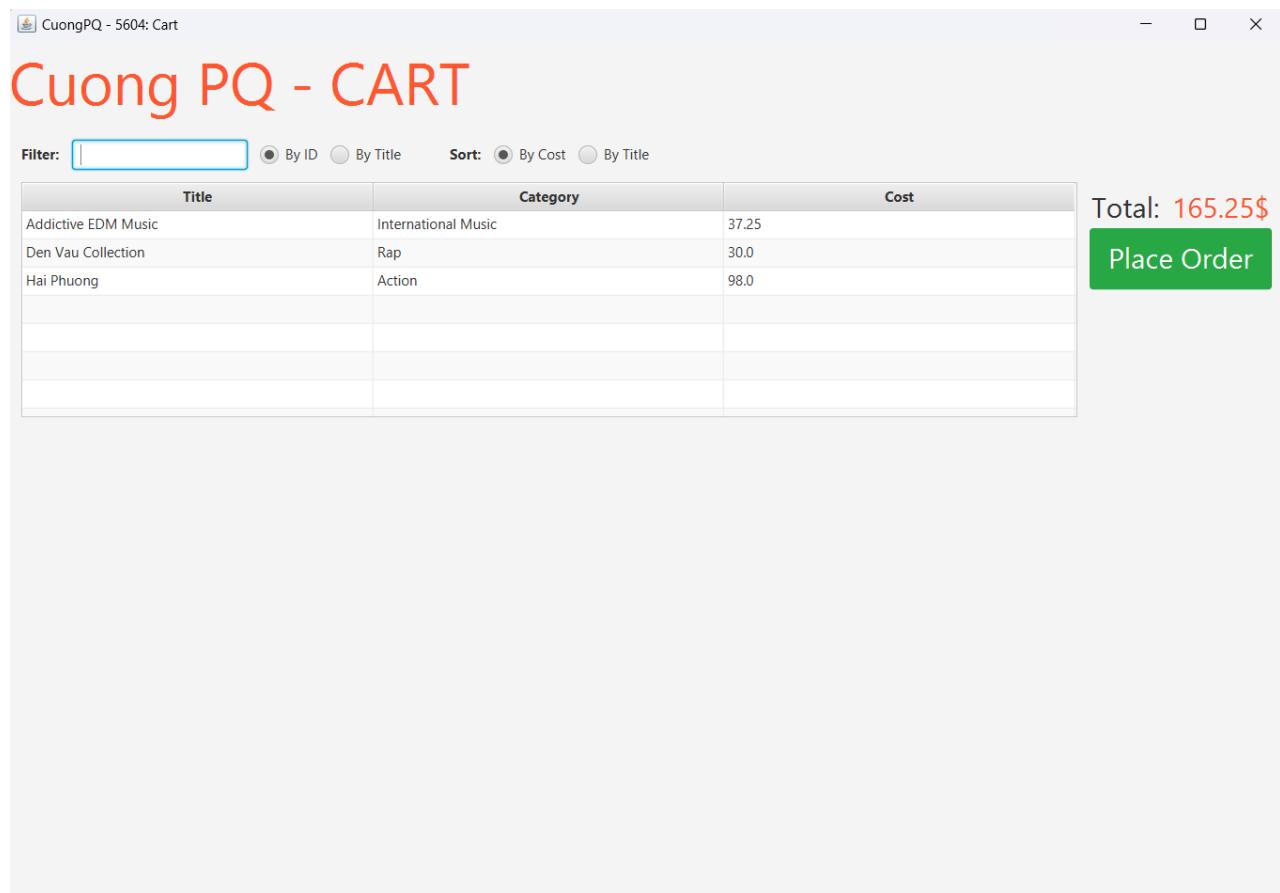


Figure 5.7: Demo CartScreen

6 Updating buttons based on selected item in TableView – ChangeListener

6.1 Edit class CartScreenController

```
1  @FXML
2  private void initialize() {
3      colMediaTitle.setCellValueFactory(new PropertyValueFactory<>("title"));
4      colMediaCategory.setCellValueFactory(new PropertyValueFactory<>("category"));
5      colMediaCost.setCellValueFactory(new PropertyValueFactory<>("cost"));
6      tblMedia.setItems(FXCollections.observableList(this.cart.getItemsOrdered()));
7      tblMedia.setPlaceholder(new Label("No item in cart"));
8      btnPlay.setVisible(false);
9      btnRemove.setVisible(false);
10
11     btnRemove.setOnAction(event -> {
12         try {
13             Media media = tblMedia.getSelectionModel().getSelectedItem();
14             cart.removeMedia(media);
15             totalPrice.setText(cart.totalCost() + "$");
16             tblMedia.setItems(FXCollections.observableList(cart.getItemsOrdered()));
17         } catch (Exception ex) {
18             JOptionPane.showMessageDialog(null, "Cuong PQ 5604: Error: No media in cart to remove", "Error", JOptionPane.ERROR_MESSAGE);
19         }
20     });
21
22     tfFilter.textProperty().addListener((observableValue, s, t1) -> showFilterMedia(t1));
23
24     tblMedia.getSelectionModel().selectedItemProperty().addListener(
25         (observableValue, media, t1) -> updateButtonBar(t1)
26     );
27     totalPrice.setText(cart.totalCost() + "$");
28
29     btnPlay.setOnAction(event -> {
30         try {
31             JDialog playDialog = MediaStore.createPlayDialog(tblMedia.getSelectionModel().getSelectedItem());
32             playDialog.setVisible(true);
33             playDialog.setSize(300, 200);
34             playDialog.pack();
35         } catch (Exception ex) {
36             JOptionPane.showMessageDialog(null, "Cuong PQ 5604: Error: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
37         }
38     });
39
40     btnPlaceOrder.setOnAction(event -> {
41         createPopUp();
42         cart.getItemsOrdered().clear();
43         tblMedia.setItems(FXCollections.observableList(cart.getItemsOrdered()));
44         totalPrice.setText(cart.totalCost() + "$");
45     });
46
47     sortCategory.selectedToggleProperty().addListener((observable, oldValue, newValue) -> {
48         if (newValue == radioBtnSortCost) {
49             cart.getItemsOrdered().sort(Media.COMPARE_BY_COST_TITLE);
50         } else if (newValue == radioBtnSortTitle) {
51             cart.getItemsOrdered().sort(Media.COMPARE_BY_TITLE_COST);
52         }
53         tblMedia.setItems(FXCollections.observableList(cart.getItemsOrdered()));
54     });
55 }
```

Figure 6.1: CartScreenController I

6.2 Demo

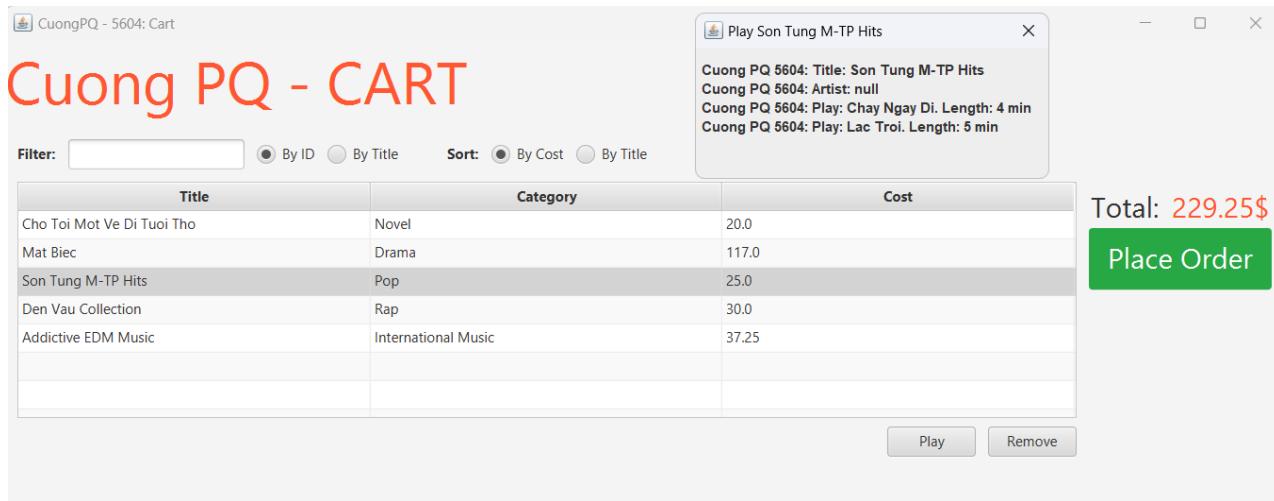


Figure 6.3: Demo media playable

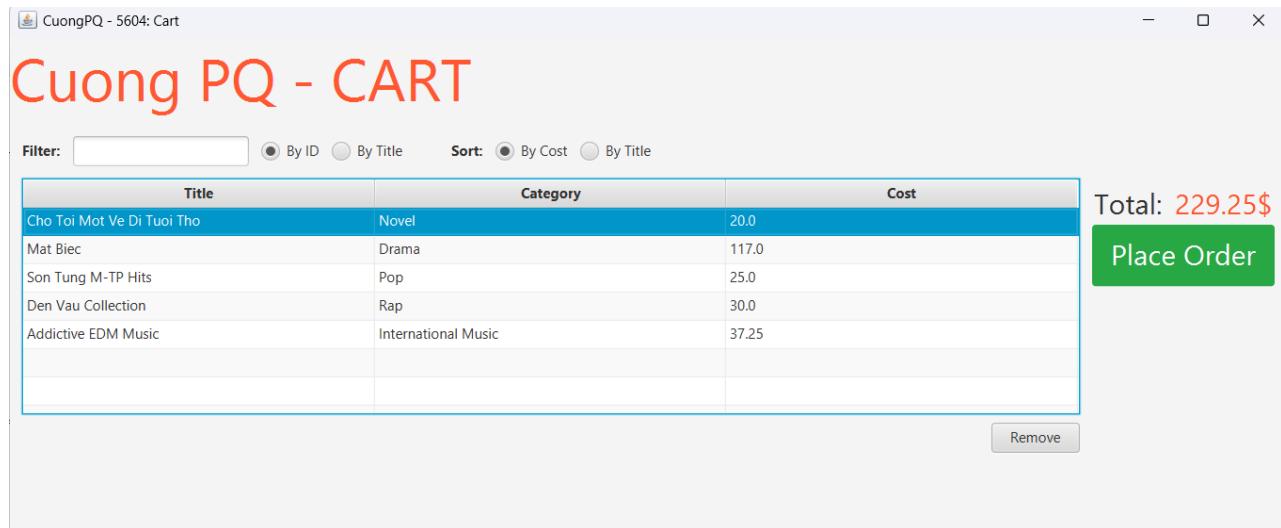


Figure 6.4: Demo media unplayable

7 Deleting a media

7.1 Code

```
1 btnRemove.setOnAction(new EventHandler<javafx.event.ActionEvent>() {
2     @Override
3     public void handle(javafx.event.ActionEvent event) {
4         Media media = tblMedia.getSelectionModel().getSelectedItem();
5         cart.removeMedia(media);
6         totalPrice.setText(Float.toString(cart.totalCost()) + "$");
7         tblMedia.setItems(FXCollections.observableList(cart.getItemsOrdered()));
8     }
9 });
```

Figure 7.1: btnRemovePressed Method

7.2 Demo

Cuong PQ - CART		
Filter:	By ID	By Title
Sort: <input checked="" type="radio"/> By Cost <input type="radio"/> By Title		
Cho Toi Mot Ve Di Tuoi Tho	Novel	20.0
Mat Biec	Drama	117.0
Son Tung M-TP Hits	Pop	25.0
Den Vau Collection	Rap	30.0
Addictive EDM Music	International Music	37.25

Total: 229.25\$

Place Order

Play Remove

Figure 7.2: button Remove

Cuong PQ - CART		
Filter:	By ID	By Title
Sort: <input checked="" type="radio"/> By Cost <input type="radio"/> By Title		
Cho Toi Mot Ve Di Tuoi Tho	Novel	20.0
Mat Biec	Drama	117.0
Son Tung M-TP Hits	Pop	25.0
Den Vau Collection	Rap	30.0

Total: 192.0\$

Place Order

Remove

Figure 7.3: button Remove

8 Complete the Aims GUI application

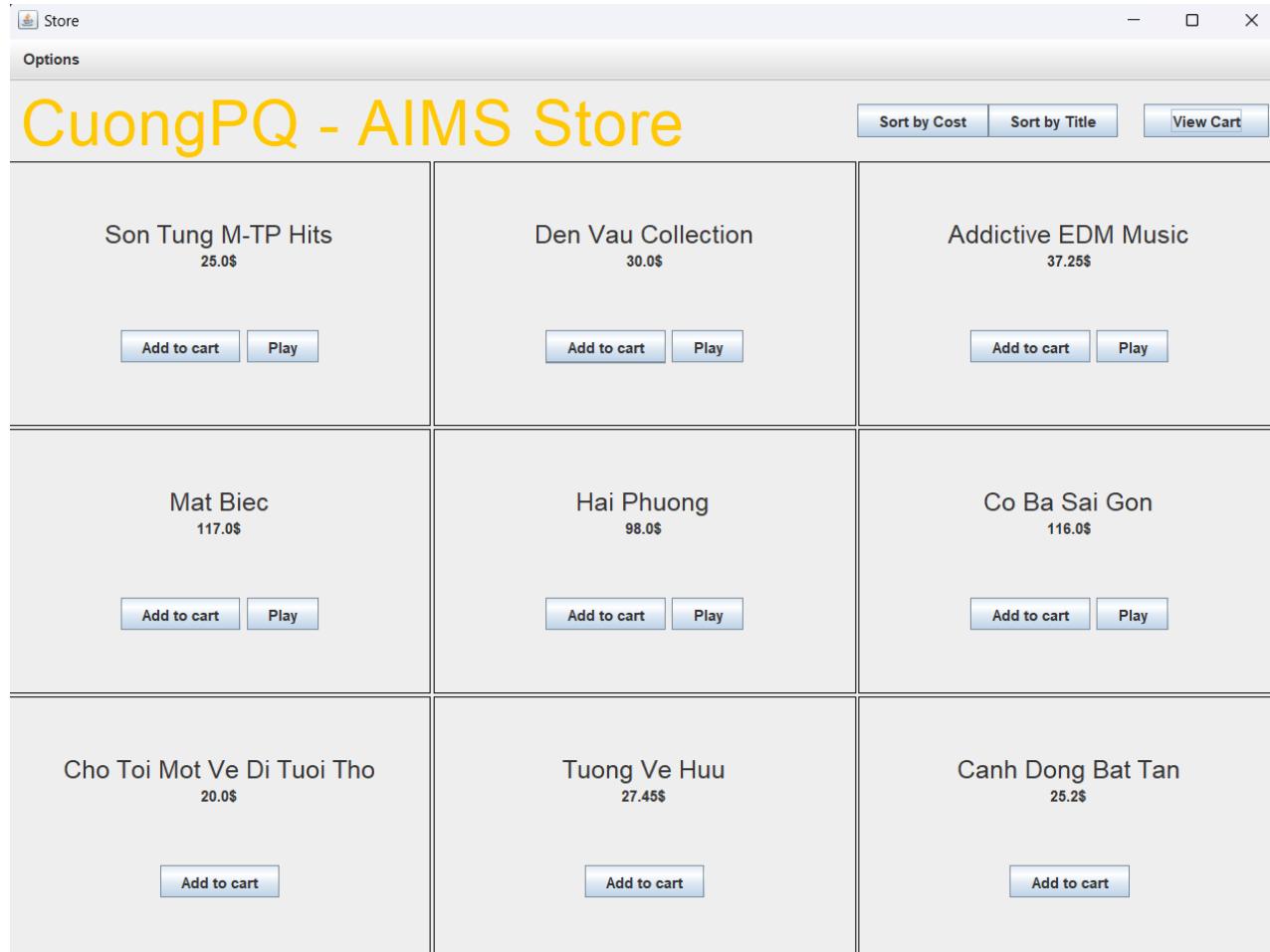


Figure 8.1: Store before add book

The screenshot shows a dialog box titled "Add Book" with fields for entering title, category, cost, and authors, and an "Add" button.

Enter title:	Giai tích II
Enter category:	Sách giáo khoa
Enter cost:	10
Enter authors:	Bùi Xuân Dieu

Add

Figure 8.2: Add book

Son Tung M-TP Hits		Den Vau Collection	Addictive EDM Music	Mat Biec
25.0\$		30.0\$	37.25\$	117.0\$
<input type="button" value="Add to cart"/>	<input type="button" value="Play"/>	<input type="button" value="Add to cart"/>	<input type="button" value="Play"/>	<input type="button" value="Add to cart"/>
Hai Phuong	98.0\$	Co Ba Sai Gon	116.0\$	Cho Toi Mot Ve Di Tuoi Tho
<input type="button" value="Add to cart"/>	<input type="button" value="Play"/>	<input type="button" value="Add to cart"/>	<input type="button" value="P"/>	<input type="button" value="Add to cart"/>
Canh Dong Bat Tan	25.2\$	Giai tich II	10.0\$	Tuong Ve Huu
<input type="button" value="Add to cart"/>		<input type="button" value="Add to cart"/>		<input type="button" value="27.45\$"/>

Figure 8.3: Store after add book

Enter title:	Thien ly oi
Enter category:	Pop
Enter cost:	25
Enter artist:	J97
Enter length:	222
Number of tracks:	1
<input type="button" value="Add"/>	

Figure 8.4: Add CD

Son Tung M-TP Hits 25.0\$	Den Vau Collection 30.0\$	Addictive EDM Music 37.25\$	Mat Biec 117.0\$
Add to cart Play	Add to cart P Add to cart	Add to cart Play	Add to cart cart Play
Hai Phuong 98.0\$	Co Ba Sai Gon 116.0\$	Cho Toi Mot Ve Di Tuoi Tho 20.0\$	Tuong Ve Huu 27.45\$
Add to cart Play	Add to cart Play	Add to cart	Add to cart Add to cart
Canh Dong Bat Tan 25.2\$	Giai tích II 10.0\$	Thien ly oi 25.0\$	
Add to cart	Add to cart	Add to cart Play	

Figure 8.5: Store after add CD

Add DVD

Enter title: Dune

Enter category: SciFi

Enter cost: 30

Enter length: 222

Add

Figure 8.6 Add DVD

CuongPQ - AIMS Store				
		Sort by Cost	Sort by Title	View Cart
Son Tung M-TP Hits 25.0\$	Add to cart Play	Den Vau Collection 30.0\$	Add to cart Play	Addictive EDM Music 37.25\$
				Mat Biec 117.0\$
Hai Phuong 98.0\$	Add to cart Play	Co Ba Sai Gon 116.0\$	Add to cart Play	Cho Toi Mot Ve Di Tuoi Tho 20.0\$
				Tuong Ve Huu 27.45\$
Canh Dong Bat Tan 25.2\$	Add to cart	Giai tich II 10.0\$	Add to cart	Thien ly oi 25.0\$
				Dune 30.0\$

Figure 8.7: Store after add DVD

Title	Category	Cost
Cho Toi Mot Ve Di Tuoi Tho	Novel	20.0
Mat Biec	Drama	117.0
Son Tung M-TP Hits	Pop	25.0
Den Vau Collection	Rap	30.0
Giai tich II	Sach giao khoa	10.0
Thien ly oi	Pop	25.0
Dune	SciFi	30.0

Total: 257.0\$

Place Order

Figure 8.8: Cart

```
@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(message: "ERROR: DVD length is non-positive");
    }
}

// Pham Quoc Cuong - 20225604
@Override
public void play() throws PlayerException [
    if(this.getLength() > 0) {
        System.out.println("Playing CD: " + this.getTitle());
        System.out.println("CD length: " + this.getLength());
        System.out.println("CD artist: " + this.getArtist());
        System.out.println(x:"CD tracks: ");
        for (Track track : tracks) {
            track.play();
        }
    } else {
        throw new PlayerException(message: "ERROR: CD length is non-positive");
    }
]

💡 You, 3 days ago • simple
💡 // Pham Quoc Cuong - 20225604      You, 3 days ago • simple
@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(message: "ERROR: Track length is non-positive");
    }
}
```

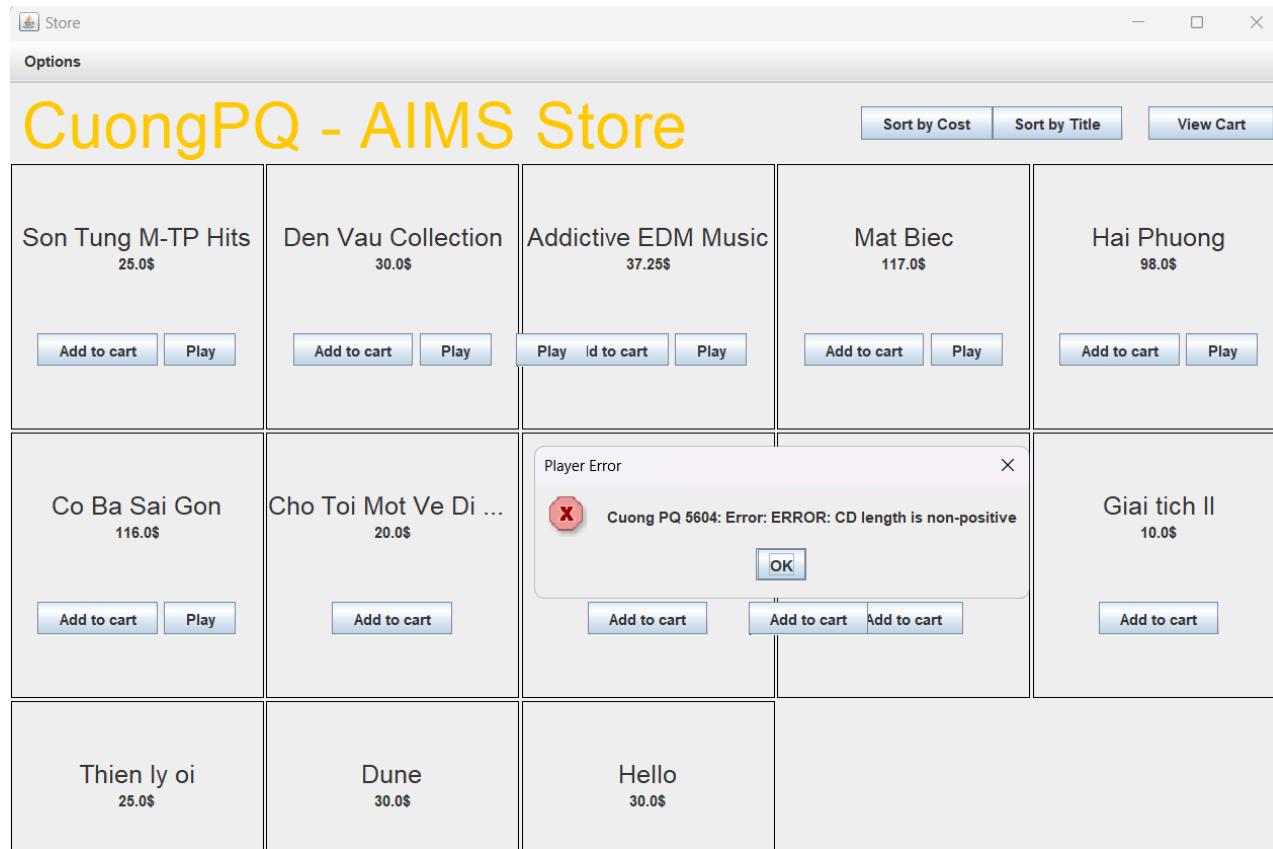
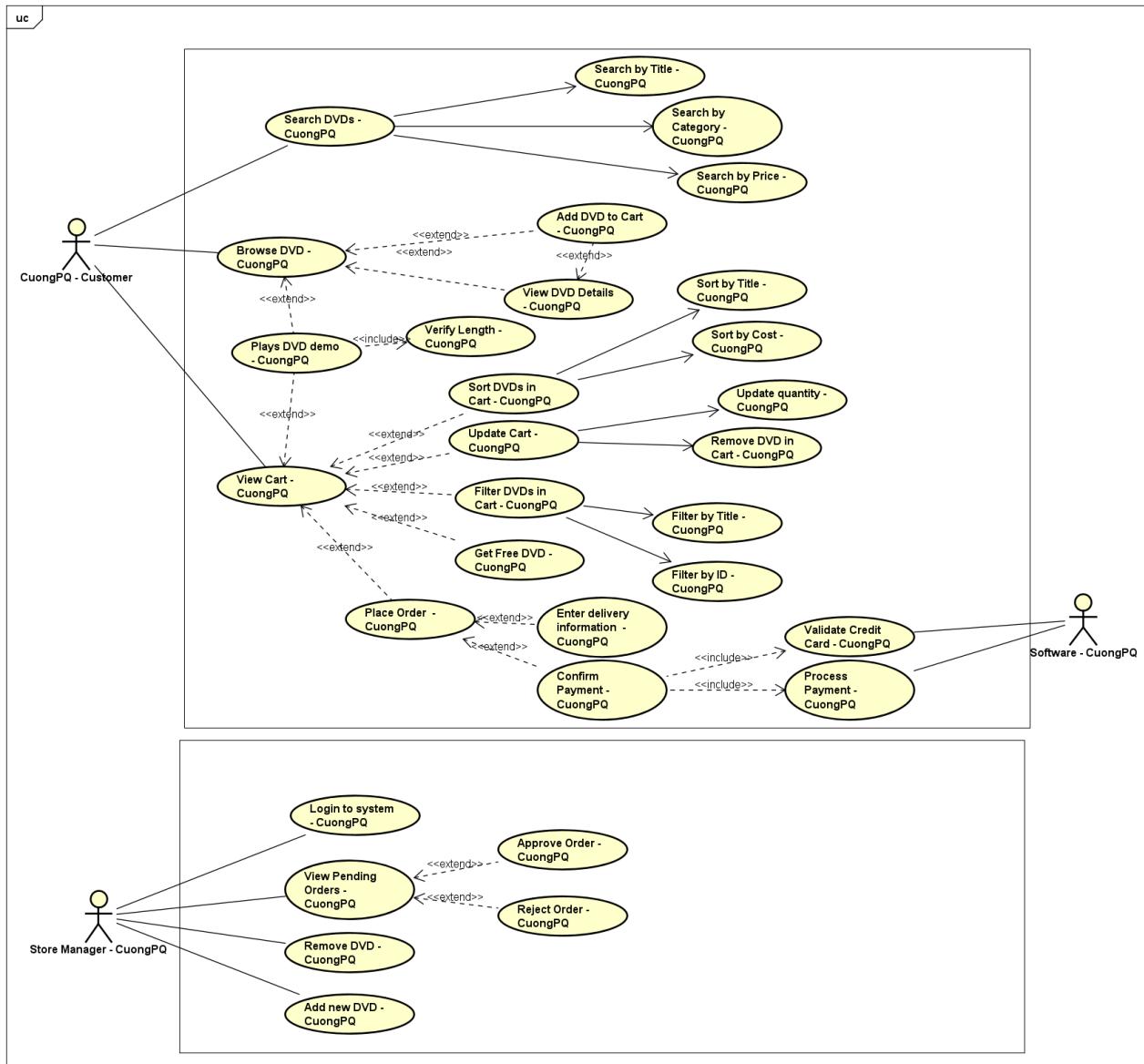
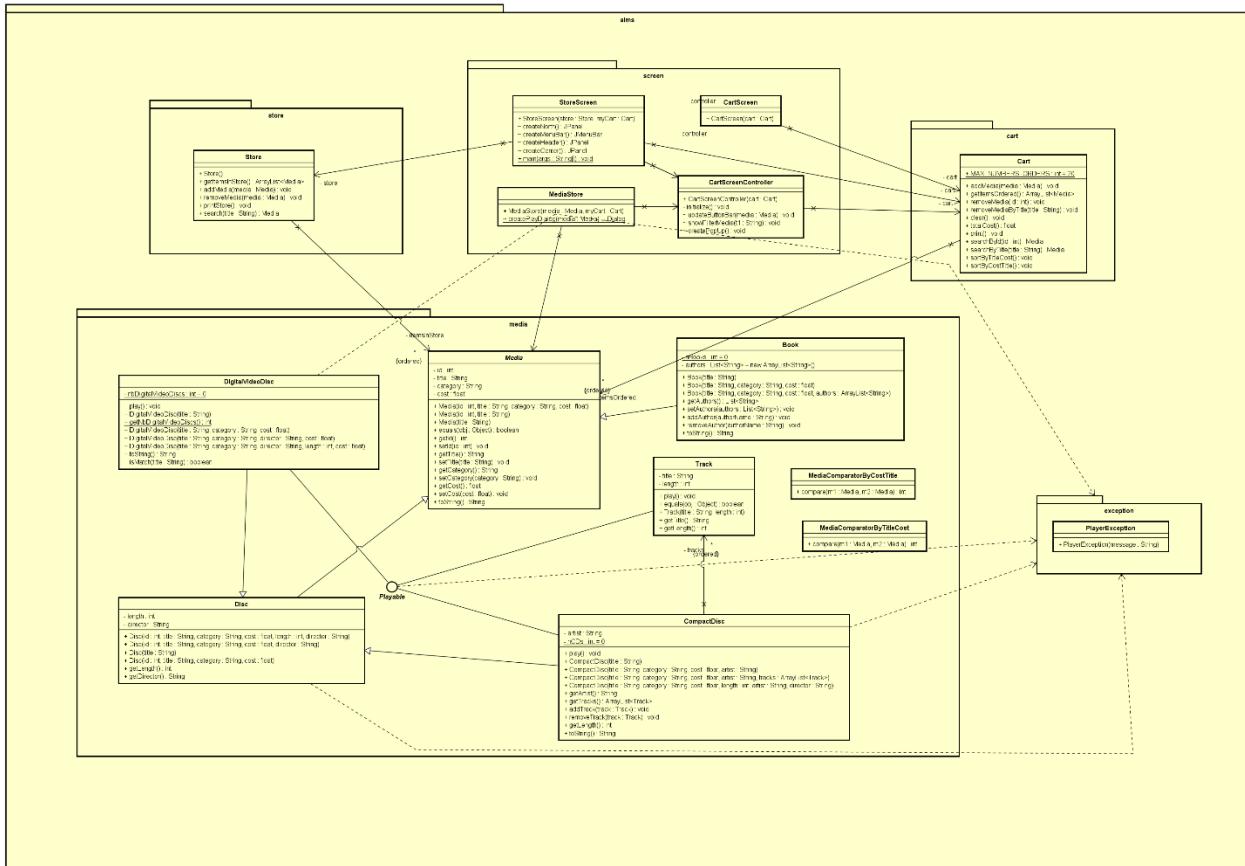


Figure 8.9: Exception

9 Use case Diagram



10 Class Diagram



11 Filter items in cart

Filter by ID:

Cuong PQ - CART

Filter: By ID By Title By Cost By Title

Title	Category	Cost
Mat Biec	Drama	117.0

Total: 411.45\$

Place Order

Filter by Title:

The screenshot shows a window titled "Cuong PQ - CART". At the top, there is a search bar with "Filter: edm" and sorting options: "By ID", "By Title" (selected), "Sort: By Cost" (selected), and "By Title". Below this is a table with columns "Title", "Category", and "Cost". One item is listed: "Addictive EDM Music" under "International Music" at a cost of "37.25". In the bottom right corner, it says "Total: 411.45\$" and there is a green "Place Order" button.

12 Sort items in store

Sort by Cost:

The screenshot shows a window titled "CuongPQ - AIMS Store". At the top, there are buttons for "Sort by Cost", "Sort by Title", and "View Cart". The main area displays nine music tracks in a 3x3 grid. The tracks are sorted by cost:

- Row 1: Cho Toi Mot Ve Di Tuoi Tho (20.0\$), Canh Dong Bat Tan (25.2\$), Son Tung M-TP Hits (25.0\$)
- Row 2: Tuong Ve Huu (27.45\$), Den Vau Collection (30.0\$), Addictive EDM Music (37.25\$)
- Row 3: Hai Phuong (98.0\$), Co Ba Sai Gon (116.0\$), Mat Biec (117.0\$)

Each track card includes an "Add to cart" button and a "Play" button.

Sort by Title:

Addictive EDM Music 37.25\$ <input type="button" value="Add to cart"/> <input type="button" value="Play"/>	Canh Dong Bat Tan 25.2\$ <input type="button" value="Add to cart"/>	Cho Toi Mot Ve Di Tuoi Tho 20.0\$ <input type="button" value="Add to cart"/>
Co Ba Sai Gon 116.0\$ <input type="button" value="Add to cart"/> <input type="button" value="Play"/>	Den Vau Collection 30.0\$ <input type="button" value="Add to cart"/> <input type="button" value="Play"/>	Hai Phuong 98.0\$ <input type="button" value="Add to cart"/> <input type="button" value="Play"/>
Mat Biec 117.0\$ <input type="button" value="Add to cart"/> <input type="button" value="Play"/>	Son Tung M-TP Hits 25.0\$ <input type="button" value="Add to cart"/> <input type="button" value="Play"/>	Tuong Ve Huu 27.45\$ <input type="button" value="Add to cart"/>

13 Sort items in cart

Sort by Cost:

Title	Category	Cost
Canh Dong Bat Tan	Novel	25.2
Addictive EDM Music	International Music	37.25
Co Ba Sai Gon	Drama	116.0
Co Ba Sai Gon	Drama	116.0
Mat Biec	Drama	117.0

Total: 411.45\$

Sort by Title:

The screenshot shows a web-based shopping cart interface. At the top, there is a header with a logo and the text "CuongPQ - 5604: Cart". Below the header, the title "Cuong PQ - CART" is displayed in large red letters. There are three sorting options: "By ID", "By Title" (which is selected), and "By Cost". A "Filter:" input field is present. To the right of the table, the total cost "Total: 411.45\$" is shown in red, and a green button labeled "Place Order" is visible. The table lists the following items:

Title	Category	Cost
Addictive EDM Music	International Music	37.25
Canh Dong Bat Tan	Novel	25.2
Co Ba Sai Gon	Drama	116.0
Co Ba Sai Gon	Drama	116.0
Mat Biec	Drama	117.0

14 Answer Questions

The Aims class must be updated to handle any exceptions generated when the play() methods are called. What happens when you don't update for them to catch?

Nếu lớp Aims không được điều chỉnh để xử lý ngoại lệ khi gọi phương thức play(), chương trình có thể gặp sự cố và dừng đột ngột mà không có cách khắc phục cụ thể. Điều này khiến người dùng khó xác định nguyên nhân gây ra lỗi, dẫn đến trải nghiệm sử dụng không thuận lợi.

Bằng cách sử dụng khối try-catch, ngoại lệ có thể được xử lý một cách chủ động. Khi lỗi xảy ra, khối catch sẽ nắm bắt và cung cấp thông tin chi tiết thông qua các phương thức như getMessage(), toString(), hoặc printStackTrace(). Điều này không chỉ giúp người dùng hiểu rõ vấn đề mà còn cho phép họ tiếp tục sử dụng ứng dụng mà không bị gián đoạn.