

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

BÁO CÁO THỰC HÀNH
IT3103-744528-2024.1
BÀI THỰC HÀNH 5

Họ và tên sv: Đỗ Hoàng Đông
MSSV: 20225807
Lớp: Việt Nhật 03 – K67
GVHD: Lê Thị Hoa
HTGD: Đặng 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

```
//Đo Hoàng Dong 20225807
package hust.soict.hedspi.aims.swing;

import java.awt.Frame;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class AWTAccumulator extends Frame {
    private TextField tfInput;    // Ô nhập dữ liệu
    private TextField tfOutput;  // Ô hiển thị kết quả
    private int sum = 0;         // Tổng tích lũy ban đầu là 0

    // Constructor để thiết lập giao diện và các sự kiện
    public AWTAccumulator() {
        setLayout(new GridLayout(2, 2)); // Layout dạng lưới 2x2

        // Nhấn và ô nhập dữ liệu
        add(new Label(text: "Enter an Integer: "));
        tfInput = new TextField(columns: 10);
        add(comp: tfInput);
        tfInput.addActionListener(new TFInputListener());

        // Nhấn và ô hiển thị kết quả
```

Figure 1.1: Source code of AWTAccumulator

```
        // Nhấn và ô hiển thị kết quả
        add(new Label(text: "The Accumulated Sum is: "));
        tfOutput = new TextField(columns: 10);
        tfOutput.setEditable(b: false); // Không cho chỉnh sửa kết quả
        add(comp: tfOutput);

        // Cài đặt cửa sổ
        setTitle(title: "AWT Accumulator");
        setSize(width: 350, height: 120);
        setVisible(b: true);
    }

    // Hàm main để chạy chương trình
    public static void main(String[] args) {
        new AWTAccumulator();
    }

    // Lớp xử lý sự kiện khi nhấn Enter trong ô nhập
    private class TFInputListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent evt) {
            int numberIn = Integer.parseInt(s: tfInput.getText()); // Lấy số nguyên
            sum += numberIn; // Cộng dồn vào tổng
            tfInput.setText(t: ""); // Xóa ô nhập
            tfOutput.setText(sum + ""); // Hiển thị tổng tích lũy
        }
    }
}
```

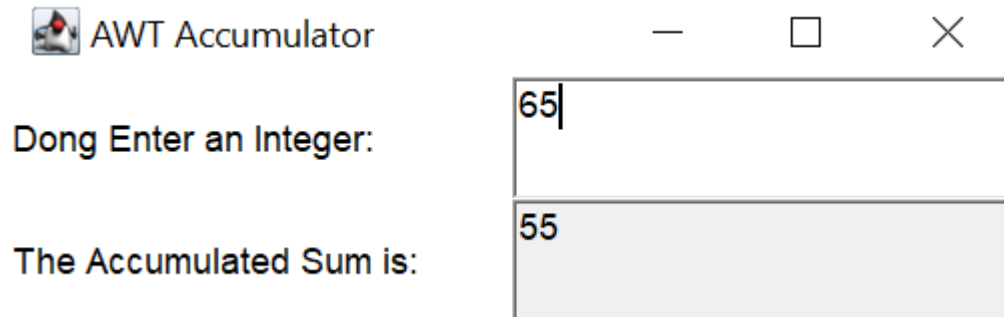


Figure 1.2: Demo of AWTAccumulator

1.2 SwingAccumulator

```

1 package hust.soict.hedspi.aims.swing;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6
7 public class SwingAccumulator extends JFrame {
8     private JTextField tfInput;    // ô nhập số nguyên
9     private JTextField tfOutput;   // ô hiển thị kết quả
10    private int sum = 0;           // Tổng tích lũy, khởi tạo bằng 0
11
12    // Constructor để thiết lập giao diện và các xử lý sự kiện
13    public SwingAccumulator() {
14        Container cp = getContentPane();
15        cp.setLayout(new GridLayout(2, 2)); // Layout dạng lưới 2x2
16
17        // Nhãn và ô nhập dữ liệu
18        cp.add(new JLabel(text: "Enter an Integer: "));
19        tfInput = new JTextField(columns: 10);
20        cp.add(comp: tfInput);
21        tfInput.addActionListener(new TFInputListener());
22
23        // Nhãn và ô hiển thị kết quả
24        cp.add(new JLabel(text: "The Accumulated Sum is: "));
25        tfOutput = new JTextField(columns: 10);
26        tfOutput.setEditable(b: false); // Không cho phép chỉnh sửa ô kết quả
27        cp.add(comp: tfOutput);
28    }
29 }

```

Figure 1.3: Source code of SwingAccumulator

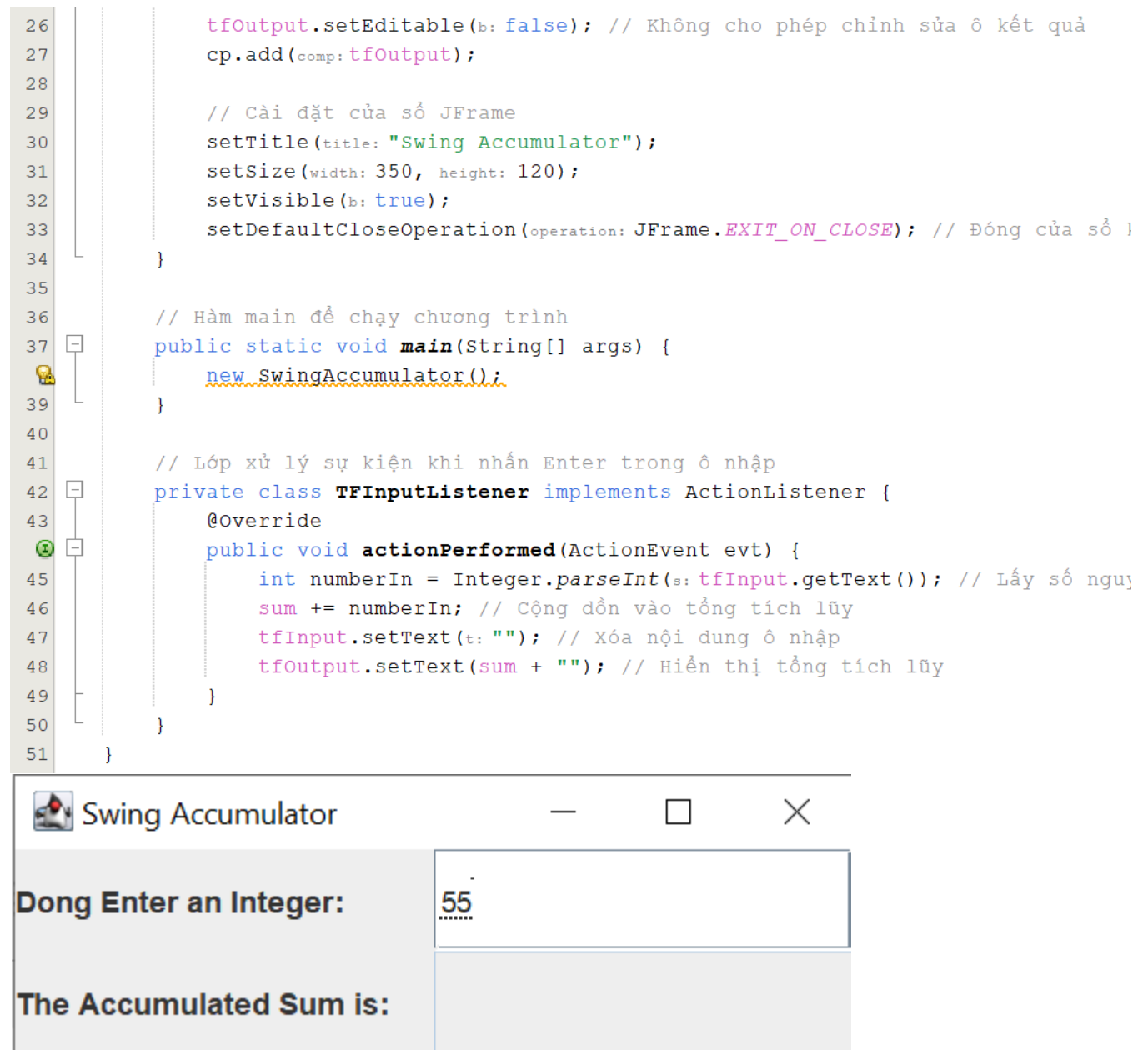


Figure 1.4: Demo of SwingAccumulator

2 Organizing Swing components with Layout Managers

2.1 Code

```

1 //Do Hoang Dong 20225807
2 package hust.soict.hedspi.aims.swing;
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6
7 public class NumberGrid extends JFrame {
8     private JButton[] btnNumbers = new JButton[10];
9     private JButton btnDelete, btnReset;
10    private JTextField tfDisplay;
11
12    public NumberGrid() {
13
14        tfDisplay = new JTextField();
15        tfDisplay.setPreferredSize(new Dimension(width: 200, height: 30));
16        tfDisplay.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
17
18        JPanel panelButtons = new JPanel(new GridLayout(4, 3));
19        addButtons(panelButtons);
20
21        Container cp = getContentPane();
22        cp.setLayout(new BorderLayout());
23        cp.add(comp: tfDisplay, constraints: BorderLayout.NORTH);
24        cp.add(comp: panelButtons, constraints: BorderLayout.CENTER);
25
26        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27        setTitle(title: "Mai Minh Quân - 20225661 - Number Grid");
28        setSize(width: 200, height: 200);
29        setVisible(b: true);
30    }
31
32    public static void main(String[] args) {
33        new NumberGrid();
34    }
35
36    void addButtons(JPanel panelButtons) {
37        ButtonListener btnListener = new ButtonListener();
38        for (int i = 1; i <= 9; i++) {
39            btnNumbers[i] = new JButton(""+i);
40            panelButtons.add(btnNumbers[i]);
41            btnNumbers[i].addActionListener(l: btnListener);
42        }
43        btnDelete = new JButton(text: "DEL");
44        panelButtons.add(comp: btnDelete);
45        btnDelete.addActionListener(l: btnListener);
46
47        btnNumbers[0] = new JButton(text: "0");
48        panelButtons.add(btnNumbers[0]);
49        btnNumbers[0].addActionListener(l: btnListener);
50
51        btnReset = new JButton(text: "C");

```

Figure 2.1: Source code of NumberGrid 1

```

26    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27    setTitle(title: "Mai Minh Quân - 20225661 - Number Grid");
28    setSize(width: 200, height: 200);
29    setVisible(b: true);
30    }
31
32    public static void main(String[] args) {
33        new NumberGrid();
34    }
35
36    void addButtons(JPanel panelButtons) {
37        ButtonListener btnListener = new ButtonListener();
38        for (int i = 1; i <= 9; i++) {
39            btnNumbers[i] = new JButton(""+i);
40            panelButtons.add(btnNumbers[i]);
41            btnNumbers[i].addActionListener(l: btnListener);
42        }
43        btnDelete = new JButton(text: "DEL");
44        panelButtons.add(comp: btnDelete);
45        btnDelete.addActionListener(l: btnListener);
46
47        btnNumbers[0] = new JButton(text: "0");
48        panelButtons.add(btnNumbers[0]);
49        btnNumbers[0].addActionListener(l: btnListener);
50
51        btnReset = new JButton(text: "C");

```



```

51         btnReset = new JButton(text: "C");
52         panelButtons.add(comp: btnReset);
53         btnReset.addActionListener(l: btnListener);
54     }
55
56     private class ButtonListener implements ActionListener {
57
58         @Override
59         public void actionPerformed(ActionEvent e) {
60             String button = e.getActionCommand();
61             if (button.charAt(index: 0) >= '0' && button.charAt(index: 0) <= '9') {
62                 tfDisplay.setText(tfDisplay.getText() + button);
63             } else if (button.equals(anObject: "DEL")) {
64                 String deleteString = tfDisplay.getText();
65                 if (deleteString.length() > 0) {
66                     deleteString = deleteString.substring(beginIndex: 0, deleteString.length() - 1);
67                 }
68                 tfDisplay.setText(t: deleteString);
69             } else {
70                 tfDisplay.setText(t: "");
71             }
72         }
73     }
74
75     public static String delLastCharacter(String str) {
76         if ((str != null) && (str.length() > 0)) {

```

Figure 2.2: Source code of NumberGrid 2

2.2 Demo

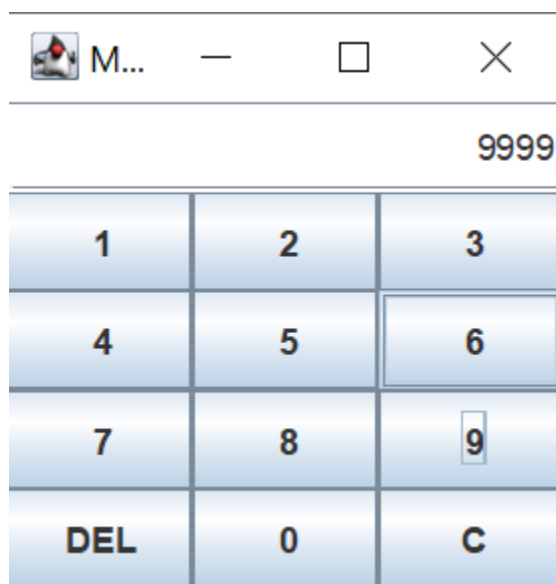


Figure 2.3: Demo buttons 0-9

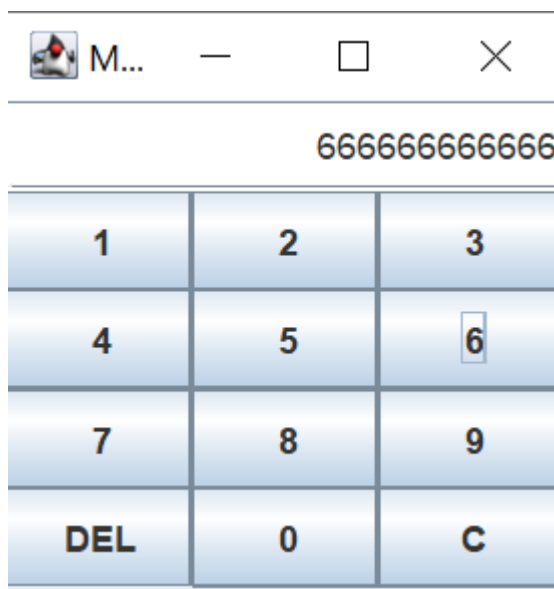


Figure 2.4: Demo DEL button

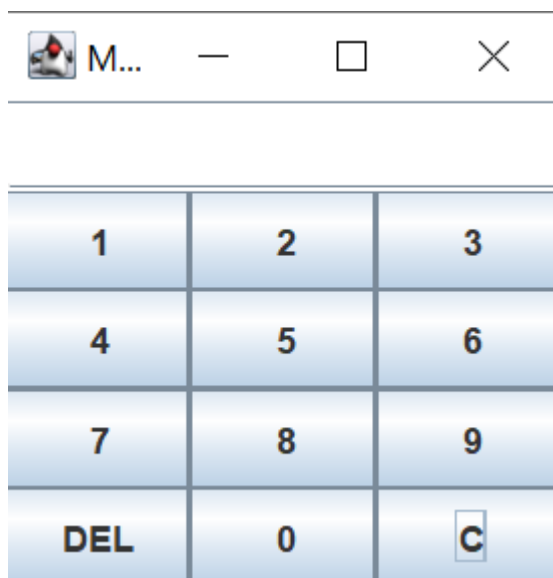


Figure 2.5: Demo C button

3 Create a graphical user interface for AIMS with Swing

3.1 Create class StoreScreen

```

1 //Đo Hoàng Dong 20225807
2 package hust.soict.hedspi.aims.Screen;
3 import javax.swing.*;
4 import java.awt.*;
5 import java.util.*;
6 import java.awt.event.*;
7 import src.hust.soict.hedspi.aims.Cart.Cart;
8 import src.hust.soict.hedspi.aims.media.*;
9 import src.hust.soict.hedspi.aims.store.Store;
10
11 public class StoreScreen extends JFrame {
12     private static Store store = new Store();
13     private static Cart cart = new Cart();
14
15
16     public static void main(String[] args) {
17         // initSetup();
18         new StoreScreen(store);
19     }
20
21     public StoreScreen(Store store) {
22         StoreScreen.store = store;
23         Container cp = getContentPane();
24         cp.setLayout(new BorderLayout());
25
26         cp.add(comp: createNorth(), constraints: BorderLayout.NORTH);

```

Figure 3.1: Class StoreScreen 1

```
28 cp.add(comp:createNorth(), constraints:BorderLayout.NORTH);
29 cp.add(comp:createCenter(), constraints:BorderLayout.CENTER);
30
31 setTitle(title: "Do Hoang Dong - 20225807 - Store");
32 setSize(width: 1024, height: 768);
33 setVisible(b: true);
34 setDefaultCloseOperation(operation: JFrame.EXIT_ON_CLOSE);
35
36 }
37
38 JPanel createNorth() {
39     JPanel north = new JPanel();
40     north.setLayout(new BoxLayout(target: north, axis: BoxLayout.Y_AXIS));
41     north.add(comp:createMenuBar());
42     north.add(comp:createHeader());
43     return north;
44 }
45
46 JMenuBar createMenuBar() {
47     JMenu menu = new JMenu(s: "Options");
48
49     JMenu smUpdateStore = new JMenu(s: "Update Store");
50     JMenuItem smAddBook = new JMenuItem(text: "Add Book");
51     JMenuItem smAddCD = new JMenuItem(text: "Add CD");
52     JMenuItem smAddDVD = new JMenuItem(text: "Add DVD");
53
54     // smUpdateStore.add(smAddBook);
55 }
```

```

49 JMenuItem smAddCD = new JMenuItem(text: "Add CD");
50 JMenuItem smAddDVD = new JMenuItem(text: "Add DVD");
51 // smUpdateStore.add(smAddBook);
52 // smUpdateStore.add(smAddCD);
53 // smUpdateStore.add(smAddDVD);
54 //
55 // smAddBook.addActionListener(new btnMenuListener());
56 // smAddCD.addActionListener(new btnMenuListener());
57 // smAddDVD.addActionListener(new btnMenuListener());
58 //
59 menu.add(menuItem: smUpdateStore);
60
61 JMenuItem viewStoreMenu = new JMenuItem(text: "View store");
62 JMenuItem viewCartMenu = new JMenuItem(text: "View cart");
63 // menu.add(viewStoreMenu);
64 // menu.add(viewCartMenu);
65 // viewStoreMenu.addActionListener(new ActionListener() {
66 //     @Override
67 //     public void actionPerformed(ActionEvent e) {
68 //         new StoreScreen(store);
69 //     }
70 // });
71 // viewCartMenu.addActionListener(new ActionListener() {
72 //     @Override
73 //     public void actionPerformed(ActionEvent e) {
74 //         new CartScreen(cart);

```

Figure 3.2: Class StoreScreen 2

```
98 //
99 JPanel createHeader() {
100
101     JPanel header = new JPanel();
102     header.setLayout(new BoxLayout(target: header, axis: BoxLayout.X_AXIS));
103
104     JLabel title = new JLabel(text: "AIMS");
105     title.setFont(new Font(name: title.getFont().getName(), style: Font.PLAIN, size: 50));
106     title.setForeground(fg: Color.CYAN);
107
108     JButton cartBtn = new JButton(text: "View cart");
109     cartBtn.setPreferredSize(new Dimension(width: 100, height: 50));
110     cartBtn.setMaximumSize(new Dimension(width: 100, height: 50));
111     //     cartBtn.addActionListener(new ActionListener() {
112     //         @Override
113     //         public void actionPerformed(ActionEvent e) {
114     //             new CartScreen(cart);
115     //         }
116     //     });
117
118     header.add(comp: Box.createRigidArea(new Dimension(width: 10, height: 10)));
119     header.add(comp: title);
120     header.add(comp: Box.createHorizontalGlue());
121     header.add(comp: cartBtn);
122     header.add(comp: Box.createRigidArea(new Dimension(width: 10, height: 10)));
123 }
```

Figure 3.3: Class StoreScreen 3


```
header.add(comp: Box.createRigidArea(new Dimension(width: 10, height: 10)));
header.add(comp: title);
header.add(comp: Box.createHorizontalGlue());
header.add(comp: cartBtn);
header.add(comp: Box.createRigidArea(new Dimension(width: 10, height: 10)));

return header;
}

//
] JPanel createCenter() {

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

    ArrayList<Media> mediaInStore = store.getItemsInStore();
    for (int i = 0; i < mediaInStore.size(); i++) {
        MediaStore cell = new MediaStore(media: mediaInStore.get(index: i), cart);
        center.add(comp: cell);
    }

    return center;
}
```

Figure 3.4: Class StoreScreen 4

3.2 Create class MediaStore

```

1 package hust.soict.hedspi.aims.Screen;
2
3 import javax.naming.LimitExceededException;
4 import javax.swing.*;
5 import java.awt.*;
6 import java.awt.event.*;
7 import src.hust.soict.hedspi.aims.Cart.Cart;
8 import src.hust.soict.hedspi.aims.media.*;
9
10 public class MediaStore extends JPanel {
11     public MediaStore(Media media, Cart cart) {
12
13         this.setLayout(new BorderLayout(target: this, axis: BorderLayout.Y_AXIS));
14
15         JLabel title = new JLabel(text: media.getTitle());
16         title.setFont(new Font(name: title.getFont().getName(), style: Font.PLAIN, size: 20));
17         title.setAlignmentX(alignmentX: CENTER_ALIGNMENT);
18
19         JLabel cost = new JLabel(""+media.getCost()+"$");
20         cost.setAlignmentX(alignmentX: CENTER_ALIGNMENT);
21
22         JPanel container = new JPanel();
23         container.setLayout(new FlowLayout(align: FlowLayout.CENTER));
24
25         // JButton addToCartButton = new JButton("Add to cart");
26         // addToCartButton.addActionListener(new ActionListener() {

```

Figure 3.7: Class MediaStore 1

```

header.add(comp: Box.createRigidArea(new Dimension(width: 10, height: 10)));
header.add(comp: title);
header.add(comp: Box.createHorizontalGlue());
header.add(comp: cartBtn);
header.add(comp: Box.createRigidArea(new Dimension(width: 10, height: 10)));

return header;
}

//
] JPanel createCenter() {

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

    ArrayList<Media> mediaInStore = store.getItemsInStore();
    for (int i = 0; i < mediaInStore.size(); i++) {
        MediaStore cell = new MediaStore(media: mediaInStore.get(index: i), cart);
        center.add(comp: cell);
    }

    return center;
}

```

Figure 3.8: Class MediaStore 2

3.3 Demo

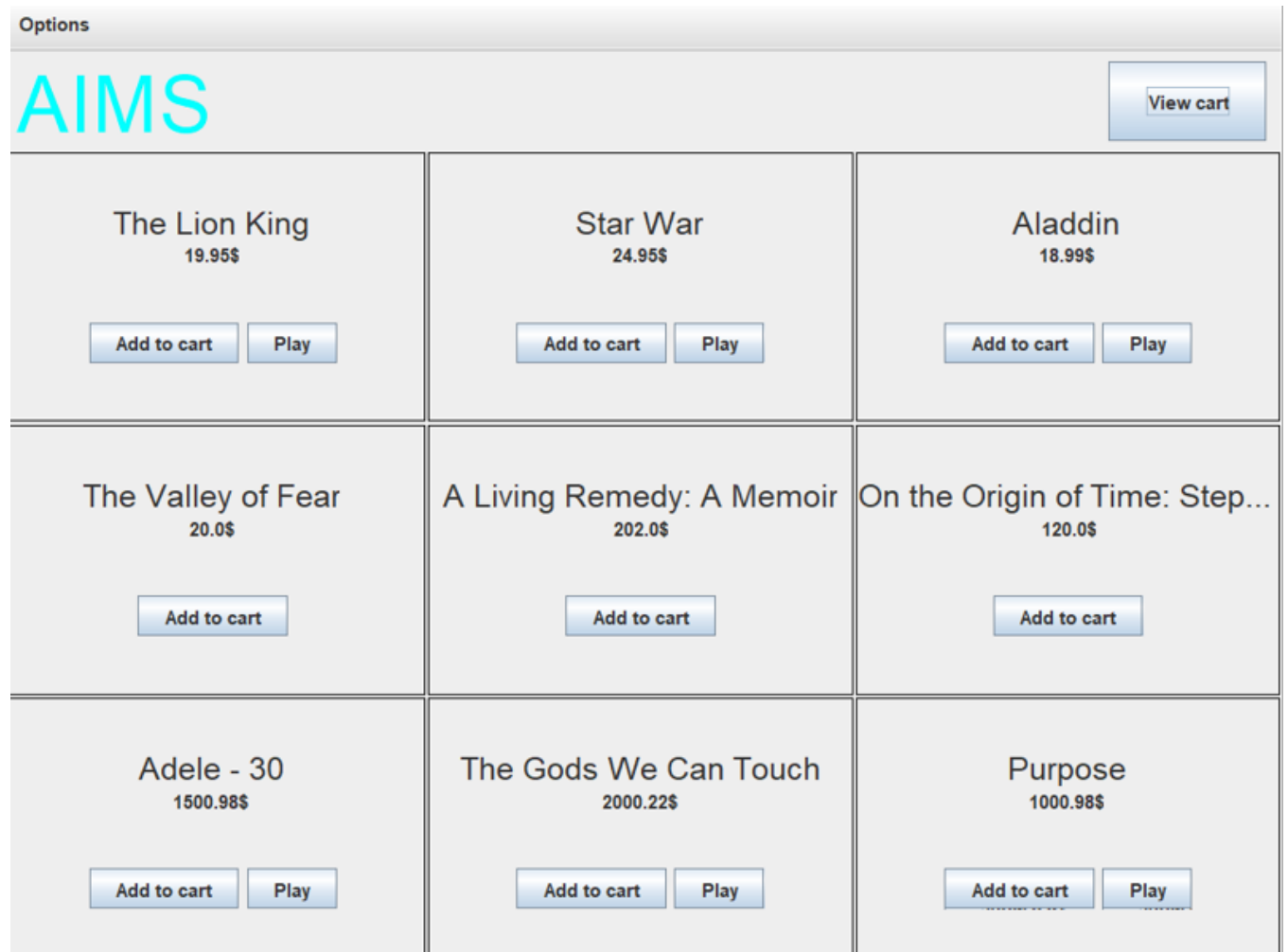


Figure 3.10: StoreScreen

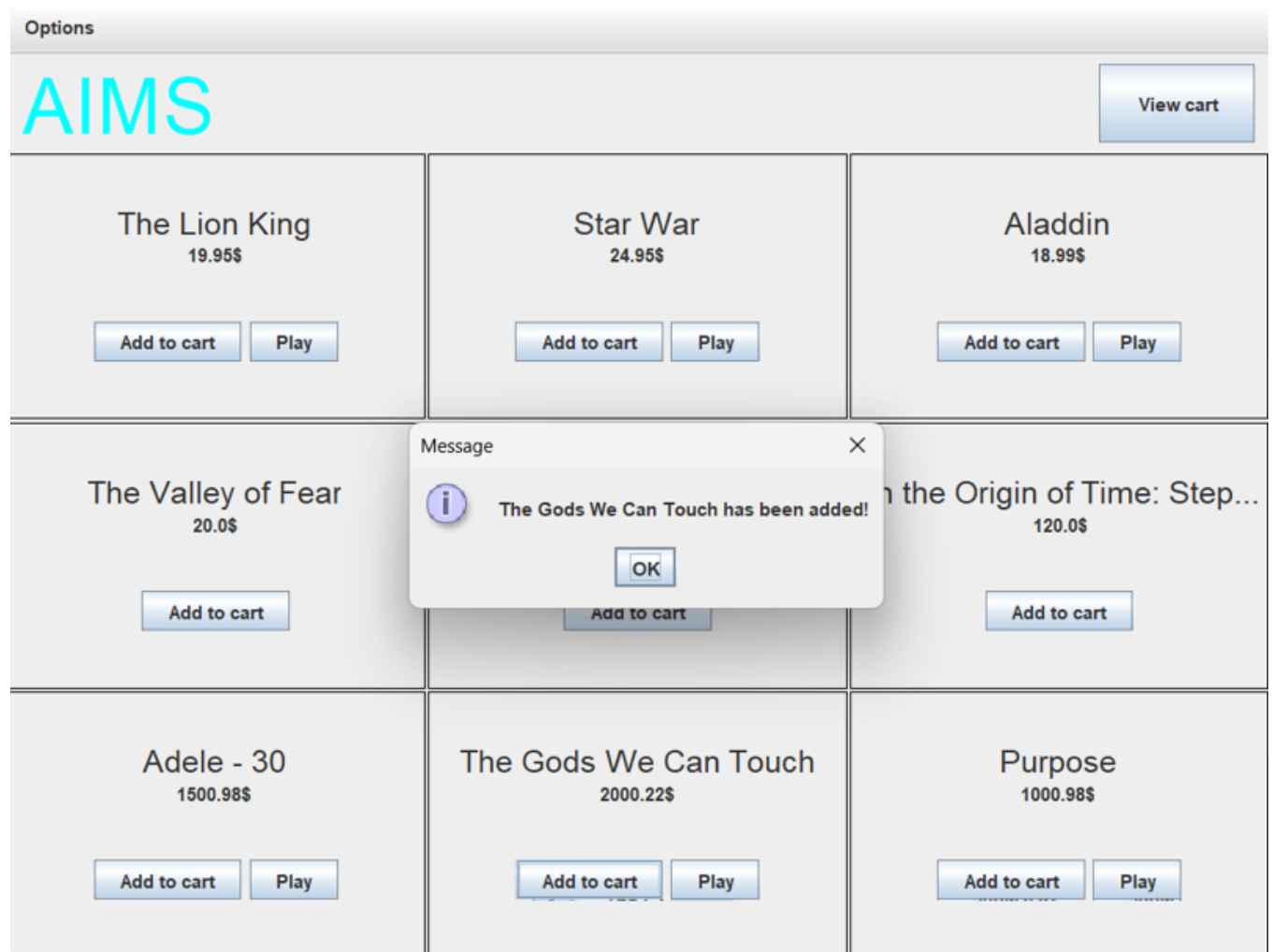


Figure 3.11 Demo Add to cart button

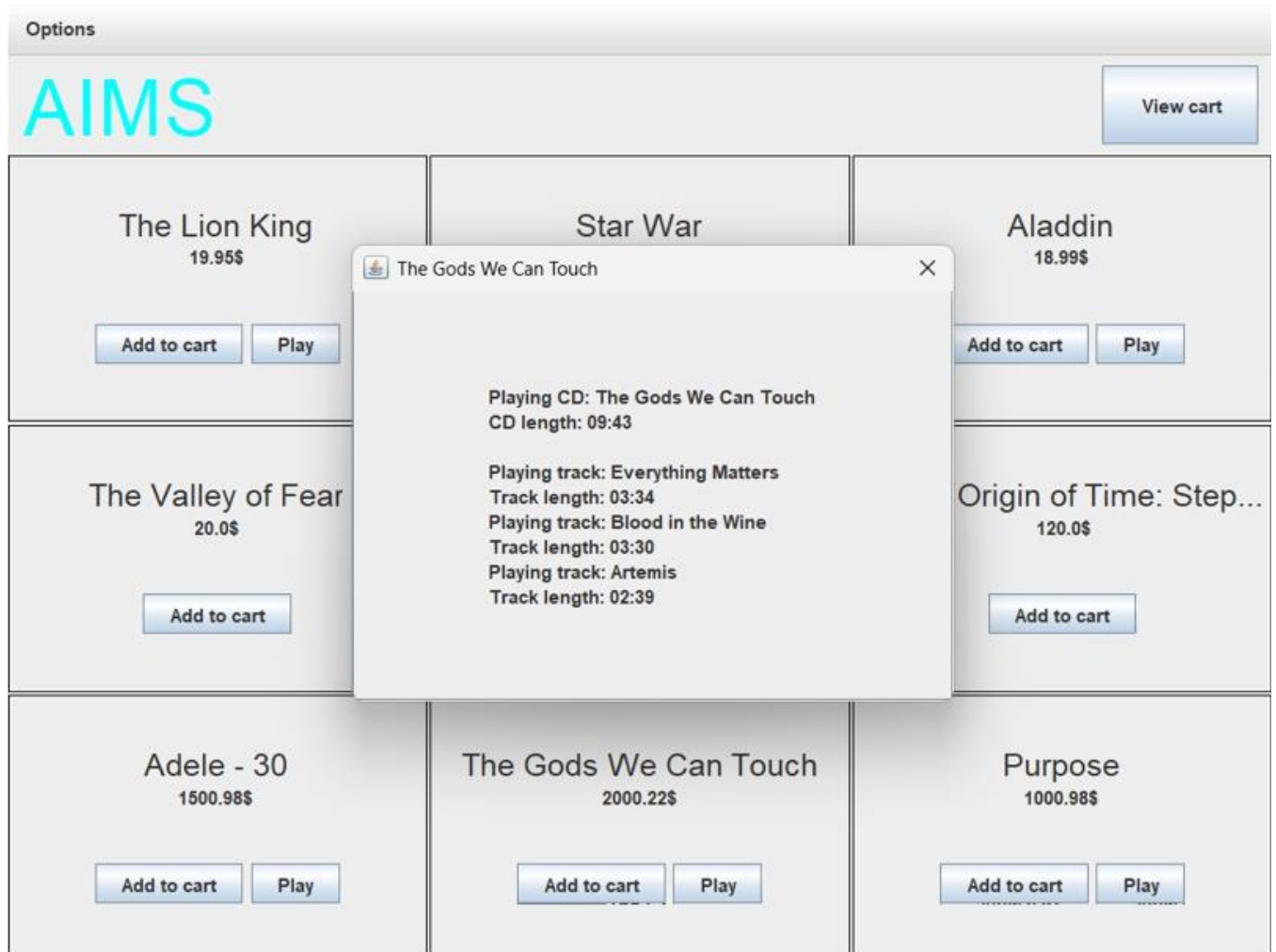


Figure 3.12 Demo Play button

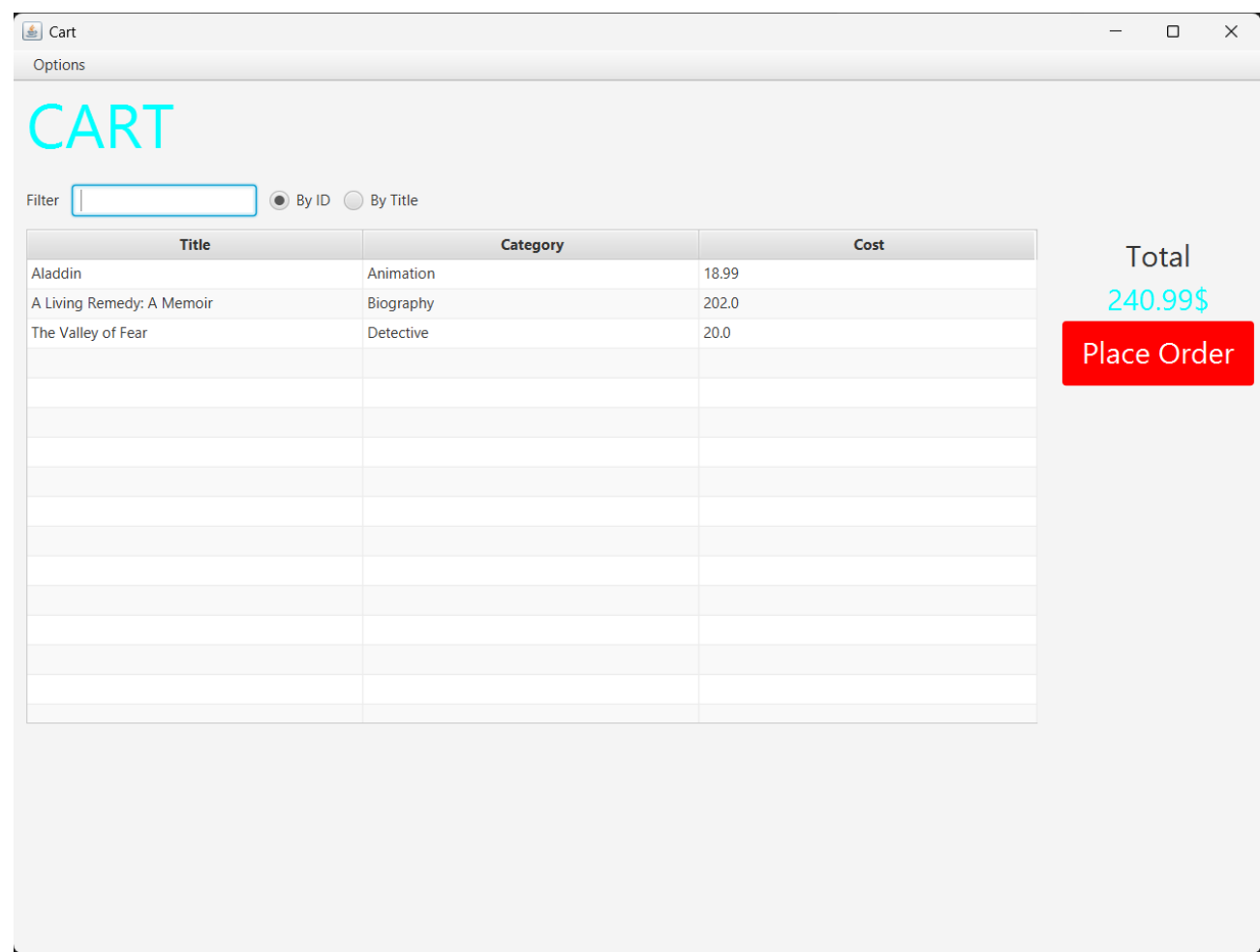


Figure 3.13 Demo View cart button

4 JavaFX API

4.1 Create class Painter

```
1 //Do Hoang Dong 225807
2 package javafx;
3
4 import javafx.application.Application;
5 import javafx.fxml.FXMLLoader;
6 import javafx.scene.Parent;
7 import javafx.scene.Scene;
8 import javafx.stage.Stage;
9
10 public class Painter extends Application {
11
12     @Override
13     public void start(Stage stage) throws Exception {
14         Parent root = FXMLLoader.load(getClass().
15             getResource(name: "javafx/Painter.fxml"));
16
17         Scene scene = new Scene(parent: root);
18         stage.setTitle(value: "Painter");
19         stage.setScene(value: scene);
20         stage.show();
21     }
22     public static void main(String[] args) {
23         launch(args);
24     }
25 }
```

Figure 4.1: Class Painte

4.2 Create Painter.fxml

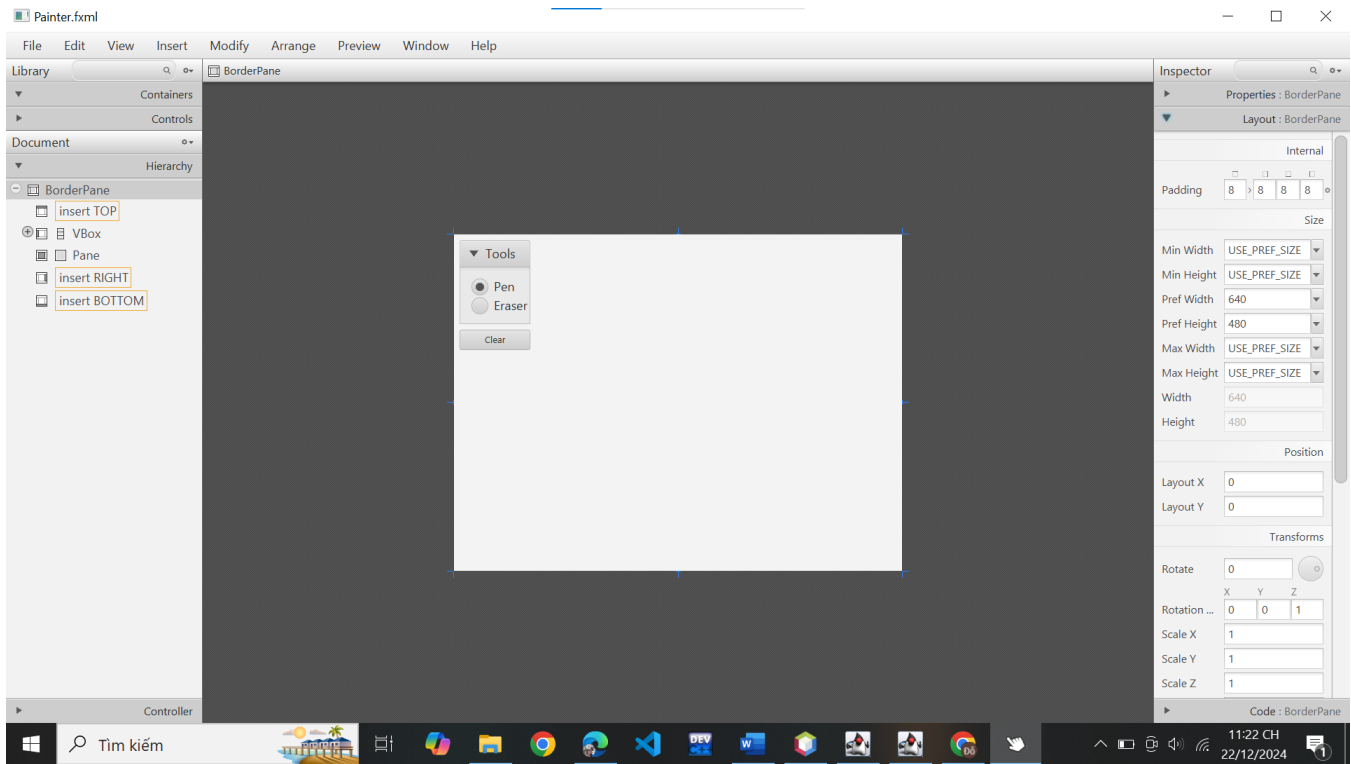


Figure 4.2: Painter.fxml 1

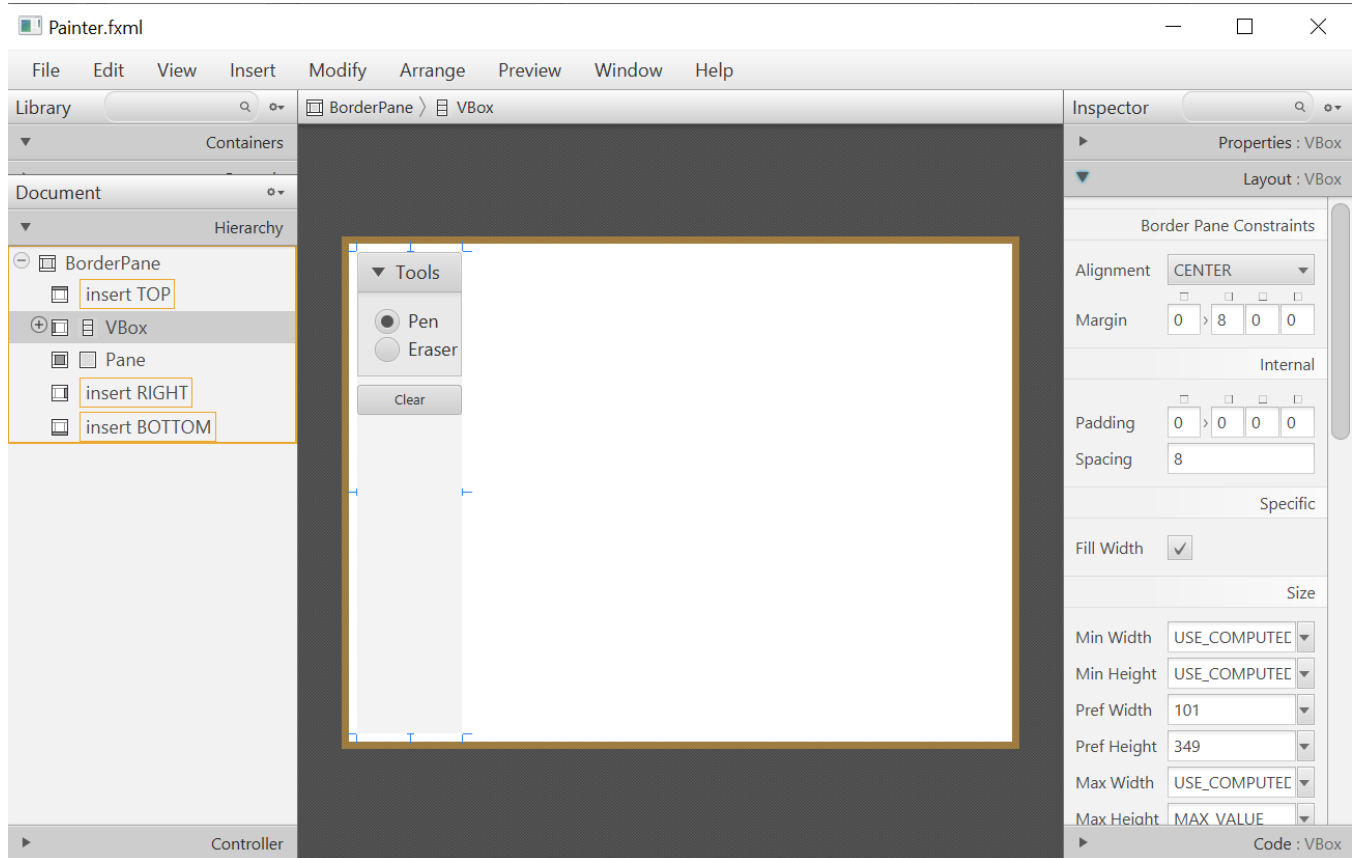


Figure 4.3: Painter.fxml 2

4.3 Create class PainterController

```
1 package javafx;
2
3 import java.net.URL;
4 import java.util.ResourceBundle;
5 import javafx.event.ActionEvent;
6 import javafx.fxml.FXML;
7 import javafx.fxml.Initializable;
8 import javafx.scene.control.RadioButton;
9 import javafx.scene.control.ToggleGroup;
10 import javafx.scene.input.MouseEvent;
11 import javafx.scene.layout.VBox;
12 import javafx.scene.paint.Color;
13 import javafx.scene.shape.Circle;
14 import javafx.scene.shape.Rectangle;
15
16 /**
17  * FXML Controller class
18  *
19  * @author HP
20  */
21 public class PainterController implements Initializable {
22
23     @FXML
24     private ResourceBundle resources;
25     @FXML
```

Figure 4.4: PainterController

```
21 public class PainterController implements Initializable {
22
23     @FXML
24     private ResourceBundle resources;
25     @FXML
26     private URL location;
27     @FXML
28     private VBox drawingAreaPane;
29     @FXML
30     private RadioButton pen;
31     @FXML
32     private ToggleGroup Tools;
33     @FXML
34     private RadioButton eraser;
35
36     /**
37      * Initializes the controller class.
38      */
39     @Override
40     public void initialize(URL url, ResourceBundle rb) {
41         assert drawingAreaPane != null :
42             "fx:id=\"drawingAreaPane\" was not injected: check your FXML file 'Painter.fxml'.";
43     }
44
45     @FXML
```

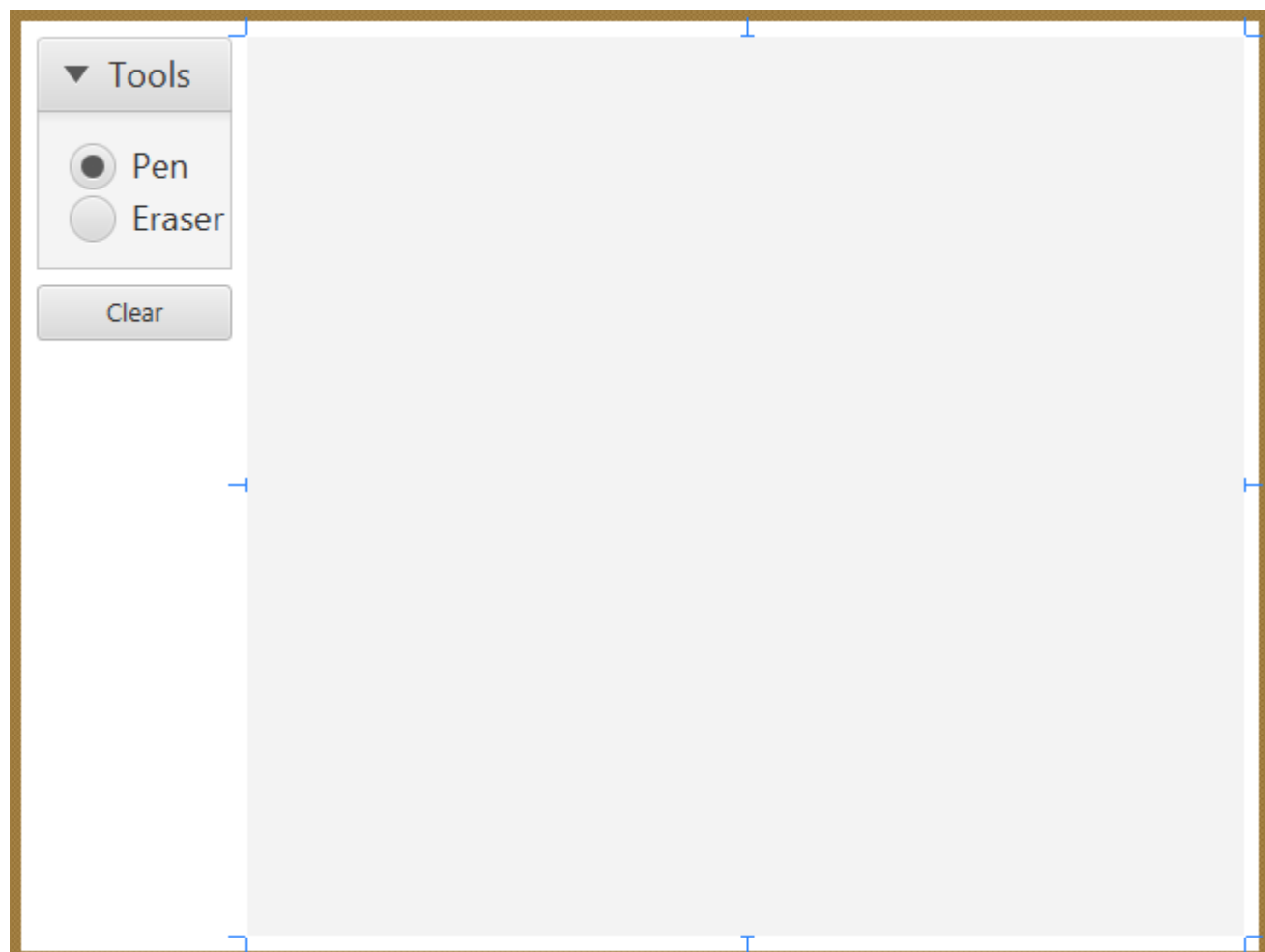


Figure 4.5:

5 View Cart Screen

Figure 5.3: Cart.fxml 3

5.1 Create class CartScreen

```
1  package hust.soict.hedspi.aims.Screen;
2
3  import javax.swing.JFrame;
4  import hust.soict.hedspi.aims.Screen.controller.CartScreenController;
5  import javafx.application.Platform;
6  import javafx.embed.swing.JFXPanel;
7  import javafx.fxml.FXMLLoader;
8  import javafx.scene.Parent;
9  import javafx.scene.Scene;
10 import src.hust.soict.hedspi.aims.Cart.Cart;
11
12 public class CartScreen extends JFrame {
13
14     private static Cart cart;
15
16     public static void main(String[] args) {
17         new CartScreen(cart);
18     }
19
20     public CartScreen(Cart cart) {
21
22         super();
23
24         CartScreen.cart = cart;
25
26         JFXPanel fxPanel = new JFXPanel();
```

```
23
24     CartScreen.cart = cart;
25
26     JFXPanel fxPanel = new JFXPanel();
27     this.add(comp: fxPanel);
28
29     this.setTitle(title: "Cart");
30     this.setSize(width: 1024, height: 768);
31     this.setVisible(b: true);
32     Platform.runLater(new Runnable() {
33         @Override
34         public void run() {
35             try {
36                 FXMLLoader loader = new FXMLLoader(url: getClass().getResource(name: "cart.fxml"));
37
38                 CartScreenController controller = new CartScreenController(cart);
39                 loader.setController(controller);
40                 Parent root = loader.load();
41                 fxPanel.setScene(new Scene(parent: root));
42             } catch (Exception e) {
43                 e.printStackTrace();
44             }
45         }
46     });
47 }
48 }
```

Create class CartScreenController

```
1 package hust.soict.hedspi.aims.Screen.controller;
2
3 import src.hust.soict.hedspi.aims.Cart.Cart;
4 import src.hust.soict.hedspi.aims.exception.PlayerException;
5 import src.hust.soict.hedspi.aims.media.Media;
6 import src.hust.soict.hedspi.aims.media.Playable;
7 import javafx.beans.value.ChangeListener;
8 import javafx.beans.value.ObservableValue;
9 import javafx.collections.transformation.FilteredList;
10 import javafx.event.ActionEvent;
11 import javafx.fxml.FXML;
12 import javafx.scene.control.*;
13 import javafx.scene.control.cell.PropertyValueFactory;
14
15 public class CartScreenController {
16     private Cart cart;
17
18     @FXML
19     private Button btnPlay;
20
21     @FXML
22     private Button btnRemove;
23
24     @FXML
25     private TableColumn<Media, Float> colMediaCost;
26 }
```

Figure 5.5: CartScreenController 1

```
28     private TableColumn<Media, String> colMediaTitle;
29
30     @FXML
31     private TableColumn<Media, String> colMediacategory;
32
33     @FXML
34     private TableView<Media> tblMedia;
35
36     @FXML
37     private ToggleGroup filterCategory;
38
39     @FXML
40     private RadioButton radioBtnFilterId;
41
42     @FXML
43     private RadioButton radioBtnFilterTitle;
44
45     @FXML
46     private Label costLabel;
47
48     @FXML
49     private TextField tfFilter;
50
51     @FXML
52     private Button placeOrder;
53
```

Figure 5.6: CartScreenController 2

```

80
81 @FXML
82 void btnRemovePressed(ActionEvent event) {
83     Media media = tblMedia.getSelectionModel().getSelectedItem();
84     cart.removeMedia(item: media);
85     costLabel.setText(cart.totalCost() + " $");
86 }
87
88 public CartScreenController(Cart cart) {
89     super();
90     this.cart = cart;
91 }
92
93 @FXML
94 void initialize() {
95     colMediaTitle.setCellValueFactory(
96         new PropertyValueFactory<Media, String>(string: "title")
97     );
98     colMediacategory.setCellValueFactory(
99         new PropertyValueFactory<Media, String>(string: "category")
100    );
101    colMediaCost.setCellValueFactory(
102        new PropertyValueFactory<Media, Float>(string: "cost")
103    );
104    //tblMedia.setItems(this.cart.getItemsOrdered());
105
106    costLabel.setText(cart.totalCost() + "$");
107
108    btnPlay.setVisible(value: false);
109    btnRemove.setVisible(value: false);
110
111    tblMedia.getSelectionModel().selectedItemProperty().addListener(
112        new ChangeListener<Media>() {
113
114            @Override
115            public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {
116                if (newValue != null) {
117                    updateButtonBar(media: newValue);
118                }
119            }
120
121            private void updateButtonBar(Media media) {
122                btnRemove.setVisible(value: true);
123                if (media instanceof Playable) {
124                    btnPlay.setVisible(value: true);
125                } else {
126                    btnPlay.setVisible(value: false);

```

```

124         btnPlay.setVisible(value: true);
125     } else {
126         btnPlay.setVisible(value: false);
127     }
128 }
129 }
130 };
131
132 tfFilter.textProperty().addListener(
133     new ChangeListener<String>() {
134         @Override
135         public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
136             showFilteredMedia(keyword: newValue);
137         }
138         private void showFilteredMedia(String keyword) {
139             FilteredList<Media> filteredList = new FilteredList<>(o1: cart.getItemsOrdered());
140
141             if (!keyword.isEmpty() && radioBtnFilterId.isSelected()) {
142                 filteredList.setPredicate(media -> {
143                     String idString = String.valueOf(media.getId());
144                     return idString.equals(keyword);
145                 });
146             } else if (!keyword.isEmpty() && radioBtnFilterTitle.isSelected()) {
147                 filteredList.setPredicate(media -> {
148                     String title = media.getTitle().toLowerCase();
149                     return title.contains(keyword.toLowerCase());

```

```

134         @Override
135         public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
136             showFilteredMedia(keyword: newValue);
137         }
138         private void showFilteredMedia(String keyword) {
139             FilteredList<Media> filteredList = new FilteredList<>(o1: cart.getItemsOrdered());
140
141             if (!keyword.isEmpty() && radioBtnFilterId.isSelected()) {
142                 filteredList.setPredicate(media -> {
143                     String idString = String.valueOf(media.getId());
144                     return idString.equals(keyword);
145                 });
146             } else if (!keyword.isEmpty() && radioBtnFilterTitle.isSelected()) {
147                 filteredList.setPredicate(media -> {
148                     String title = media.getTitle().toLowerCase();
149                     return title.contains(keyword.toLowerCase());
150                 });
151             } else {
152                 filteredList.setPredicate(predicate: null);
153             }
154             tblMedia.setItems(value: filteredList);
155         }
156     });
157 }
158 }

```


5.2 Demo

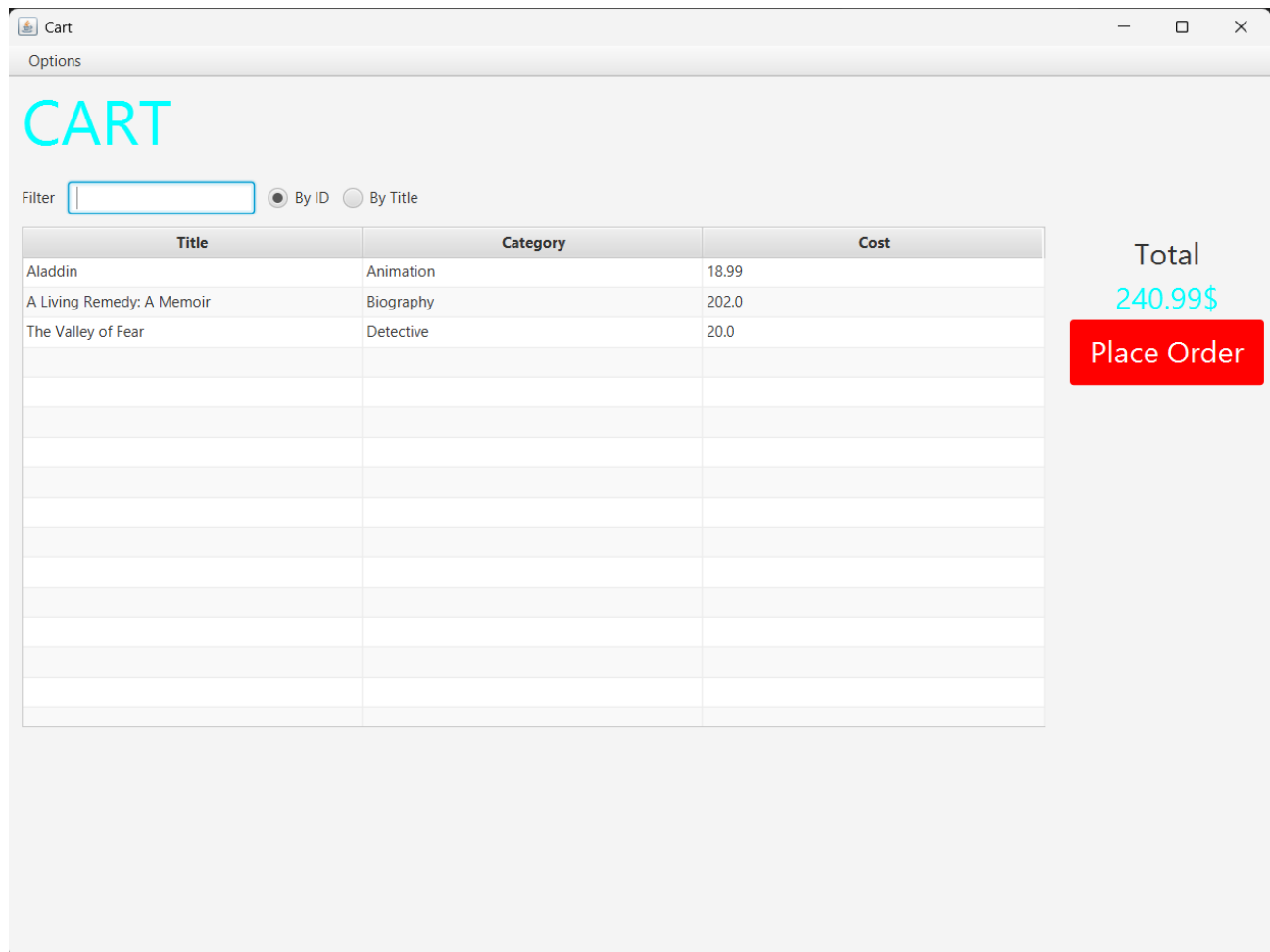


Figure 5.7: Demo CartScreen

6 Updating buttons based on selected item in TableView – ChangeListener

6.1 Edit class CartScreenController

```

        @Override
        public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {
            if (newValue != null) {
                updateButtonBar(media: newValue);
            }
        }

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

```

Figure 6.1: CartScreenController 1

```

tfFilter.textProperty().addListener(
    new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
            showFilteredMedia(keyword: newValue);
        }
        private void showFilteredMedia(String keyword) {
            FilteredList<Media> filteredList = new FilteredList<>();
            if (!keyword.isEmpty() && radioBtnFilterId.isSelected()) {
                filteredList.setPredicate(media -> {
                    String idString = String.valueOf(media.getId());
                    return idString.equals(keyword);
                });
            } else if (!keyword.isEmpty() && radioBtnFilterTitle.isSelected()) {
                filteredList.setPredicate(media -> {
                    String title = media.getTitle().toLowerCase();
                    return title.contains(keyword.toLowerCase());
                });
            } else {
                filteredList.setPredicate(predicate: null);
            }
            tblMedia.setItems(value: filteredList);
        }
    }
);

```

Figure 6.2: CartScreenController 2

6.2 Demo

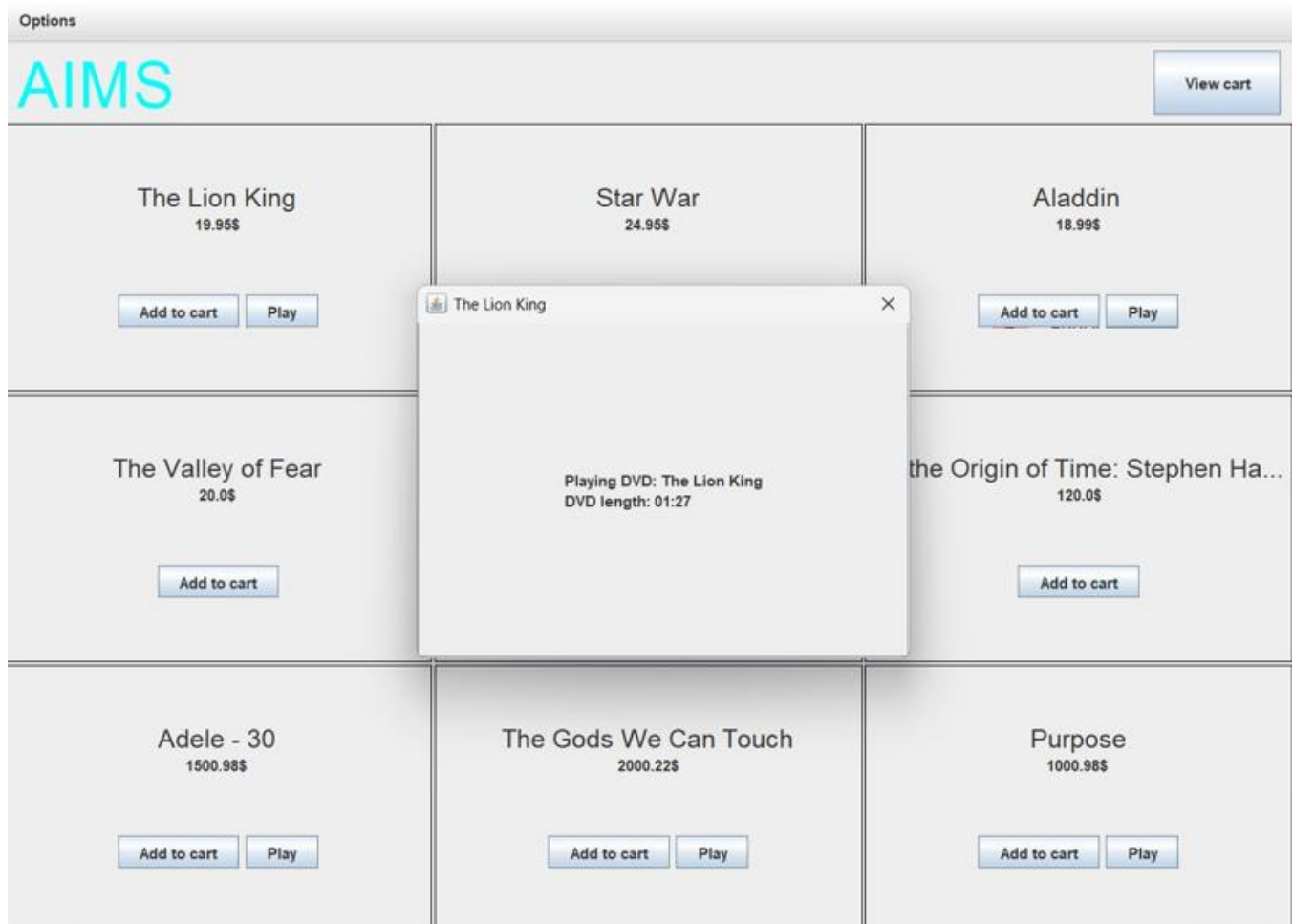


Figure 6.3: Demo media playable

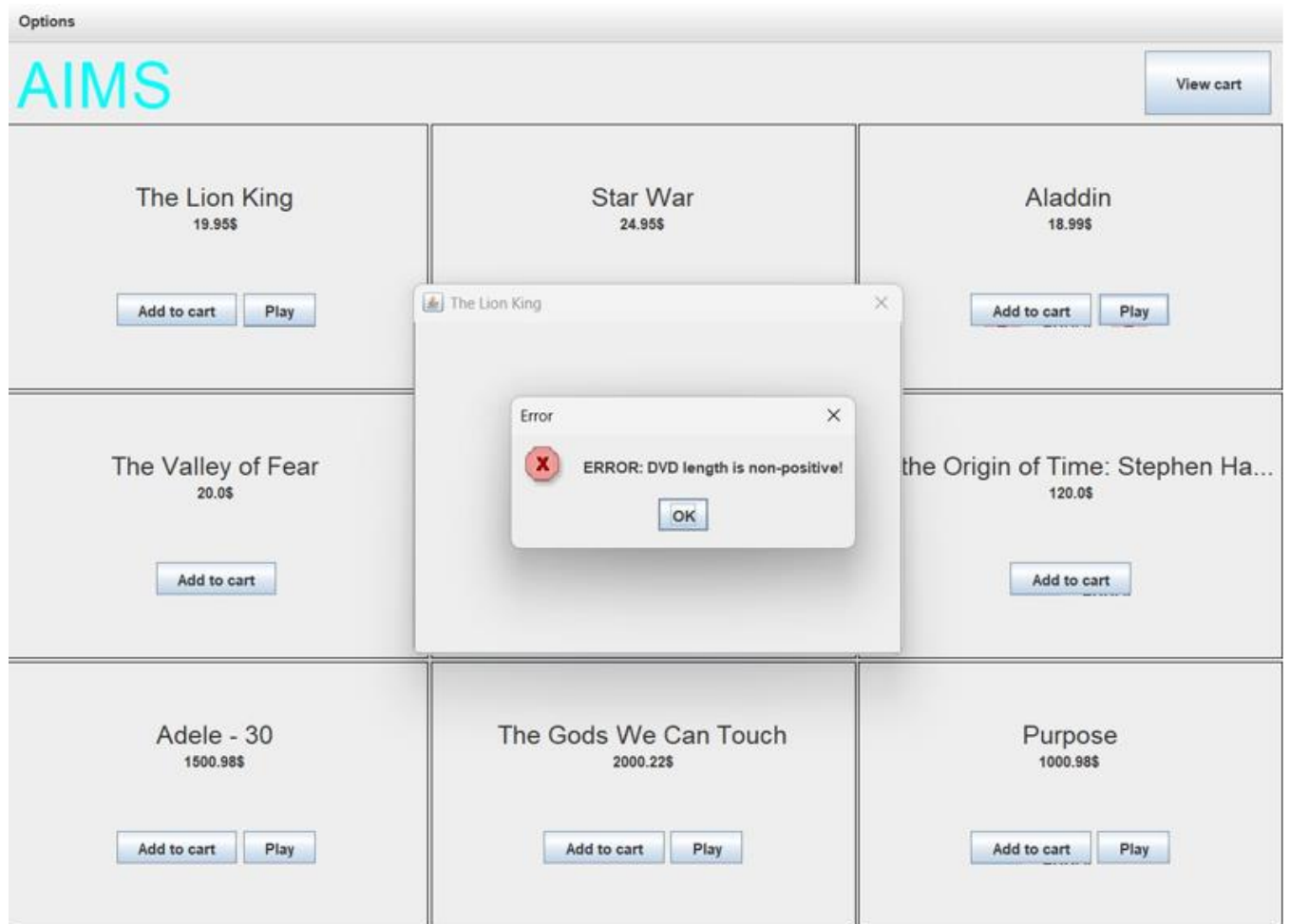


Figure 6.4: Demo media unplayable

7 Deleting a media

7.1 Code

```
62 //      removeMedia() {
63 //      void btnPlayPressed(ActionEvent event) {
64 //          Media media = tblMedia.getSelectionModel().getSelectedItem();
65 //          Alert alert;
66 //          try {
67 //              alert = new Alert(Alert.AlertType.NONE, media.playGUI());
68 //              alert.setTitle("Playing");
69 //              alert.setHeaderText(null);
70 //              alert.getDialogPane().getButtonTypes().add(ButtonType.OK);
71 //              alert.showAndWait();
72 //          } catch (PlayerException e) {
73 //              alert = new Alert(Alert.AlertType.ERROR, e.getMessage());
74 //              alert.setTitle("ERROR");
75 //              alert.setHeaderText(null);
76 //              alert.showAndWait();
77 //          }
78 //      }
79 //      }
80 }
```

Figure 7.1: btnRemovePressed Method

7.2 Demo

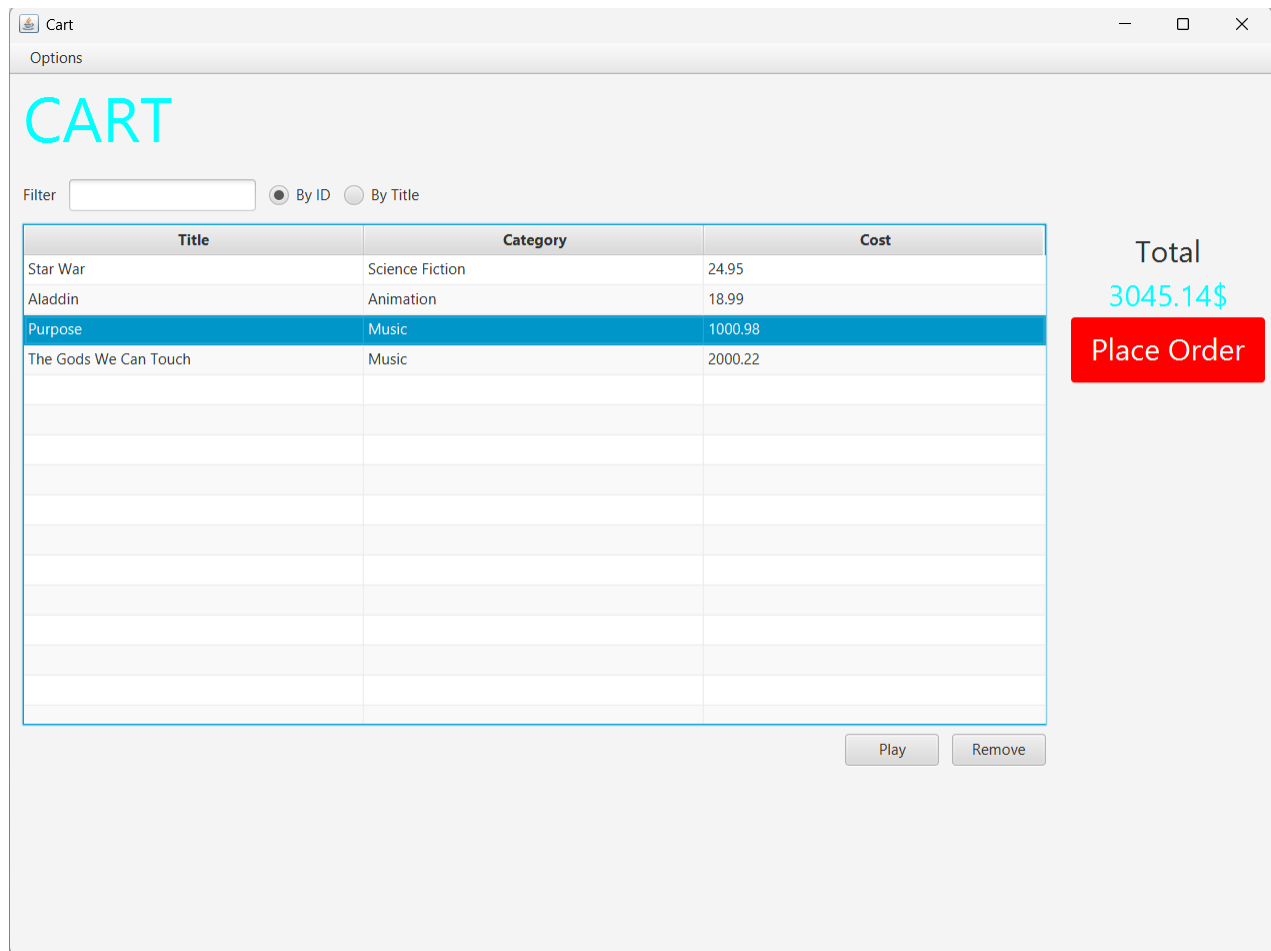


Figure 7.2: button Remove

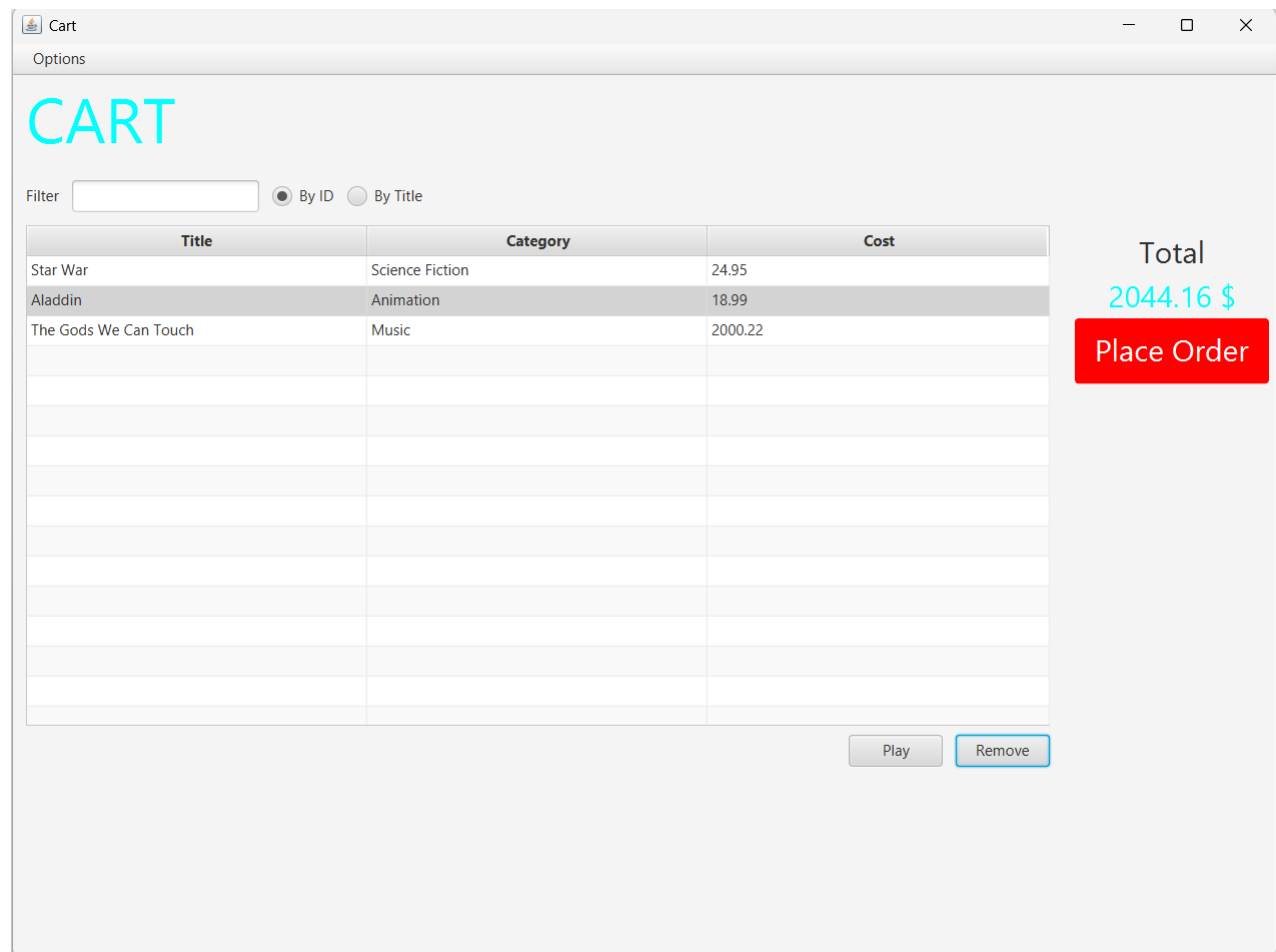


Figure 7.3: button Remove

8 Complete the Aims GUI application

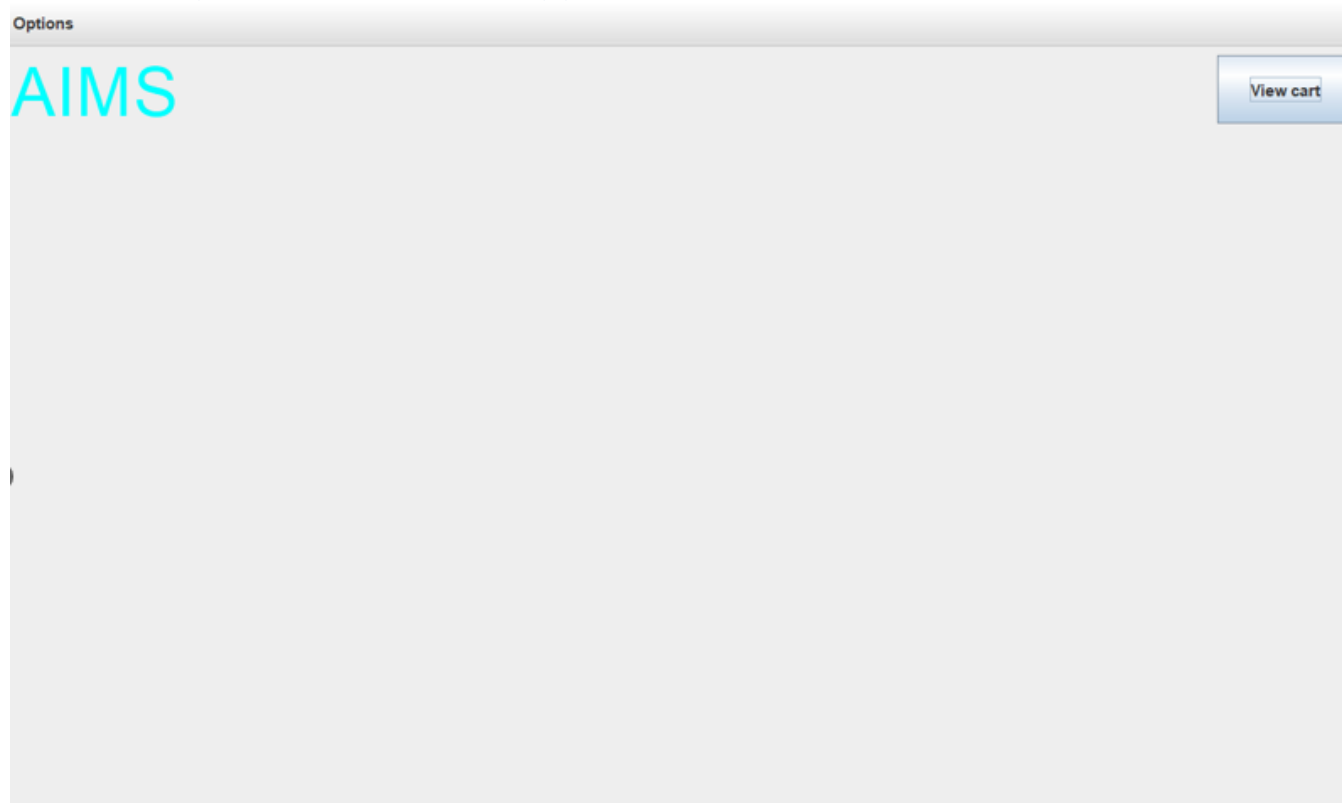


Figure 8.1: Store before add book

Options

ADD BOOK

Title

Category

Cost

Figure 8.2: Add book

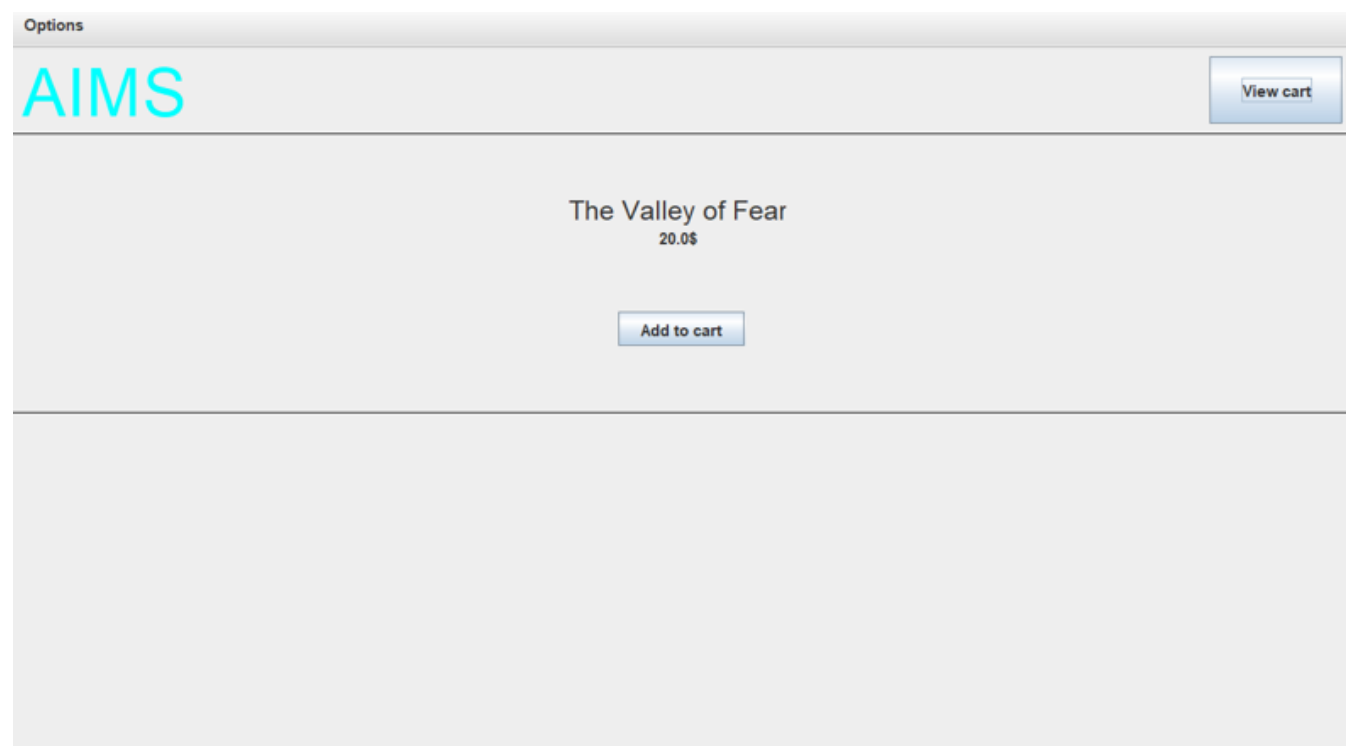


Figure 8.3: Store after add book

Options

ADD CD

Title

Category

Artist

Cost

Figure 8.4: Add CD

Options

ADD DVD

Title	<input type="text" value="The Lion King"/>
Category	<input type="text" value="Animation"/>
Director	<input type="text" value="Roger Aller"/>
Length	<input type="text" value="87"/>
Cost	<input type="text" value="19.95"/>

Save

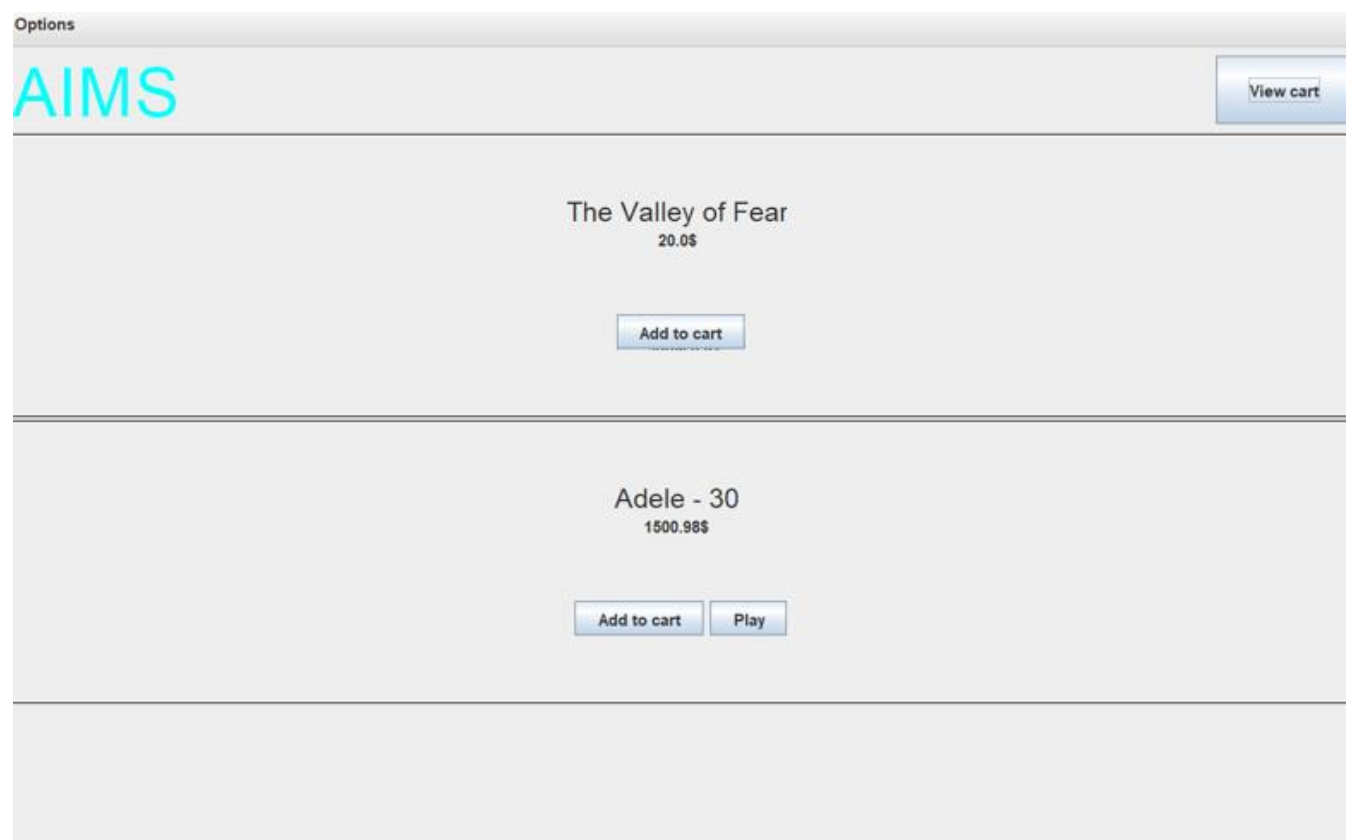
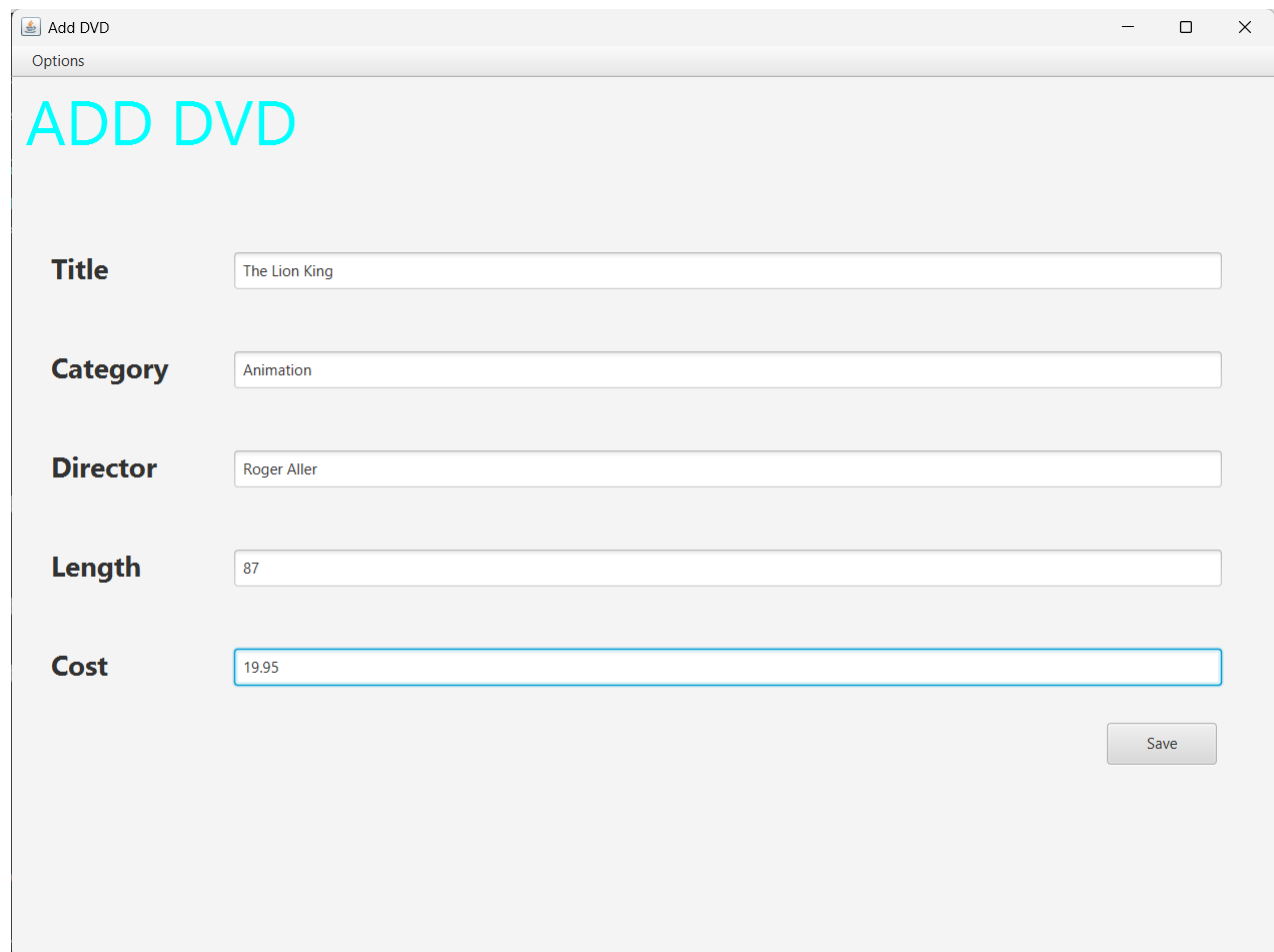


Figure 8.5: Store after add CD



The screenshot shows a window titled "Add DVD" with a standard macOS-style title bar (minimize, maximize, close buttons). Below the title bar is a menu bar with the word "Options". The main content area has a large cyan heading "ADD DVD". Below this heading are five form fields, each with a label to its left: "Title" (containing "The Lion King"), "Category" (containing "Animation"), "Director" (containing "Roger Aller"), "Length" (containing "87"), and "Cost" (containing "19.95"). The "Cost" field has a blue border. At the bottom right of the form is a "Save" button.

Field	Value
Title	The Lion King
Category	Animation
Director	Roger Aller
Length	87
Cost	19.95

Figure 8.6 Add DVD

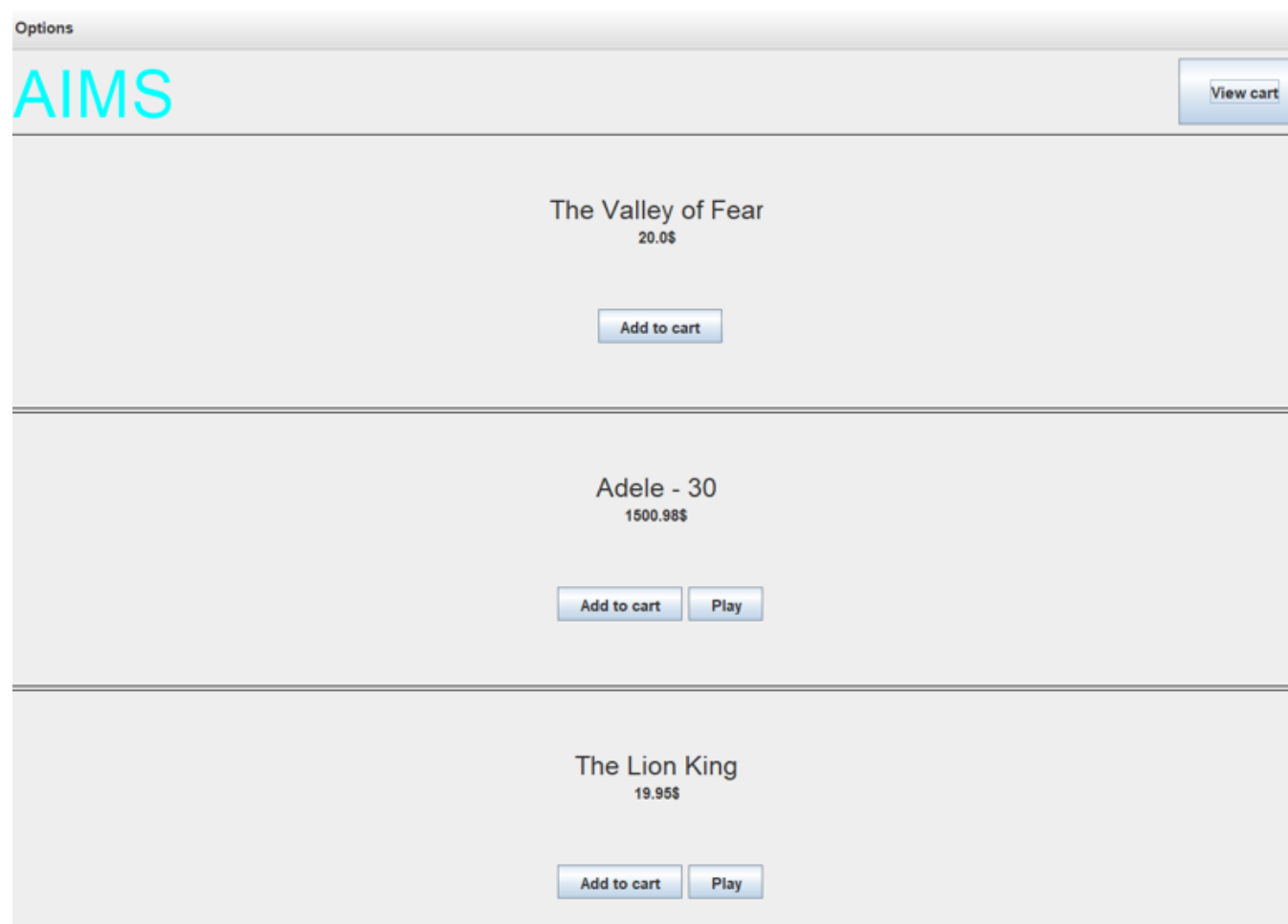


Figure 8.7: Store after add DVD

Options

CART

Filter

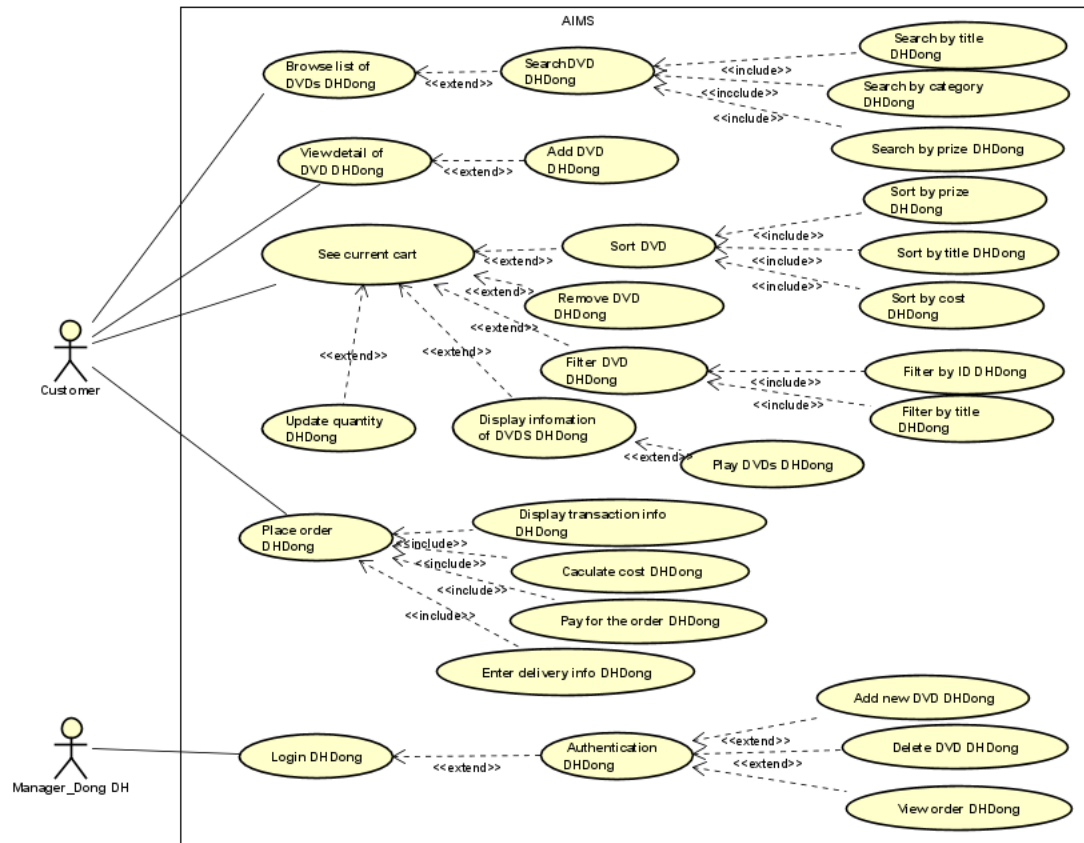
☒ By ID ☐ By Title

Title	Category	Cost
The Valley of Fear	Detective	20.0
Adele - 30	Music	1500.98

Total
1520.98\$
[Place Order](#)

Figure 8.8: Cart

9 Use case Diagram



10. Class Diagram

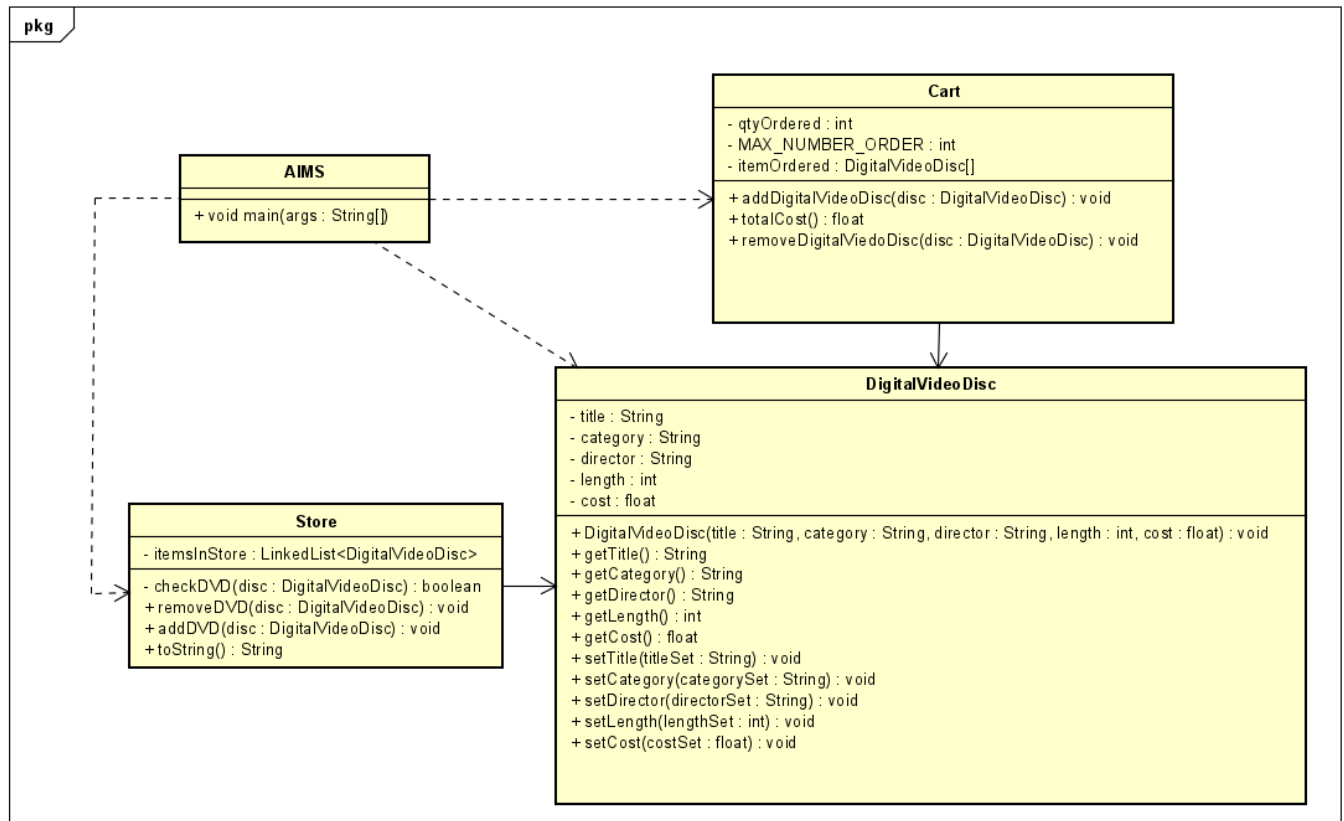


Figure 8.9: Exception