

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



BÁO CÁO THỰC HÀNH
IT3103-744527-2024.1
BÀI THỰC HÀNH – LAB 5

Họ và tên SV: Đặng Hồng Minh

MSSV: 20225740

Lớp: CNTT Việt Nhật – K67

GVHD: Lê Thị Hoa

HTGD: Đặng Mạnh Cường

Hà Nội 12/2024


Contents

1.	Swing components	2
1.1	AWTAccumulator	2
1.2	SwingAccumulator	3
2	Organizing Swing components with Layout Managers	4
2.1	Code	4
2.2	Demo	5
3	Create a graphical user interface for AIMS with Swing	6
3.1	Create class StoreScreen	6
3.2	Create class MediaStore	10
3.3	Demo	13
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	21
5.1	Create cart.fxml	21
5.2	Create class CartScreen	23
5.3	Create class CartScreenController	24
5.4	Demo	28
7	Deleting a media	30
7.1	Code	30
7.2	Demo	31
8	Complete the Aims GUI application	32
9	Use case Diagram	39
10	Class Diagram	40

BÁO CÁO THỰC HÀNH LAB 5

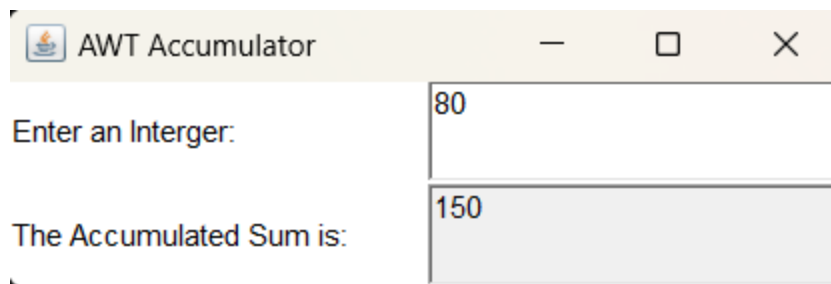
1. Swing components

1.1 AWTAccumulator

A screenshot of a code editor showing the source code for a Java Swing application named AWTAccumulator. The code is written in Java and is displayed on a dark background with syntax highlighting. The code includes package declarations, imports, a main class that extends JFrame, and an inner class that implements ActionListener. The main class sets up a GUI with two text fields, a label for input, and a label for the accumulated sum. The inner class handles the input and updates the sum. The code is numbered from 1 to 35 on the left side of the editor.

```
1 //Dang Hong Minh - 20225740
2 package GUIProject.hust.soict.hedspi.swing;
3 import java.awt.*;
4 import java.awt.event.*;
5 public class AWTAccumulator extends Frame {
6     private TextField tfInput;
7     private TextField tfOutput;
8     private int sum = 0;
9     public AWTAccumulator() {
10         setLayout(new GridLayout(rows:2,cols:2));
11         add(new Label(text:"Enter an Interger: "));
12         tfInput = new TextField(columns:10);
13         add(tfInput);
14         tfInput.addActionListener(new TFINputListener());
15         add(new Label(text:"The Accumulated Sum is: "));
16         tfOutput = new TextField(columns:10);
17         tfOutput.setEditable(b:false);
18         add(tfOutput);
19         setTitle(title:"AWT Accumulator");
20         setSize(width:350,height:120);
21         setVisible(b:true);
22     }
23     Run | Debug
24     public static void main (String arg[]) {
25         new AWTAccumulator();
26     }
27     private class TFINputListener implements ActionListener {
28         @Override
29         public void actionPerformed(ActionEvent evt) {
30             int numberIn = Integer.parseInt(tfInput.getText());
31             sum+= numberIn;
32             tfInput.setText(t:"");
33             tfOutput.setText(sum + "");
34         }
35     }
```

Figure 1.1: Source code of AWTAccumulator



1.2 SwingAccumulator

Figure 1.2: Demo of AWTAccumulator

```

1  //Dang Hong Minh - 20225740
2  package GUIProject.hust.soict.hedspi.swing;
3  import java.awt.*;
4  import java.awt.event.*;
5  import javax.swing.*;
6  public class SwingAccumulator extends JFrame {
7      private JTextField tfInput;
8      private JTextField tfOutput;
9      private int sum = 0;
10     public SwingAccumulator() {
11         Container cp = getContentPane();
12         cp.setLayout(new GridLayout(2,2));
13         cp.add(new JLabel(text:"Enter an Integer: "));
14         tfInput = new JTextField(columns:10);
15         cp.add(tfInput);
16         tfInput.addActionListener(new TFInputListener());
17         cp.add(new JLabel(text:"The Accumulated Sum is: "));
18         tfOutput = new JTextField(columns:10);
19         tfOutput.setEditable(b:false);
20         cp.add(tfOutput);
21         setTitle(title:"Swing Accumulator");
22         setSize(width:350, height:120);
23         setVisible(b:true);
24     }

```

Figure 1.3: Source code of SwingAccumulator

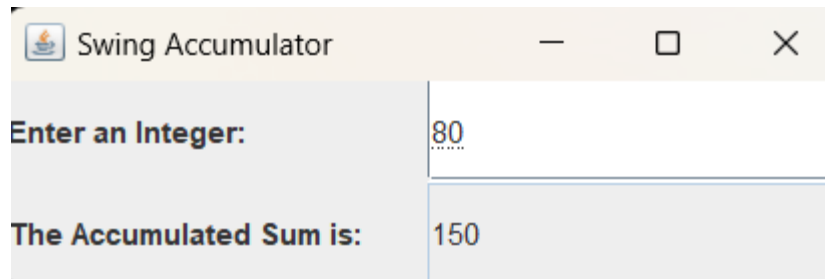


Figure 1.4: Demo of SwingAccumulator

2 Organizing Swing components with Layout Managers

2.1 Code

```

1  //Dang Hong Minh - 20225740
2  package GUIProject.hust.soict.hedspi.swing;
3  import java.awt.*;
4  import java.awt.event.*;
5  import javax.swing.*;
6  public class NumberGrid extends JFrame {
7      private JButton[] btnNumbers = new JButton[10];
8      private JButton btnDelete, btnReset;
9      private JTextField tfDisplay;
10     public NumberGrid() {
11         tfDisplay = new JTextField();
12         tfDisplay.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
13         JPanel panelButtons = new JPanel(new GridLayout(rows:4,cols:3));
14         addButtons(panelButtons);
15         Container cp = getContentPane();
16         cp.setLayout(new BorderLayout());
17         cp.add(tfDisplay, BorderLayout.NORTH);
18         cp.add(panelButtons, BorderLayout.CENTER);
19         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         setTitle(title:"Number Grid");
21         setSize(width:200,height:200);
22         setVisible(b:true);
23     }
24     void addButtons(JPanel panelButtons) {
25         ButtonListener btnListener = new ButtonListener();
26         for(int i = 1; i<=9;i++) {
27             btnNumbers[i]= new JButton(""+i);
28             panelButtons.add(btnNumbers[i]);
29             btnNumbers[i].addActionListener(btnListener);
30         }
31         btnDelete = new JButton(text:"DEL");
32         panelButtons.add(btnDelete);

```

Figure 2.1: Source code of NumberGrid 1

```

33      btnDelete.addActionListener(btnListener);
34      btnNumbers[0] = new JButton(text:"0");
35      panelButtons.add(btnNumbers[0]);
36      btnNumbers[0].addActionListener(btnListener);
37      btnReset = new JButton(text:"C");
38      panelButtons.add(btnReset);
39      btnReset.addActionListener(btnListener);
40  }
41  private class ButtonListener implements ActionListener {
42      @Override
43      public void actionPerformed(ActionEvent e) {
44          String button = e.getActionCommand();
45          if(button.charAt(index:0)>= '0' && button.charAt(index:0)<= '9') {
46              tfDisplay.setText(tfDisplay.getText()+button);
47          }
48          else if (button.equals(anObject:"DEL")) {
49              //handles the "DEL" case
50              String text = tfDisplay.getText();
51              if (text.length() > 0) {
52                  tfDisplay.setText(text.substring(beginIndex:0, text.length() - 1));
53              }
54          }
55          else {
56              //handles the "C" case
57              tfDisplay.setText(t:"");
58          }
59      }
60  }
61  Run | Debug
62  public static void main(String arg[]) {
63      new NumberGrid();
64  }

```

Figure 2.2: Source code of NumberGrid 2

2.2 Demo

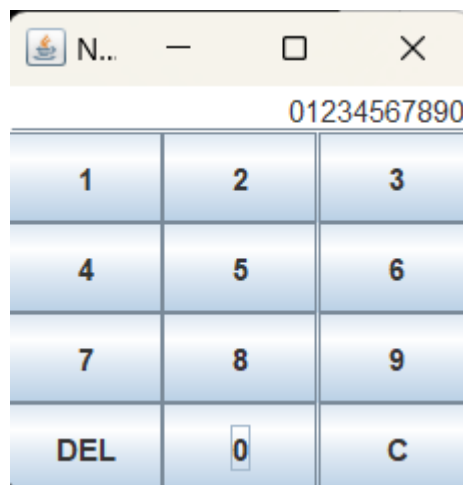


Figure 2.3: Demo buttons 0-9

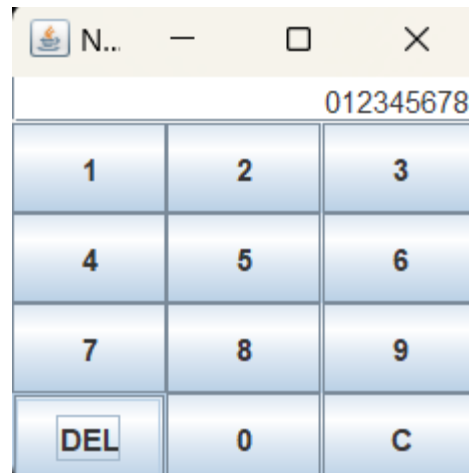


Figure 2.4: Demo DEL button

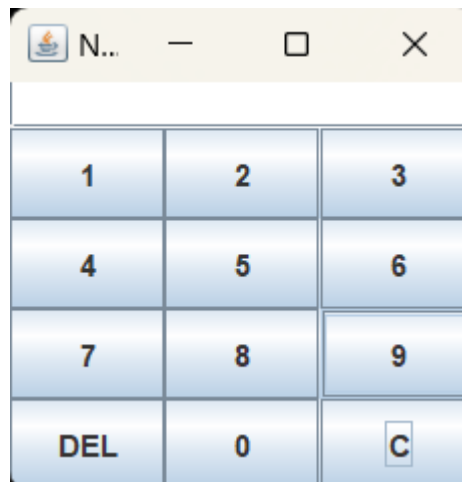


Figure 2.5: Demo C button

3 Create a graphical user interface for AIMS with Swing

3.1 Create class StoreScreen

```

1 //Dang Hong Minh - 20225740
2 package AimsProject.hust.soict.hedspi.aims.screen;
3 import AimsProject.hust.soict.hedspi.aims.store.*;
4 import AimsProject.hust.soict.hedspi.aims.cart.*;
5 import AimsProject.hust.soict.hedspi.aims.media.*;
6 import java.awt.*;
7 import java.awt.event.*;
8 import java.util.ArrayList;
9 import javax.swing.*;
10 public class StoreScreen extends JFrame {
11     private Store store;
12     private Cart cart;
13     public static void main(String[] args) throws Exception {
14         Store myStore = new Store();
15         Cart myCart = new Cart();
16         //Create media
17         DigitalVideoDisc dvd1 = new DigitalVideoDisc(title:"The Goblin King", category:"Animation", director:"Someone", length:87, cost:19.95f);
18         DigitalVideoDisc dvd2 = new DigitalVideoDisc(title:"Star Wars", category:"Sci-fi", director:"George Lucas", length:87, cost:24.95f);
19         DigitalVideoDisc dvd3 = new DigitalVideoDisc(title:"Aladin", category:"Animation", cost:18.99f);
20         Track track1 = new Track(title:"Ep 1", length:10);
21         Track track2 = new Track(title:"Ep 2 + 3", length:21);
22         Track track3 = new Track(title:"Ep 4", length:15);
23         CompactDisc cd1 = new CompactDisc(title:"Bishoku - SS1", category:"Animation", artist:"Various", cost:30.95f);
24         cd1.addTrack(track1);
25         cd1.addTrack(track2);
26         CompactDisc cd2 = new CompactDisc(title:"Bishoku - SS2", category:"Animation", artist:"Various", cost:25.99f);
27         cd2.addTrack(track3);
28         Book book1 = new Book(title:"Vietnam Encyclopedia", category:"Science", cost:159.99f);
29         book1.addAuthor(authorName:"Wei Zhu");
30         Media dvd4 = new DigitalVideoDisc(title:"Movie 04", category:"Genre C", director:"Director", length:120, cost:12.90f);
31         Media dvd5 = new DigitalVideoDisc(title:"Movie 05", category:"Genre D", director:"Director", length:90, cost:9.9f);
32         Media dvd6 = new DigitalVideoDisc(title:"Movie 06", category:"Genre C", director:"Director", length:100, cost:11.88f);
33         Media dvd7 = new DigitalVideoDisc(title:"Movie 07", category:"Genre E", director:"Director", length:89, cost:8.68f);
34         Media dvd8 = new DigitalVideoDisc(title:"Movie 08", category:"Genre F", director:"Also Director", length:180, cost:20.1f);
35         Media dvd9 = new DigitalVideoDisc(title:"Movie 09", category:"Genre F", director:"Also Director", length:160, cost:18.4f);
36         //Add media
37         myStore.addMedia(dvd1);
38         myStore.addMedia(dvd2);
39         myStore.addMedia(dvd3);
40         myStore.addMedia(cd1);
41         myStore.addMedia(cd2);
42         myStore.addMedia(book1);
43         myStore.addMedia(dvd4);
44         myStore.addMedia(dvd5);

```

Figure 3.1: Class StoreScreen 1


```

45     myStore.addMedia(dvd6);
46     myStore.addMedia(dvd7);
47     myStore.addMedia(dvd8);
48     myStore.addMedia(dvd9);
49     new StoreScreen(myStore, myCart);
50 }
51 public StoreScreen(Store store, Cart cart) {
52     this.store = store;
53     this.cart = cart;
54     Container cp = getContentPane();
55     cp.setLayout(new BorderLayout());
56     cp.add(createNorth(), BorderLayout.NORTH);
57     cp.add(createCenter(), BorderLayout.CENTER);
58     setVisible(b:true);
59     setTitle(title:"Store");
60     setSize(width:1024, height:768);
61     Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
62     int w = getSize().width;
63     int h = getSize().height;
64     int x = (dim.width - w) / 2;
65     int y = (dim.height - h) / 2;
66     setLocation(x, y);
67 }
68 JPanel createNorth() {
69     JPanel north = new JPanel();
70     north.setLayout(new BoxLayout(north, BoxLayout.Y_AXIS));
71     north.add(createMenuBar());
72     north.add(createHeader());
73     return north;
74 }
75 JPanel createCenter() {
76     JPanel center = new JPanel();
77     center.setLayout(new GridLayout(rows:5, cols:5, hgap:2, vgap:2));
78     ArrayList<Media> mediaInStore = (ArrayList<Media>) store.getItemsInStore();
79     for (Media media : mediaInStore) {
80         MediaStore cell = new MediaStore(media, cart);
81         center.add(cell);
82     }
83     return center;
84 }
85 JMenuBar createMenuBar() {
86     JMenu menu = new JMenu(s:"Options");
87     JMenu smUpdateStore = new JMenu(s:"Update Store");

```

Figure 3.2: Class StoreScreen 2

```

88     JMenuItem addBook = new JMenuItem(text:"Add Book");
89     addBook.addActionListener(new AddBookListener());
90     smUpdateStore.add(addBook);
91     JMenuItem addCD = new JMenuItem(text:"Add CD");
92     addCD.addActionListener(new AddCDListener());
93     smUpdateStore.add(addCD);
94     JMenuItem addDVD = new JMenuItem(text:"Add DVD");
95     addDVD.addActionListener(new AddDVDListener());
96     smUpdateStore.add(addDVD);
97     menu.add(smUpdateStore);
98     menu.add(new JMenuItem(text:"View store"));
99     JMenuItem cart = new JMenuItem(text:"View cart");
100    cart.addActionListener(new ViewCartListener());
101    menu.add(cart);
102    JMenuBar menuBar = new JMenuBar();
103    menuBar.setLayout(new FlowLayout(FlowLayout.LEFT));
104    menuBar.add(menu);
105    return menuBar;
106 }
107 JPanel createHeader() {
108     JPanel header = new JPanel();
109     header.setLayout(new BoxLayout(header, BoxLayout.X_AXIS));
110     JLabel title = new JLabel(text:"AIMS:~$ MinhDH5740");
111     title.setFont(new Font(title.getFont().getName(), Font.PLAIN, size:50));
112     title.setForeground(Color.CYAN);
113     JButton cart = new JButton(text:"View cart");
114     cart.setPreferredSize(new Dimension(width:100, height:50));
115     cart.setMaximumSize(new Dimension(width:100, height:50));
116     cart.addActionListener(new ViewCartListener());
117     header.add(Box.createRigidArea(new Dimension(width:10, height:10)));
118     header.add(title);
119     header.add(Box.createHorizontalGlue());
120     header.add(cart);
121     header.add(Box.createRigidArea(new Dimension(width:10, height:10)));
122     return header;
123 }
124 private class ViewCartListener implements ActionListener {
125     @Override
126     public void actionPerformed(ActionEvent e) {
127         new CartScreen(store, cart);
128         dispose();
129     }
130 }

```

Figure 3.3: Class StoreScreen 3

```
131     private class AddDVDListener implements ActionListener {
132     |     @Override
133     |     public void actionPerformed(ActionEvent e) {
134     |         new AddDVDToStoreScreen(store, cart);
135     |         dispose();
136     |     }
137     |
138     | }
139     private class AddBookListener implements ActionListener {
140     |     @Override
141     |     public void actionPerformed(ActionEvent e) {
142     |         new AddBookToStoreScreen(store, cart);
143     |         dispose();
144     |     }
145     | }
146     private class AddCDListener implements ActionListener {
147     |     @Override
148     |     public void actionPerformed(ActionEvent e) {
149     |         new AddCDToStoreScreen(store, cart);
150     |         dispose();
151     |     }
152     | }
153 }
```

Figure 3.4: Class StoreScreen 4

3.2 Create class MediaStore

```

1 //Dang Hong Minh - 20225740
2 package AimsProject.hust.soict.hedspi.aims.screen;
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6 import AimsProject.hust.soict.hedspi.aims.exception.*;
7 import AimsProject.hust.soict.hedspi.aims.media.*;
8 import AimsProject.hust.soict.hedspi.aims.cart.Cart;
9 public class MediaStore extends JPanel {
10     private Media media;
11     private Cart cart;
12     public MediaStore(Media media, Cart cart) {
13         this.media = media;
14         this.cart = cart;
15         this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
16         JLabel title = new JLabel(media.getTitle());
17         title.setFont(new Font(title.getFont().getName(), Font.PLAIN, size:20));
18         title.setAlignmentX(CENTER_ALIGNMENT);
19         JLabel cost = new JLabel("" + media.getCost() + " $");
20         cost.setAlignmentX(CENTER_ALIGNMENT);
21         JPanel container = new JPanel();
22         container.setLayout(new FlowLayout(FlowLayout.CENTER));
23         JButton addToCartButton = new JButton(text:"Add to cart");
24         addToCartButton.addActionListener(new AddToCartListener());
25         container.add(addToCartButton);
26         JButton detailsButton = new JButton(text:"View details");
27         detailsButton.addActionListener(new DetailsListener());
28         container.add(detailsButton);
29         if (media instanceof Playable) {
30             JButton playButton = new JButton(text:"Play");
31             playButton.addActionListener(new PlayButtonListener());
32             container.add(playButton);
33         }
34         this.add(Box.createVerticalGlue());
35         this.add(title);
36         this.add(cost);
37         this.add(Box.createVerticalGlue());
38         this.add(container);
39         this.setBorder(BorderFactory.createLineBorder(Color.BLACK));
40     }
41     private class PlayButtonListener implements ActionListener {
42         @Override
43         public void actionPerformed(ActionEvent e) {
44             try {
45                 ((Disc) media).play();

```

Figure 3.7: Class MediaStore 1

```

46     } catch (PlayerException ex) {
47         JPanel p = new JPanel();
48         JDialog d = new JDialog();
49         JLabel j1 = new JLabel(media.getTitle() + " cannot be played");
50         JLabel j2 = new JLabel(text:"Media length is non-positive");
51         p.setLayout(new BoxLayout(p, BoxLayout.Y_AXIS));
52         j1.setAlignmentX(Component.CENTER_ALIGNMENT);
53         j2.setAlignmentX(Component.CENTER_ALIGNMENT);
54         p.add(Box.createVerticalGlue());
55         p.add(j1);
56         p.add(j2);
57         p.add(Box.createVerticalGlue());
58         d.add(p);
59         d.setSize(width:250, height:100);
60         Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
61         int w = d.getSize().width;
62         int h = d.getSize().height;
63         int x = (dim.width - w) / 2;
64         int y = (dim.height - h) / 2;
65         d.setLocation(x, y);
66         d.setVisible(b:true);
67     }
68 }
69 }
70 private class AddToCartListener implements ActionListener {
71     @Override
72     public void actionPerformed(ActionEvent e) {
73         JPanel p = new JPanel();
74         JDialog d = new JDialog();
75         JLabel l = new JLabel();
76         try {
77             cart.addMedia(media);
78             l.setText(media.getTitle() + " added to cart");
79         } catch (CartFullException ex) {
80             l.setText(text:"The cart is full");
81         } finally {
82             p.setLayout(new BoxLayout(p, BoxLayout.Y_AXIS));
83             l.setAlignmentX(Component.CENTER_ALIGNMENT);
84             p.add(Box.createVerticalGlue());
85             p.add(l);
86             p.add(Box.createVerticalGlue());
87             d.add(p);

```

Figure 3.8: Class MediaStore 2

```

88         d.setSize(width:200, height:100);
89         Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
90         int w = d.getSize().width;
91         int h = d.getSize().height;
92         int x = (dim.width - w) / 2;
93         int y = (dim.height - h) / 2;
94         d.setLocation(x, y);
95         d.setVisible(b:true);
96     }
97 }
98
99 private class DetailsListener implements ActionListener {
100     @Override
101     public void actionPerformed(ActionEvent e) {
102         JPanel p = new JPanel();
103         JDialog d = new JDialog();
104         JLabel l = new JLabel("<html>" + media.getDetails().replaceAll(regex:"\\n", replacement:"<br/>")
105 + "</html>", SwingConstants.CENTER);
106         p.setLayout(new BoxLayout(p, BoxLayout.Y_AXIS));
107         l.setAlignmentX(Component.CENTER_ALIGNMENT);
108         p.add(Box.createVerticalGlue());
109         p.add(l);
110         p.add(Box.createVerticalGlue());
111         d.add(p);
112         d.setSize(width:200, height:200);
113         Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
114         int w = d.getSize().width;
115         int h = d.getSize().height;
116         int x = (dim.width - w) / 2;
117         int y = (dim.height - h) / 2;
118         d.setLocation(x, y);
119         d.setVisible(b:true);
120     }
121 }
122 }

```

Figure 3.9: Class MediaStore 3

3.3 Demo



Figure 3.10: StoreScreen

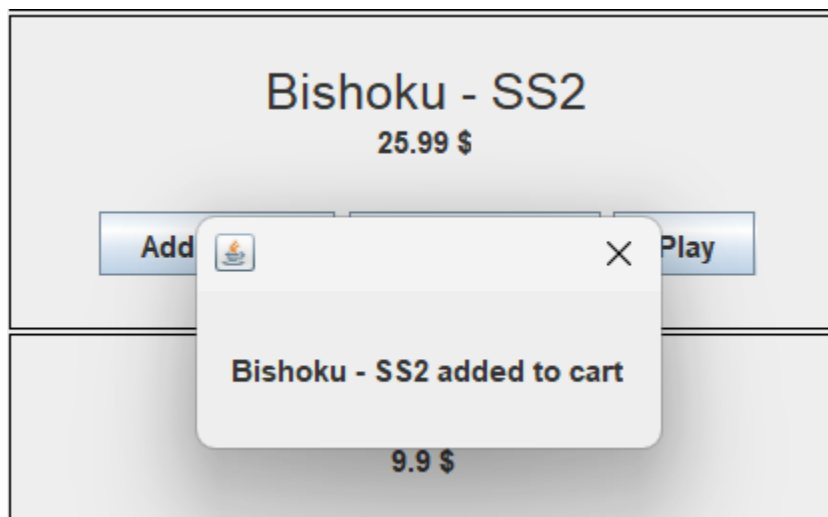


Figure 3.11 Demo Add to cart button

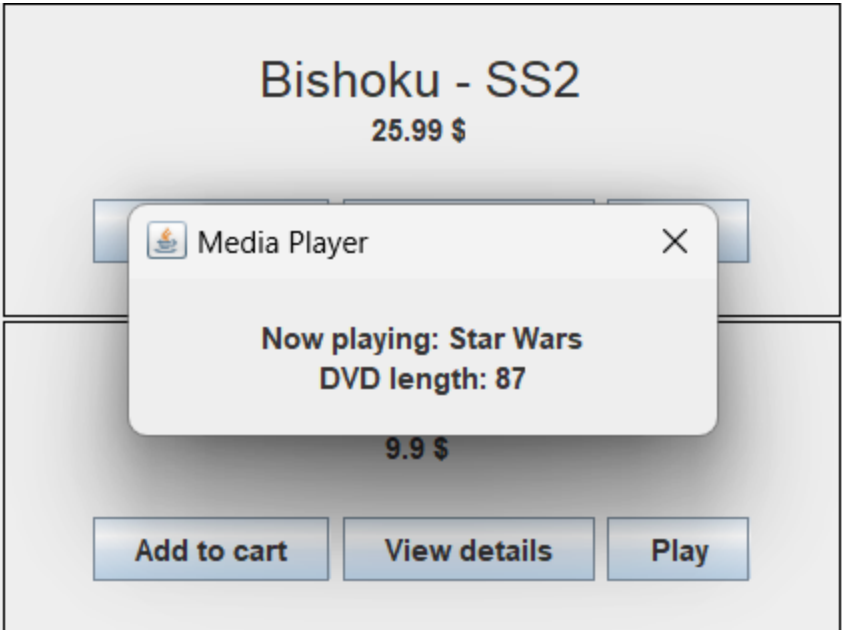


Figure 3.12 Demo Play button

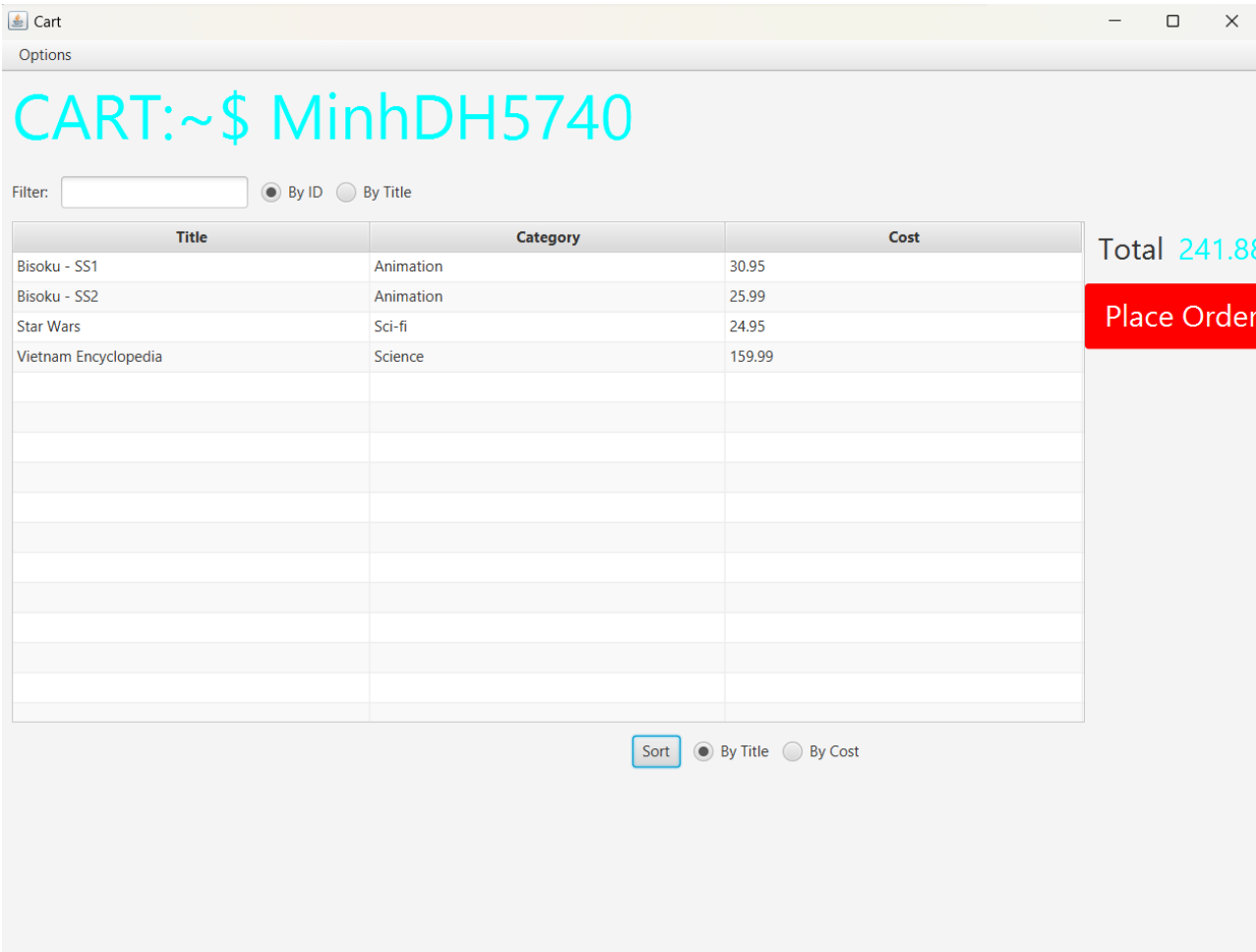
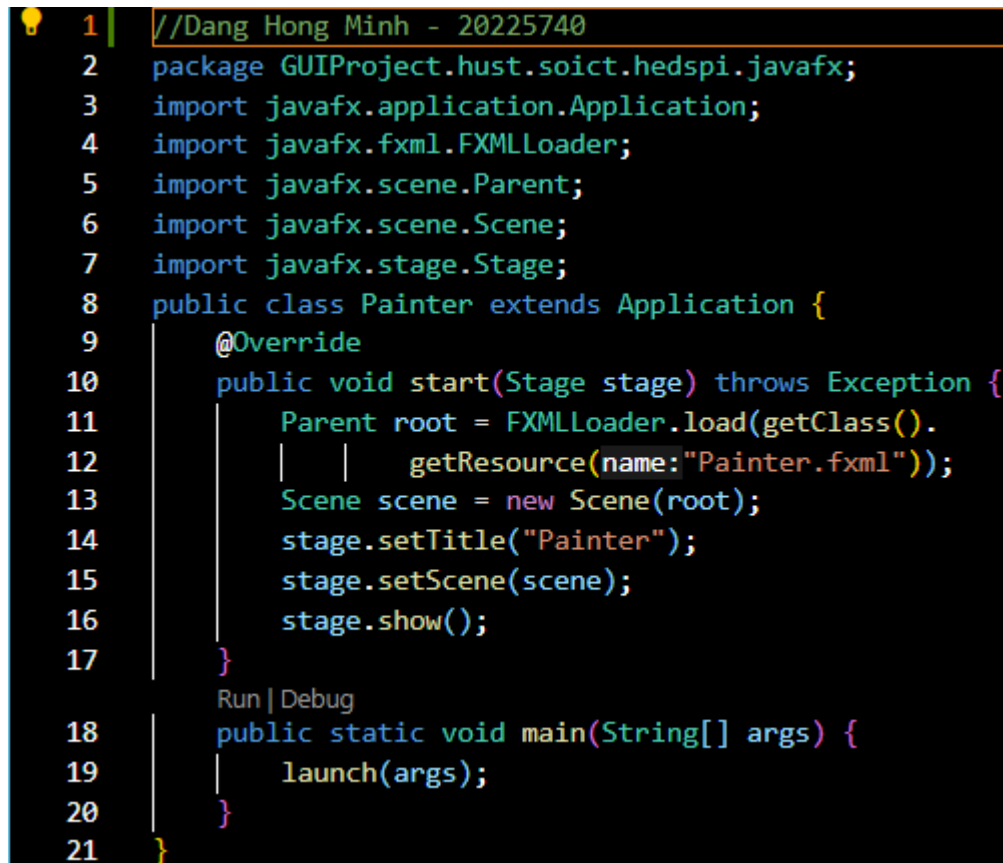


Figure 3.13 Demo View cart button

4 JavaFX API

4.1 Create class Painter



```
1 //Dang Hong Minh - 20225740
2 package GUIProject.hust.soict.hedspi.javafx;
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Parent;
6 import javafx.scene.Scene;
7 import javafx.stage.Stage;
8 public class Painter extends Application {
9     @Override
10    public void start(Stage stage) throws Exception {
11        Parent root = FXMLLoader.load(getClass().
12            | getResource(name:"Painter.fxml"));
13        Scene scene = new Scene(root);
14        stage.setTitle("Painter");
15        stage.setScene(scene);
16        stage.show();
17    }
18    Run | Debug
19    public static void main(String[] args) {
20        | launch(args);
21    }
```

Figure 4.1: Class Painter

4.2 Create Painter.fxml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.geometry.Insets?>
4  <?import javafx.scene.control.Button?>
5  <?import javafx.scene.control.RadioButton?>
6  <?import javafx.scene.control.TitledPane?>
7  <?import javafx.scene.control.ToggleGroup?>
8  <?import javafx.scene.layout.AnchorPane?>
9  <?import javafx.scene.layout.BorderPane?>
10 <?import javafx.scene.layout.Pane?>
11 <?import javafx.scene.layout.VBox?>
12 <?import javafx.scene.text.Font?>
13
14 <BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="480.0" prefWidth="640.0" xmlns="http://javafx.com/javafx/11"
15 |
16 |   <padding>
17 |     <Insets bottom="8.0" left="8.0" right="8.0" top="8.0" />
18 |   </padding>
19 |   <left>
20 |     <VBox maxHeight="1.7976931348623157E308" spacing="8.0" BorderPane.alignment="CENTER">
21 |       <BorderPane.margin>
22 |         <Insets right="8.0" />
23 |       </BorderPane.margin>
24 |       <children>
25 |         <TitledPane animated="false" prefHeight="82.0" prefWidth="74.0" text="Tools">
26 |           <content>
27 |             <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0">
28 |               <children>
29 |                 <RadioButton fx:id="isEraserMode" layoutX="6.0" layoutY="6.0" mnemonicParsing="false" onAction="#penMode" selected="true" text="Pen">
30 |                   <toggleGroup>
31 |                     <ToggleGroup fx:id="identical" />
32 |                   </toggleGroup></RadioButton>
33 |                 <RadioButton fx:id="isEraserMode" layoutX="6.0" layoutY="30.0" mnemonicParsing="false" onAction="#eraserMode" text="Eraser" toggleGroup="$identical">
34 |                   </RadioButton>
35 |                 </children>
36 |             </AnchorPane>
37 |           </content>
38 |         </TitledPane>
39 |         <Button maxWidth="1.7976931348623157E308" mnemonicParsing="false" onAction="#clearButtonPressed" text="Clear">
40 |           <font>
41 |             <Font size="13.0" />
42 |           </font></Button>
43 |       </children>
44 |     </VBox>
45 |   </left>
46 |   <center>
47 |     <Pane fx:id="drawingAreaPane" onMouseDragged="#drawingAreaMouseDragged" prefHeight="200.0" prefWidth="200.0" style="-fx-background-color: white;" BorderPane.alignment="center">
48 |       <BorderPane.margin>
49 |         <Insets />
50 |       </BorderPane.margin></Pane>
51 |   </center>
52 </BorderPane>

```

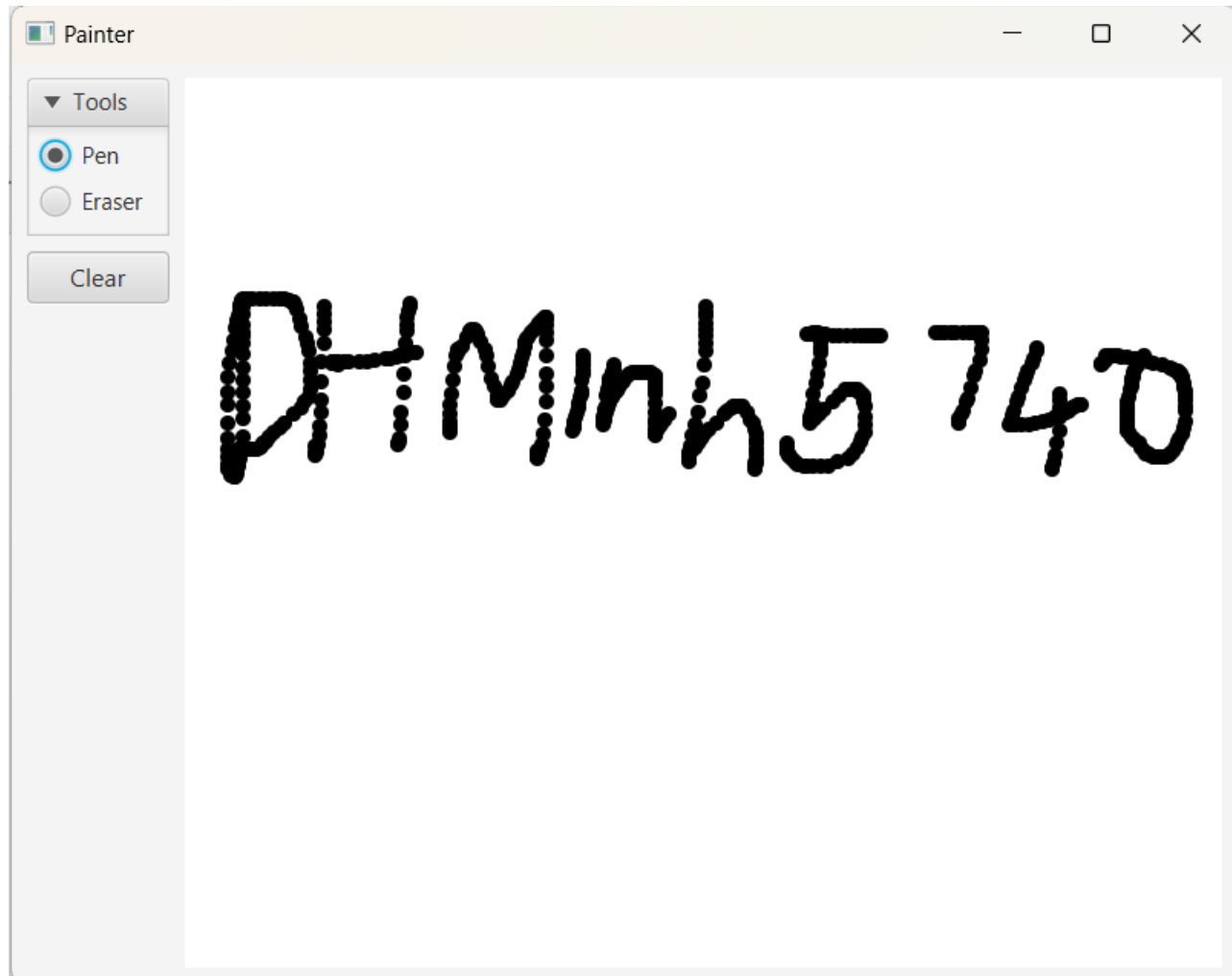
Figure 4.2: Painter.fxml 1

4.3 Create class PainterController

```
1 //Dang Hong Minh - 20225740
2 package GUIProject.hust.soict.hedspi.javafx;
3 import javafx.event.ActionEvent;
4 import javafx.fxml.FXML;
5 import javafx.scene.input.MouseEvent;
6 import javafx.scene.layout.Pane;
7 import javafx.scene.paint.Color;
8 import javafx.scene.shape.Circle;
9 public class PainterController {
10     boolean isEraserMode = false;
11     @FXML
12     private Pane drawingAreaPane;
13     @FXML
14     void clearButtonPressed(ActionEvent event) {
15         drawingAreaPane.getChildren().clear();
16     }
17     @FXML
18     void drawingAreaMouseDragged(MouseEvent event) {
19         //Check if the target is the drawing area
20         if (event.getTarget() == drawingAreaPane) {
21             if (isEraserMode) {
22                 Circle eraser = new Circle(event.getX(), event.getY(), 4, Color.WHITE);
23                 drawingAreaPane.getChildren().add(eraser);
24             } else {
25                 Circle pen = new Circle(event.getX(), event.getY(), 4, Color.BLACK);
26                 drawingAreaPane.getChildren().add(pen);
27             }
28         }
29     }
30     @FXML
31     void penMode(ActionEvent event) {
32         isEraserMode = false;
33     }
34     @FXML
35     void eraserMode (ActionEvent event) {
36         isEraserMode = true;
37     }
38 }
```

Figure 4.4: PainterController

Figur



e 4.5: Use Pen

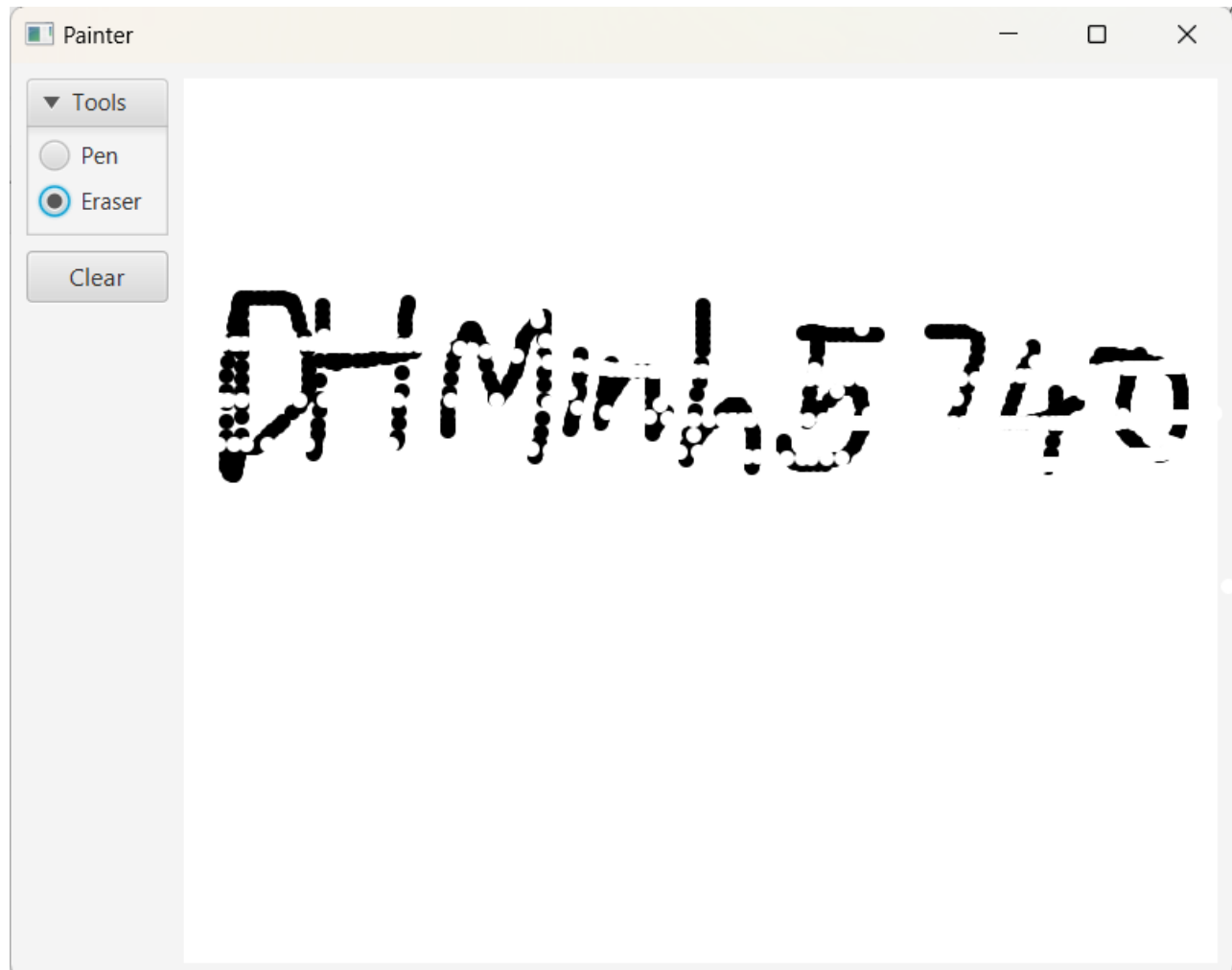


Figure 4.6: Use Eraser

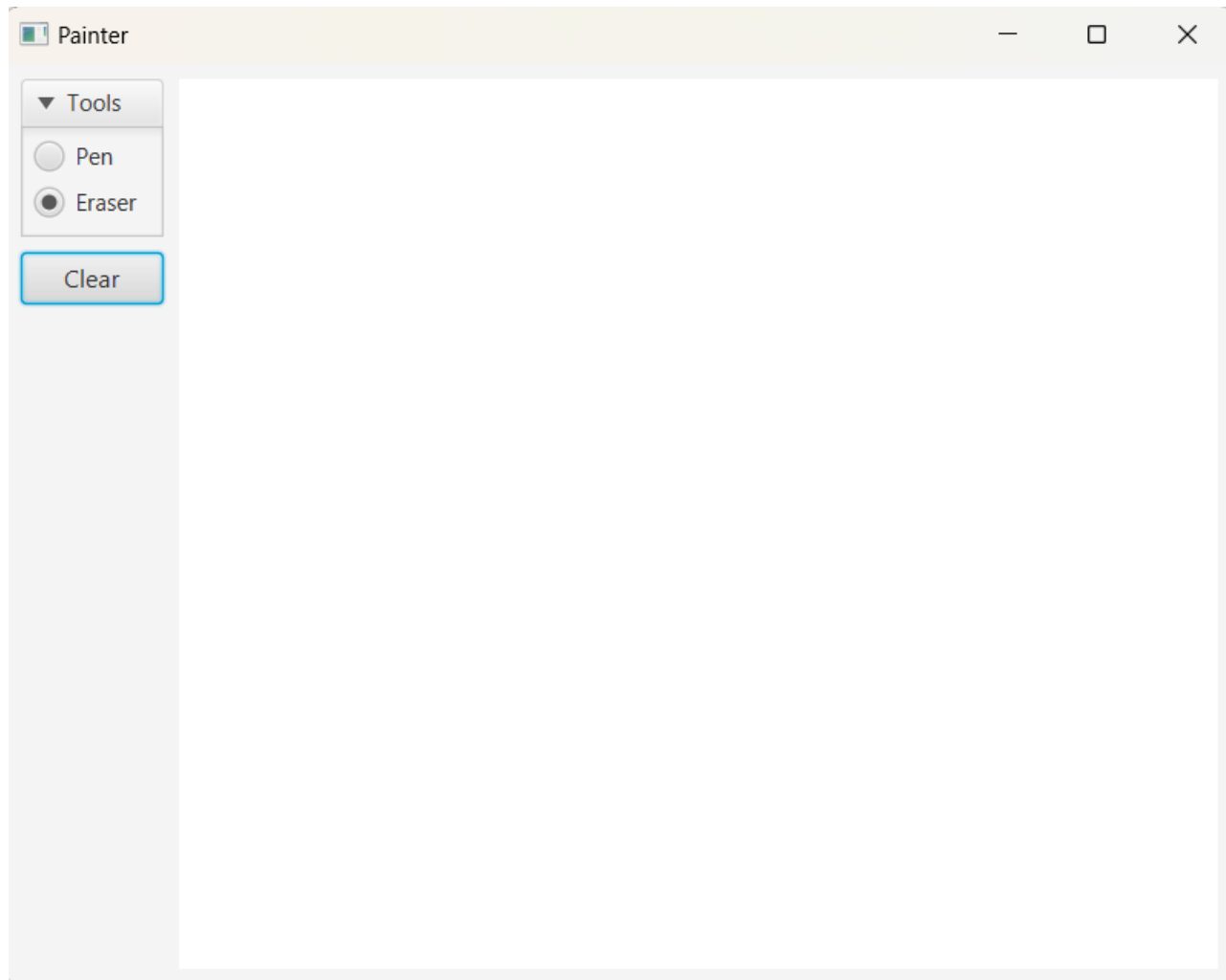


Figure 4.7: Clear button

5 View Cart Screen

5.1 Create cart.fxml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <?import javafx.geometry.Insets?>
4 <?import javafx.scene.control.Button?>
5 <?import javafx.scene.control.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="1.7976931348623157E308" maxWidth="1.7976931348623157E308" minHeight="-Infinity" minWidth="-Infinity" prefHeight="768.0" prefWidth="1024.0" xmlns="http://ja
20 <top>
21 <VBox BorderPane.alignment="CENTER">
22 <children>
23 <MenuBar>
24 <menus>
25 <Menu mnemonicParsing="false" text="Options">
26 <items>
27 <Menu mnemonicParsing="false" text="Update Store">
28 <items>
29 <MenuItem mnemonicParsing="false" onAction="#addBookToStore" text="Add Book" />
30 <MenuItem mnemonicParsing="false" onAction="#addCDToStore" text="Add CD" />
31 <MenuItem mnemonicParsing="false" onAction="#addDVDToStore" text="Add DVD" />
32 </items>
33 </Menu>
34 <MenuItem mnemonicParsing="false" onAction="#viewStore" text="View Store" />
35 <MenuItem mnemonicParsing="false" text="View Cart" />
36 </items>
37 </Menu>
38 </menus>
39 </MenuBar>
40 <Label text="CART:~$ MinhDH5740" textFill="AQUA">
41 <padding>
42 <Insets left="10.0" />
43 </padding>
44 <font>
45 <font-size="50.0" />

```

Figure 5.1: Cart.fxml 1

```

46 </font>
47 </Label>
48 </children>
49 </VBox>
50 </top>
51 <center>
52 <VBox BorderPane.alignment="CENTER">
53 <padding>
54 <Insets left="10.0" />
55 </padding>
56 <children>
57 <HBox alignment="CENTER_LEFT" prefWidth="200.0" spacing="10.0">
58 <padding>
59 <Insets bottom="10.0" top="10.0" />
60 </padding>
61 <children>
62 <Label text="Filter:" />
63 <TextField fx:id="tfFilter" />
64 <RadioButton fx:id="radioBtnFilterID" mnemonicParsing="false" onAction="#setFilterByID" selected="true" text="By ID">
65 <toggleGroup>
66 <ToggleGroup fx:id="filterCategory" />
67 </toggleGroup>
68 </RadioButton>
69 <RadioButton fx:id="radioBtnFilterTitle" mnemonicParsing="false" onAction="#setFilterByTitle" text="By Title" toggleGroup="filterCategory" />
70 </children>
71 </HBox>
72 <TableView fx:id="tblMedia">
73 <columns>
74 <TableColumn fx:id="colMediaTitle" prefWidth="75.0" text="Title" />
75 <TableColumn fx:id="colMediaCategory" prefWidth="75.0" text="Category" />
76 <TableColumn fx:id="colMediaCost" prefWidth="75.0" text="Cost" />
77 </columns>
78 <columnResizePolicy>
79 <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
80 </columnResizePolicy>
81 </TableView>
82 <HBox alignment="TOP_RIGHT" prefWidth="200.0" spacing="10.0">
83 <children>
84 <HBox alignment="CENTER_LEFT" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" spacing="10.0">
85 <children>

```

Figure 5.2: Cart.fxml 2

```

86         <Button mnemonicParsing="false" onAction="#sortBtnPressed" text="Sort" />
87         <RadioButton mnemonicParsing="false" onAction="#setSortByTitle" selected="true" text="By Title">
88             <toggleGroup>
89                 <ToggleGroup fx:id="sortCategory" />
90             </toggleGroup>
91         </RadioButton>
92         <RadioButton mnemonicParsing="false" onAction="#setSortByCost" text="By Cost" toggleGroup="$sortCategory" />
93     </children>
94 </HBox>
95 <HBox alignment="CENTER_RIGHT" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" spacing="10.0">
96     <children>
97         <Button fx:id="btnPlay" mnemonicParsing="false" onAction="#playButtonPressed" text="Play" />
98         <Button fx:id="btnRemove" mnemonicParsing="false" onAction="#removeButtonPressed" text="Remove" />
99         <Button fx:id="btnDetails" mnemonicParsing="false" onAction="#detailsButtonPressed" text="Details" />
100     </children>
101 </HBox>
102 </children>
103 <padding>
104     <Insets bottom="40.0" top="10.0" />
105 </padding>
106 </HBox>
107 </children>
108 </VBox>
109 </center>
110 <right>
111     <VBox alignment="TOP_CENTER" prefHeight="200.0" spacing="10.0" BorderPane.alignment="CENTER">
112         <padding>
113             <Insets top="50.0" />
114         </padding>
115         <children>
116             <HBox alignment="CENTER" spacing="10.0">
117                 <children>
118                     <Label text="Total">
119                         <font>
120                             <Font size="24.0" />
121                         </font>
122                     </Label>
123                     <Label fx:id="costLabel" text="0 $" textFill="AQUA">
124                         <font>
125                             <Font size="24.0" />
126                         </font>
127                     </Label>
128                 </children>
129             </HBox>
130             <Button mnemonicParsing="false" onAction="#placeOrderPressed" style="-fx-background-color: RED;" text="Place Order" textFill="WHITE">
131                 <font>
132                     <Font size="24.0" />
133                 </font>
134             </Button>
135         </children>
136     </VBox>
137 </right>
138 </BorderPane>

```

Figure 5.3: Cart.fxml 3

5.2 Create class CartScreen


```

1 //Dang Hong Minh - 20225740
2 package AimsProject.hust.soict.hedspi.aims.screen;
3 import java.awt.event.*;
4 import java.io.IOException;
5 import javax.swing.JFrame;
6 import AimsProject.hust.soict.hedspi.aims.cart.Cart;
7 import AimsProject.hust.soict.hedspi.aims.media.*;
8 import AimsProject.hust.soict.hedspi.aims.store.Store;
9 import javafx.application.Platform;
10 import javafx.embed.swing.JFXPanel;
11 import javafx.fxml.FXMLLoader;
12 import javafx.scene.Parent;
13 import javafx.scene.Scene;
14 public class CartScreen extends JFrame {
15     public CartScreen(Store store, Cart cart) {
16         super();
17         this.setSize(width:1024,height:768);
18         JFXPanel fxPanel = new JFXPanel();
19         this.add(fxPanel);
20         this.setTitle(title:"Cart");
21         this.setVisible(b:true);
22         JFrame frame = this;
23         this.addWindowListener(new WindowAdapter() {
24             @Override
25             public void windowClosing(WindowEvent e) {
26                 new StoreScreen(store, cart);
27                 dispose();
28             }
29         });
30         Platform.runLater(new Runnable() {
31             @Override
32             public void run() {
33                 try {
34                     FXMLLoader loader = new FXMLLoader(
35                         | | getClass().getResource(name:"cart.fxml"));
36                     CartScreenController controller = new CartScreenController(store, cart, frame);
37                     loader.setController(controller);
38                     Parent root = loader.load();
39                     fxPanel.setScene(new Scene(root));
40                 } catch (IOException e) {
41                     e.printStackTrace();
42                 }
43             }
44         });
45     }
46     public static void main(String args[]) throws Exception {
47         // Test
48         DigitalVideoDisc dvd1 = new DigitalVideoDisc(title:"The Goblin King", category:"Animation", director:"Someone", length:87, cost:19.95f);
49         DigitalVideoDisc dvd2 = new DigitalVideoDisc(title:"Star Wars", category:"Sci-fi", director:"George Lucas", length:87, cost:24.95f);
50         DigitalVideoDisc dvd3 = new DigitalVideoDisc(title:"Aladin", category:"Animation", cost:18.99f);
51         Track track1 = new Track(title:"Mondstadt", length:10);
52         Track track2 = new Track(title:"Inazuma", length:21);
53         Track track3 = new Track(title:"Natlan");
54         CompactDisc cd1 = new CompactDisc(title:"Bisoku SS1", category:"Animation", artist:"Various", cost:30.95f);
55         cd1.addTrack(track1);
56         cd1.addTrack(track2);
57         CompactDisc cd2 = new CompactDisc(title:"Bisoku SS2", category:"Animation", artist:"Various", cost:25.99f);
58         cd2.addTrack(track3);
59         Cart myCart = new Cart();
60         myCart.addMedia(dvd1);
61         myCart.addMedia(dvd2);
62         myCart.addMedia(dvd3);
63         myCart.addMedia(cd1);
64         myCart.addMedia(cd2);
65         Store myStore = new Store();
66         new CartScreen(myStore, myCart);
67     }
68 }

```

Figure 5.4: CartScreen class

5.3 Create class CartScreenController

```

1 //Dang Hong Minh - 20225740
2 package AimsProject.hust.soict.hedspi.aims.screen;
3 import AimsProject.hust.soict.hedspi.aims.cart.Cart;
4 import AimsProject.hust.soict.hedspi.aims.media.*;
5 import AimsProject.hust.soict.hedspi.aims.store.Store;
6 import AimsProject.hust.soict.hedspi.aims.exception.*;
7 import javax.swing.JFrame;
8 import javafx.beans.value.*;
9 import javafx.event.ActionEvent;
10 import javafx.fxml.FXML;
11 import javafx.scene.control.Alert;
12 import javafx.scene.control.Alert.AlertType;
13 import javafx.scene.control.Button;
14 import javafx.scene.control.Label;
15 import javafx.scene.control.TableColumn;
16 import javafx.scene.control.TableView;
17 import javafx.scene.control.TextField;
18 import javafx.scene.control.cell.PropertyValueFactory;
19 import javafx.collections.transformation.FilteredList;
20 public class CartScreenController {
21     private Store store;
22     private Cart cart;
23     private boolean filterByID = true;
24     private boolean sortByTitle = true;
25     private FilteredList<Media> filteredCart;
26     private JFrame stage;
27     @FXML
28     private TableView<Media> tblMedia;
29     @FXML
30     private TableColumn<Media, String> colMediaTitle;
31     @FXML
32     private TableColumn<Media, String> colMediaCategory;
33     @FXML
34     private TableColumn<Media, String> colMediaCost;
35     @FXML
36     private Button btnPlay;
37     @FXML
38     private Button btnRemove;
39     @FXML
40     private Button btnDetails;
41     @FXML
42     private TextField ttfFilter;
43     @FXML
44     private Label costLabel;
45     public CartScreenController(Store store, Cart cart, JFrame stage) {
46         super();
47         this.store = store;
48         this.cart = cart;
49         this.stage = stage;
50     }
51     @FXML
52     public void initialize() {
53         filteredCart = new FilteredList<Media>(this.cart.getItemsOrdered(), s -> true);
54         colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"title"));
55         colMediaCategory.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"category"));

```

Figure 5.5: CartScreenController 1

```

56         colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"cost"));
57         tblMedia.setItems(filteredCart);
58         btnPlay.setVisible(false);
59         btnRemove.setVisible(false);
60         btnDetails.setVisible(false);
61         costLabel.setText(String.valueOf(this.cart.totalCost()));
62         tblMedia.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Media>() {
63             @Override
64             public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {
65                 updateButtonBar(newValue);
66             }
67         });
68         tfFilter.textProperty().addListener(new ChangeListener<String> () {
69             @Override
70             public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
71                 showFilteredMedia(newValue);
72             }
73         });
74     }
75     private void updateButtonBar(Media media) {
76         if (media == null) {
77             btnRemove.setVisible(false);
78             btnDetails.setVisible(false);
79             btnPlay.setVisible(false);
80         } else {
81             btnRemove.setVisible(true);
82             btnDetails.setVisible(true);
83             if (media instanceof Playable) {
84                 btnPlay.setVisible(true);
85             } else {
86                 btnPlay.setVisible(false);
87             }
88         }
89     }
90     private void showFilteredMedia(String filter) {
91         if (filter == null || filter.length() == 0) {
92             filteredCart.setPredicate(s -> true);
93         } else {
94             if (filterByID) {
95                 try {
96                     filteredCart.setPredicate(s -> s.getId() == Integer.parseInt(filter));
97                 } catch (NumberFormatException e) {}
98             } else {
99                 filteredCart.setPredicate(s -> s.getTitle().toLowerCase().contains(filter));
100             }
101         }
102     }
103     @FXML
104     private void removeButtonPressed(ActionEvent event) {
105         Media media = tblMedia.getSelectionModel().getSelectedItem();
106         try {
107             this.cart.removeMedia(media);
108         } catch (NonExistingItemException e) {
109             Alert alert = new Alert(AlertType.ERROR);
110             alert.setTitle(title:"Notification");

```

Figure 5.6: CartScreenController 2

```
111         alert.setHeaderText(headerText:"Failed to remove");
112         alert.setContentText(contentText:"Media not in cart");
113         alert.showAndWait();
114     }
115     costLabel.setText(String.valueOf(this.cart.totalCost()));
116 }
117 @FXML
118 private void playButtonPressed(ActionEvent event) {
119     Media media = this.tblMedia.getSelectionModel().getSelectedItem();
120     try {
121         ((Playable)media).play();
122     } catch (PlayerException e) {
123         Alert alert = new Alert(AlertType.WARNING);
124         alert.setTitle(title:"Media Player");
125         alert.setHeaderText(headerText:"ERROR: Media length is non-positive.");
126         alert.setContentText(contentText:"Media cannot be played.");
127         alert.showAndWait();
128     }
129 }
130 @FXML
131 private void detailsButtonPressed(ActionEvent event) {
132     Media media = this.tblMedia.getSelectionModel().getSelectedItem();
133     Alert alert = new Alert(AlertType.INFORMATION);
134     alert.setTitle(title:"Detail infomation");
135     alert.setHeaderText("Viewing " + media.getTitle() + " detail infomation.");
136     alert.setContentText(media.getDetails());
137     alert.showAndWait();
138 }
139 @FXML
140 private void placeOrderPressed(ActionEvent event) {
141     if (this.cart.getSize() > 0) {
142         Alert alert = new Alert(AlertType.INFORMATION);
143         alert.setTitle(title:"Notification");
144         alert.setHeaderText(headerText:"Success!");
145         alert.setContentText(contentText:"Your order has been placed.");
146         alert.showAndWait();
147         this.cart.empty();
148         costLabel.setText(String.valueOf(this.cart.totalCost()));
149     } else {
150         Alert alert = new Alert(AlertType.ERROR);
151         alert.setTitle(title:"Notification");
152         alert.setHeaderText(headerText:"ERROR: Failed to place order.");
153         alert.setContentText(contentText:"Your cart is empty");
154         alert.showAndWait();
155     }
156 }
157 @FXML
158 private void setFilterByID() {
159     this.filterByID = true;
160 }
161 @FXML
162 private void setFilterByTitle() {
163     this.filterByID = false;
164 }
165 @FXML
```

```
166     private void sortBtnPressed() {
167         if (sortByTitle) {
168             this.cart.sortByTitle();
169         } else {
170             this.cart.sortByCost();
171         }
172     }
173     @FXML
174     private void setSortByTitle() {
175         this.sortByTitle = true;
176     }
177     @FXML
178     private void setSortByCost() {
179         this.sortByTitle = false;
180     }
181     @FXML
182     private void viewStore() {
183         new StoreScreen(store, cart);
184         stage.setVisible(b:false);
185     }
186     @FXML
187     private void addDVDToStore() {
188         new AddDVDToStoreScreen(store, cart);
189         stage.setVisible(b:false);
190     }
191     @FXML
192     private void addBookToStore() {
193         new AddBookToStoreScreen(store, cart);
194         stage.setVisible(b:false);
195     }
196     @FXML
197     private void addCDToStore() {
198         new AddCDToStoreScreen(store, cart);
199         stage.setVisible(b:false);
200     }
201 }
```

5.4 Demo

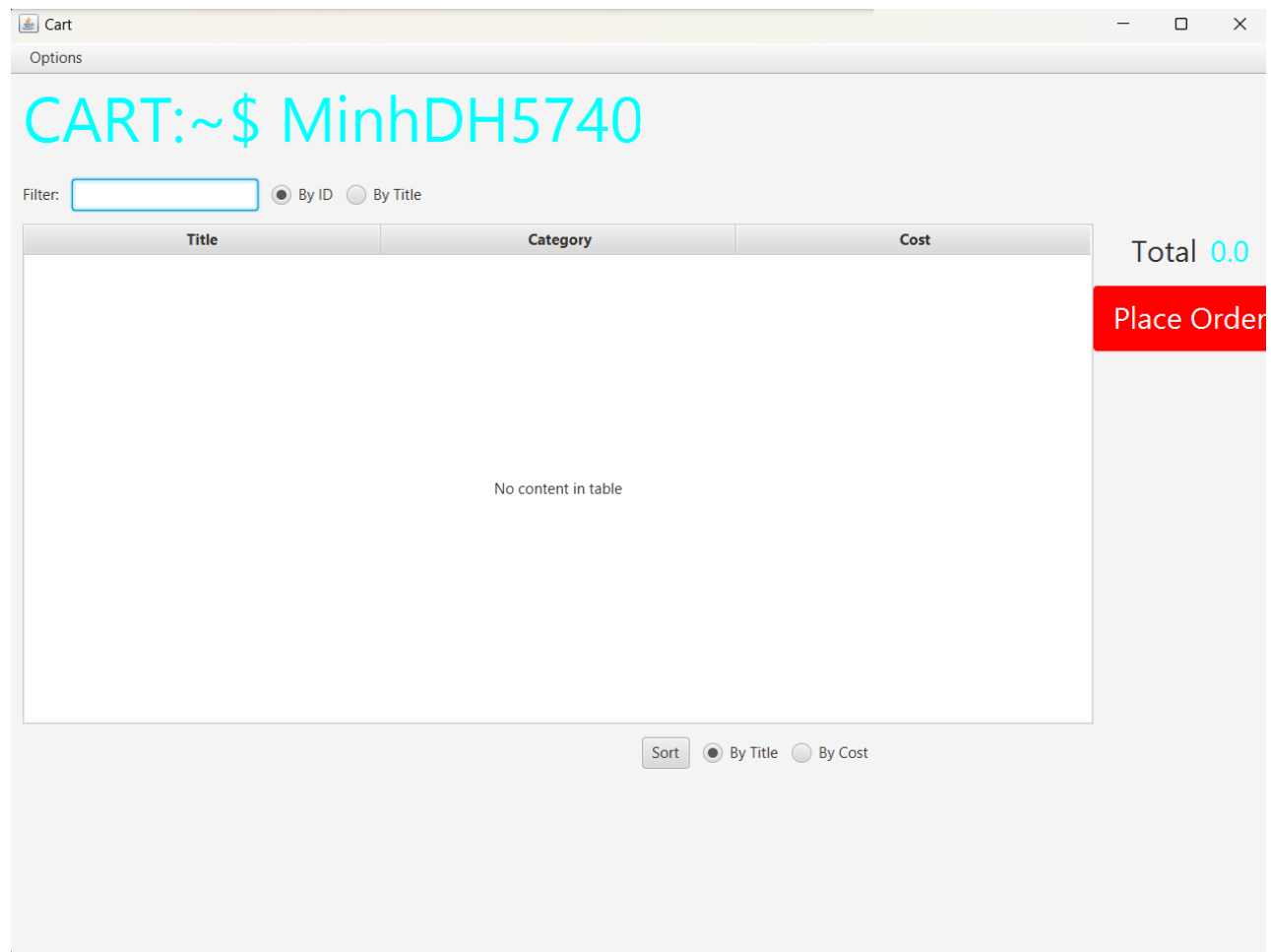


Figure 5.7: Demo CartScreen

6 Updating buttons based on selected item in TableView – ChangeListener

```

51  @FXML
52  public void initialize() {
53      filteredCart = new FilteredList<Media>(this.cart.getItemsOrdered(), s -> true);
54      colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"title"));
55      colMediaCategory.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"category"));
56      colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, String>(property:"cost"));
57      tblMedia.setItems(filteredCart);
58      btnPlay.setVisible(false);
59      btnRemove.setVisible(false);
60      btnDetails.setVisible(false);
61      costLabel.setText(String.valueOf(this.cart.totalCost()));
62      tblMedia.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Media>() {
63          @Override
64          public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {
65              updateButtonBar(newValue);
66          }
67      });
68      tfFilter.textProperty().addListener(new ChangeListener<String> () {
69          @Override
70          public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
71              showFilteredMedia(newValue);
72          }
73      });
74  }

```

Figure 6.1: CartScreenController 1

```

75  private void updateButtonBar(Media media) {
76      if (media == null) {
77          btnRemove.setVisible(false);
78          btnDetails.setVisible(false);
79          btnPlay.setVisible(false);
80      } else {
81          btnRemove.setVisible(true);
82          btnDetails.setVisible(true);
83          if (media instanceof Playable) {
84              btnPlay.setVisible(true);
85          } else {
86              btnPlay.setVisible(false);
87          }
88      }
89  }

```

Figure 6.2: CartScreenController 2

7 Deleting a media

7.1 Code

```

103     @FXML
104     private void removeButtonPressed(ActionEvent event) {
105         Media media = tblMedia.getSelectionModel().getSelectedItem();
106         try {
107             this.cart.removeMedia(media);
108         } catch (NonExistingItemException e) {
109             Alert alert = new Alert(AlertType.ERROR);
110             alert.setTitle(title: "Notification");
111             alert.setHeaderText(headerText: "Failed to remove");
112             alert.setContentText(contentText: "Media not in cart");
113             alert.showAndWait();
114         }
115         costLabel.setText(String.valueOf(this.cart.totalCost()));
116     }

```

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

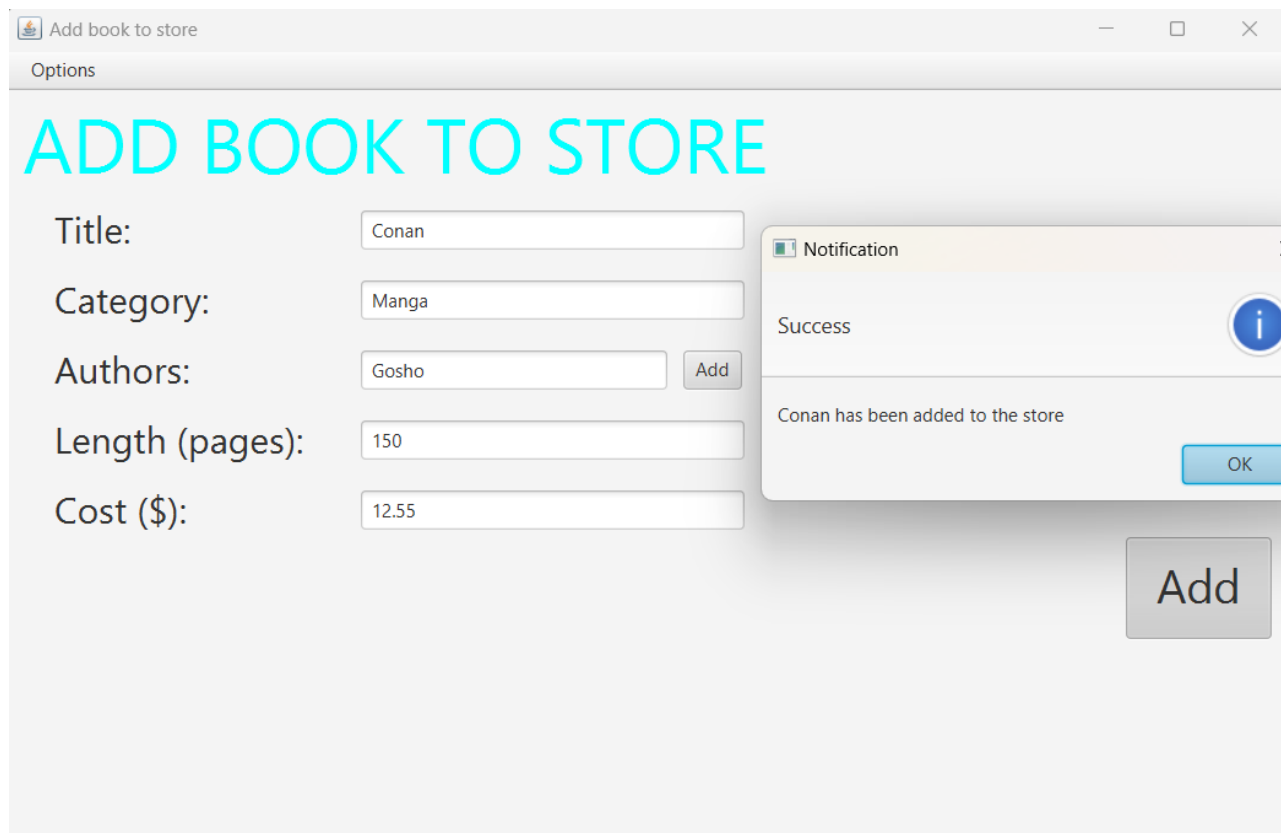


Figure 8.2: Add book



Figure 8.3: Store after add book

Options

ADD CD TO STORE

Title:

Category:

Artist:

Tracks:

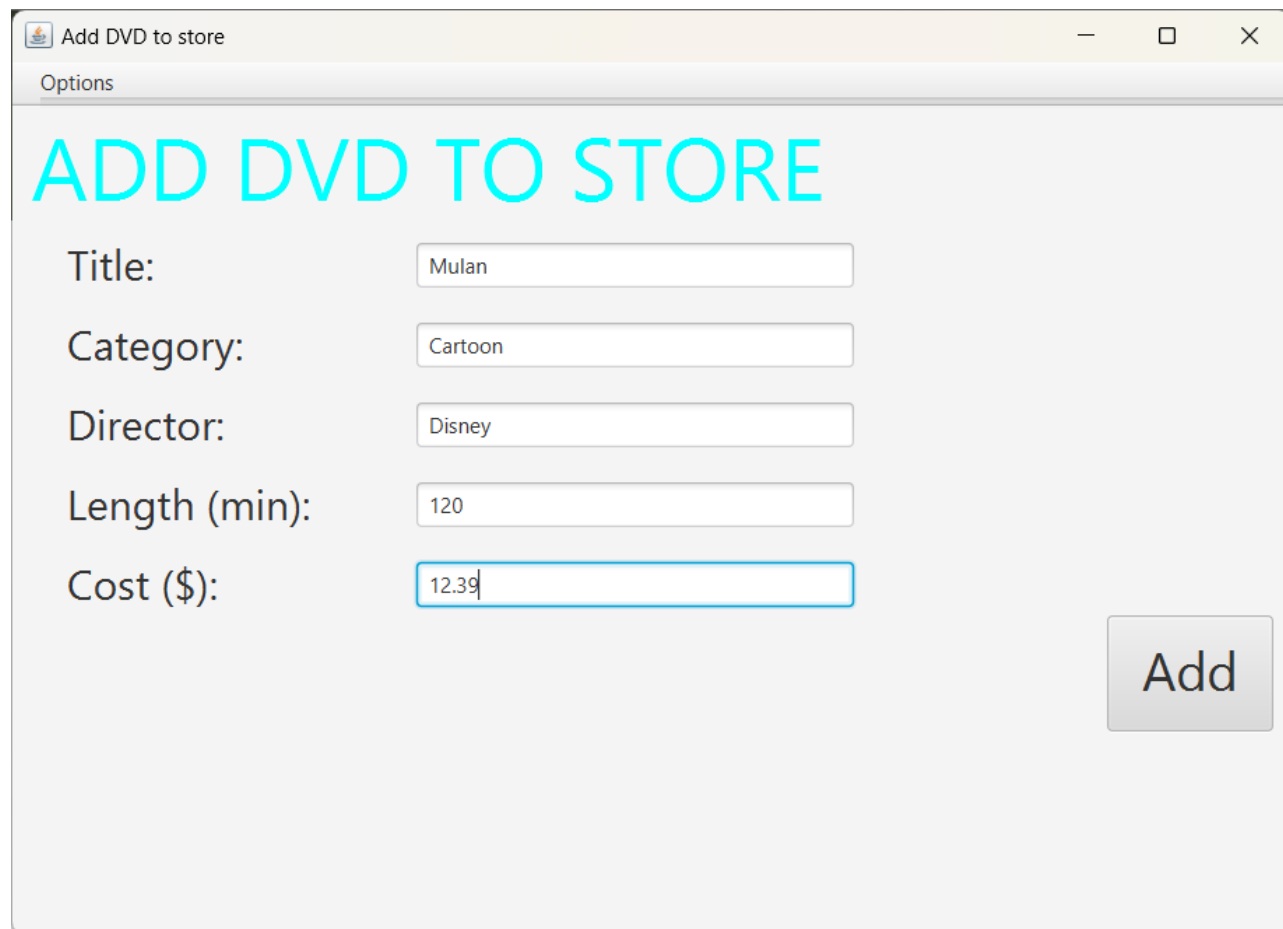
Director:

Cost (\$):

Figure 8.4: Add CD



Figure 8.5: Store after add CD



Options

ADD DVD TO STORE

Title:

Category:

Director:

Length (min):

Cost (\$):

Add

Figure 8.6 Add DVD



Figure 8.7: Store after add DVD

9 Use case Diagram

