

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

\*\*\*\*\*



**BÁO CÁO THỰC HÀNH**  
**IT3103-744529-2024.1**  
**BÀI THỰC HÀNH 5**

**Họ và tên sinh viên: Đoàn Thanh Tùng**  
**MSSV: 20225946**  
**Lớp: Việt Nhật 02 – K67**  
**GVHD: Lê Thị Hoa**  
**HTGD: Bùi Trọng Dũng**

## 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 3.2.1: Class StoreScreen 2 .....	10
Figure 3.2.2: Class StoreScreen 2 .....	10
Figure 3.3: Class StoreScreen 3 .....	10
Figure 3.4: Class StoreScreen 4 .....	11
Figure 3.5: Class MediaStore 1.....	13
Figure 3.6: Class MediaStore 2.....	13
Figure 3.7: Class MediaStore 3.....	14
Figure 3.8: StoreScreen .....	14
Figure 3.9 Demo Add to cart button .....	15
Figure 3.10 Demo Play button.....	15
Figure 3.11 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: after 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
Figure 9.1: Usecase Diagram .....	30
Figure 10.1: Class Diagram .....	30

## 1. Swing components

### 1.1 AWTAccumulator

```
//DOAN THANH TUNG - 20225946
public class AWTAccumulator extends Frame {
    private TextField tfInput;
    private TextField tfOutput;
    private int sum = 0;

    public AWTAccumulator() {
        setLayout(new GridLayout(2, 2));
        add(new Label("20225946 || Enter an Interger: "));
        tfInput = new TextField(10);
        add(tfInput);
        tfInput.addActionListener(new TFInputListener());
        add(new Label("20225946 || The Accumulated Sum is: "));
        tfOutput = new TextField(10);
        tfOutput.setEditable(false);
        add(tfOutput);

        setTitle("AWT Accumulator");
        setSize(350, 120);
        setVisible(true);
    }

    public static void main(String arg[]) {
        new AWTAccumulator();
    }

    private class TFInputListener implements ActionListener {
        public void actionPerformed(ActionEvent evt) {
            int numberIn = Integer.parseInt(tfInput.getText());
            sum += numberIn;
            tfInput.setText("");
            tfOutput.setText(sum + "");
        }
    }
}
```

Figure 1.1: Source code of AWTAccumulator

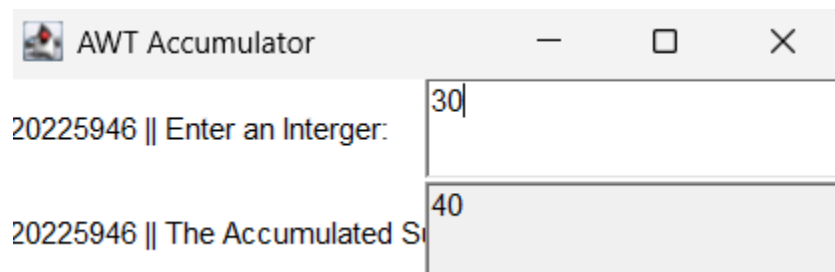


Figure 1.2: Demo of AWTAccumulator

## 1.2 SwingAccumulator

```
//DOAN THANH TUNG - 20225946
public class SwingAccumulator extends JFrame {
    private JTextField tfInput;
    private JTextField tfOutput;
    private int sum = 0;

    public SwingAccumulator() {
        Container cp = getContentPane();
        cp.setLayout(new GridLayout(2,2));
        cp.add(new JLabel("20225946 || Enter an Integer: "));
        tfInput = new JTextField(10);
        cp.add(tfInput);
        tfInput.addActionListener(new TFInputListener());
        cp.add(new JLabel("20225946 || The Accumulated Sum is: "));
        tfOutput = new JTextField(10);
        tfOutput.setEditable(false);
        cp.add(tfOutput);
        setTitle("Swing Accumulator");
        setSize(350, 120);
        setVisible(true);
    }

    public static void main(String arg[]) {
        new SwingAccumulator();
    }

    private class TFInputListener implements ActionListener {
        public void actionPerformed(ActionEvent evt) {
            int numberIn = Integer.parseInt(tfInput.getText());
            sum+= numberIn;
            tfInput.setText("");
            tfOutput.setText(sum+"");
        }
    }
}
```

Figure 1.3: Source code of SwingAccumulator

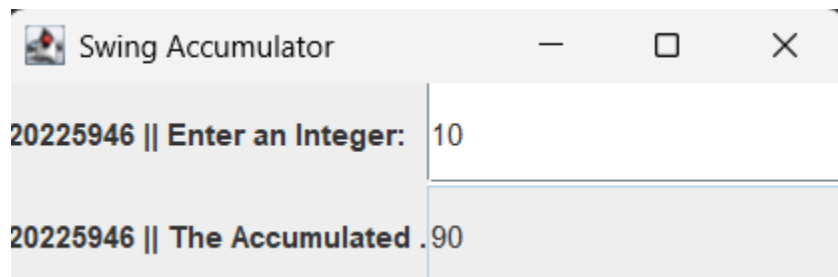


Figure 1.4: Demo of SwingAccumulator

## 2 Organizing Swing components with Layout Managers

### 2.1 Code

```
//DOAN THANH TUNG - 20225946
public class NumberGrid extends JFrame {
    private JButton[] btnNumbers = new JButton[10];
    private JButton btnDelete, btnReset;
    private JTextField tfDisplay;

    public NumberGrid() {
        tfDisplay = new JTextField();
        tfDisplay.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
        JPanel panelButtons = new JPanel(new GridLayout(4, 3));
        addButtons(panelButtons);
        Container cp = getContentPane();
        cp.setLayout(new BorderLayout());
        cp.add(tfDisplay, BorderLayout.NORTH);
        cp.add(panelButtons, BorderLayout.CENTER);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle("Number Grid");
        setSize(200, 200);
        setVisible(true);
    }
}
```

Figure 2.1: Source code of NumberGrid 1

```
//DOAN THANH TUNG - 20225946
void addButtons(JPanel panelButtons) {
    ButtonListener btnListener = new ButtonListener();
    for(int i = 1; i<=9;i++) {
        btnNumbers[i]= new JButton(""+i);
        panelButtons.add(btnNumbers[i]);
        btnNumbers[i].addActionListener(btnListener);
    }
    btnDelete = new JButton("DEL");
    panelButtons.add(btnDelete);
    btnDelete.addActionListener(btnListener);
    btnNumbers[0] = new JButton("0");
    panelButtons.add(btnNumbers[0]);
    btnNumbers[0].addActionListener(btnListener);
    btnReset = new JButton("C");
    panelButtons.add(btnReset);
    btnReset.addActionListener(btnListener);
}
```

Figure 2.2: Source code of NumberGrid 2



## 2.2 Demo

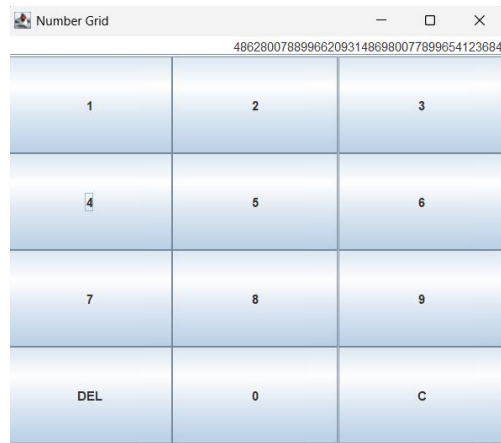


Figure 2.3: Demo buttons 0-9

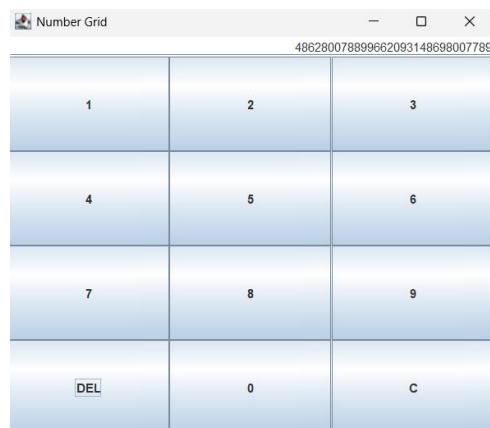


Figure 2.4: Demo DEL button



Figure 2.5: Demo C button

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

#### 3.1 Create class StoreScreen

```
//DOAN THANH TUNG - 20225946  
public class StoreScreen {  
    private Store store;  
      
}
```

Figure 3.1: Class StoreScreen 1

```
//DOAN THANH TUNG - 20225946
JPanel createNorth() {
    JPanel north = new JPanel();
    north.setLayout(new BorderLayout(north, BorderLayout.Y_AXIS));
    north.add(createMenuBar());
    north.add(createHeader());
    return north;
}

JMenuBar createMenuBar() {
    JMenu menu = new JMenu("Options");

    JMenu smUpdateStore = new JMenu("Update Store");
    JMenuItem addBook = new JMenuItem("Add Book");
    JMenuItem addCD = new JMenuItem("Add CD");
    JMenuItem addDVD = new JMenuItem("Add DVD");
    menu.add(smUpdateStore);
    menu.add(new JMenuItem("View store"));
    JMenuItem cart = new JMenuItem("View cart");
    JMenuBar menuBar = new JMenuBar();
    menuBar.setLayout(new FlowLayout(FlowLayout.LEFT));
    menuBar.add(menu);

    return menuBar;
}
```

Figure 3.2.1: Class StoreScreen 2

```
// DOAN THANH TUNG - 20225946
JPanel createHeader() {
    JPanel header = new JPanel();
    header.setLayout(new BorderLayout(header, BorderLayout.X_AXIS));
    JLabel title = new JLabel("AIMS");
    title.setFont(new Font(title.getFont().getName(), Font.PLAIN, 50));
    title.setForeground(Color.CYAN);
    JButton cart = new JButton("View cart");
    cart.setPreferredSize(new Dimension(100, 50));
    cart.setMaximumSize(new Dimension(100, 50));
    header.add(Box.createRigidArea(new Dimension(10, 10)));
    header.add(title);
    header.add(Box.createHorizontalGlue());
    header.add(cart);
    header.add(Box.createRigidArea(new Dimension(10, 10)));

    return header;
}
```

Figure 3.2.2: Class StoreScreen 2

```
// DOAN THANH TUNG - 20225946
JPanel createCenter() {
    JPanel center = new JPanel();
    center.setLayout(new GridLayout(3, 3, 2, 2));
    ArrayList<Media> mediaInStore = store.ItemsInStore();
    for (Media media : mediaInStore) {
        MediaStore cell = new MediaStore(media);
        center.add(cell);
    }
    return center;
}
```

Figure 3.3: Class StoreScreen 3

```
// DOAN THANH TUNG - 20225946|  
public StoreScreen(Store store) {  
    this.store = store;  
    Container cp = getContentPane();  
    cp.setLayout(new BorderLayout());  
    cp.add(createNorth(), BorderLayout.NORTH);  
    cp.add(createCenter(), BorderLayout.CENTER);  
    setVisible(true);  
    setTitle("Store");  
    setSize(1024, 768);  
}
```

Figure 3.4: Class StoreScreen 4

### 3.2 Create class MediaStore

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

import java.awt.Color;
//DOAN THANH TUNG - 202225946
public class MediaStore extends JPanel {
    private Media media;
    private Cart cart;

    public MediaStore(Media media, Cart cart) {
        this.media = media;
        this.cart = cart;
        this.setLayout(new BorderLayout(this, BorderLayout.Y_AXIS));
        JLabel title = new JLabel(media.getTitle());
        title.setFont(new Font(title.getFont().getName(), Font.PLAIN, 20));
        title.setAlignmentX(CENTER_ALIGNMENT);

        JLabel cost = new JLabel("" + media.getCost() + " $");
        cost.setAlignmentX(CENTER_ALIGNMENT);

        JPanel container = new JPanel();
        container.setLayout(new FlowLayout(FlowLayout.CENTER));

        JButton addToCartButton = new JButton("Add to cart");
        addToCartButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                addMedia();
            }
        });
        container.add(addToCartButton);
```

Figure 3.5: Class MediaStore 1

```
        container.add(addToCartButton);
        if (media instanceof Playable) {
            JButton playButton = new JButton("Play");
            playButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    playMedia();
                }
            });
            container.add(playButton);
        }
        this.add(Box.createVerticalGlue());
        this.add(title);
        this.add(cost);
        this.add(Box.createVerticalGlue());
        this.add(container);
        this.setBorder(BorderFactory.createLineBorder(Color.BLACK));
    }

    private void playMedia() {
        try {
            if (media instanceof CompactDisc) {
                ((CompactDisc) media).play();
            } else if (media instanceof DigitalVideoDisc) {
                ((DigitalVideoDisc) media).play();
            }
        } catch (UserException e) {
            JOptionPane.showMessageDialog(null, "Error: " + e.getMessage(), "Playback Error",
                JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

Figure 3.6: Class MediaStore 2

```
private void playMedia() {
    try {
        if (media instanceof CompactDisc) {
            ((CompactDisc) media).play();
        } else if (media instanceof DigitalVideoDisc) {
            ((DigitalVideoDisc) media).play();
        }
    } catch (UserException e) {
        JOptionPane.showMessageDialog(null, "Error: " + e.getMessage(), "Playback Error",
            JOptionPane.ERROR_MESSAGE);
    }
}

private void addMedia() {
    if (cart.getLength() < Cart.MAX_NUMBERS_ORDERED) {
        cart.addMedia(media);
        JOptionPane.showMessageDialog(null, media.getTitle() + " has been added to your cart.",
            "Cart Update",
            JOptionPane.INFORMATION_MESSAGE);
    } else {
        JOptionPane.showMessageDialog(null, "The cart is full. You cannot add more items.",
            "Cart Full",
            JOptionPane.WARNING_MESSAGE);
    }
}
}
```

Figure 3.7: Class MediaStore 3

### 3.3 Demo

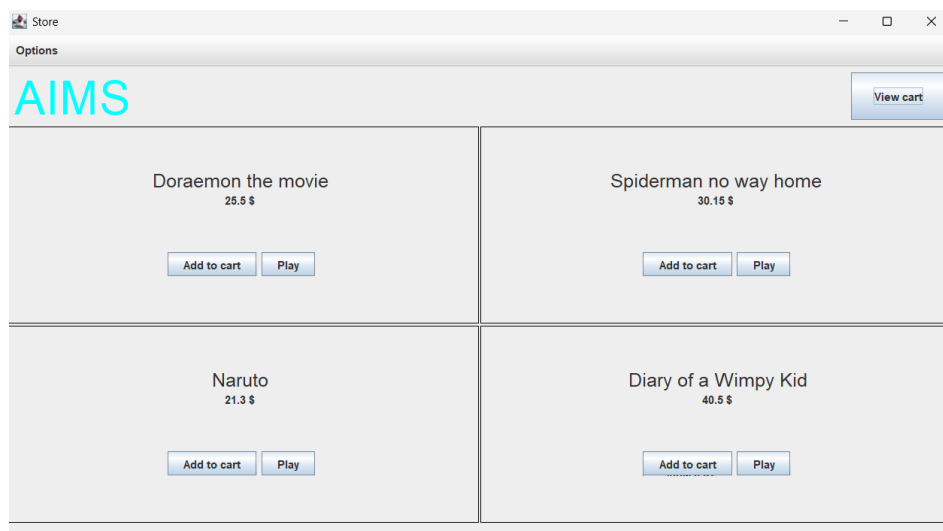


Figure 3.8: StoreScreen

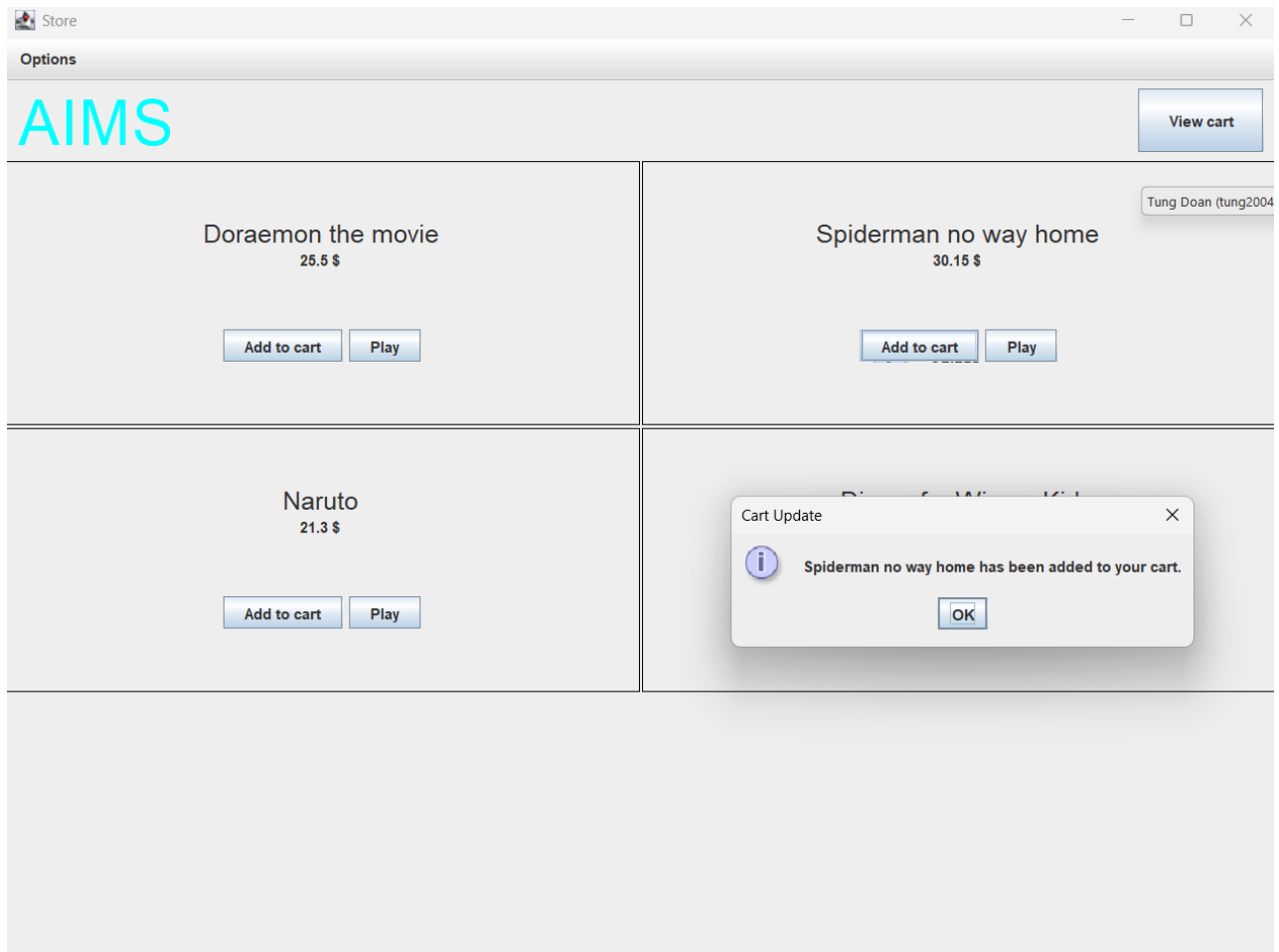


Figure 3.9 Demo Add to cart button



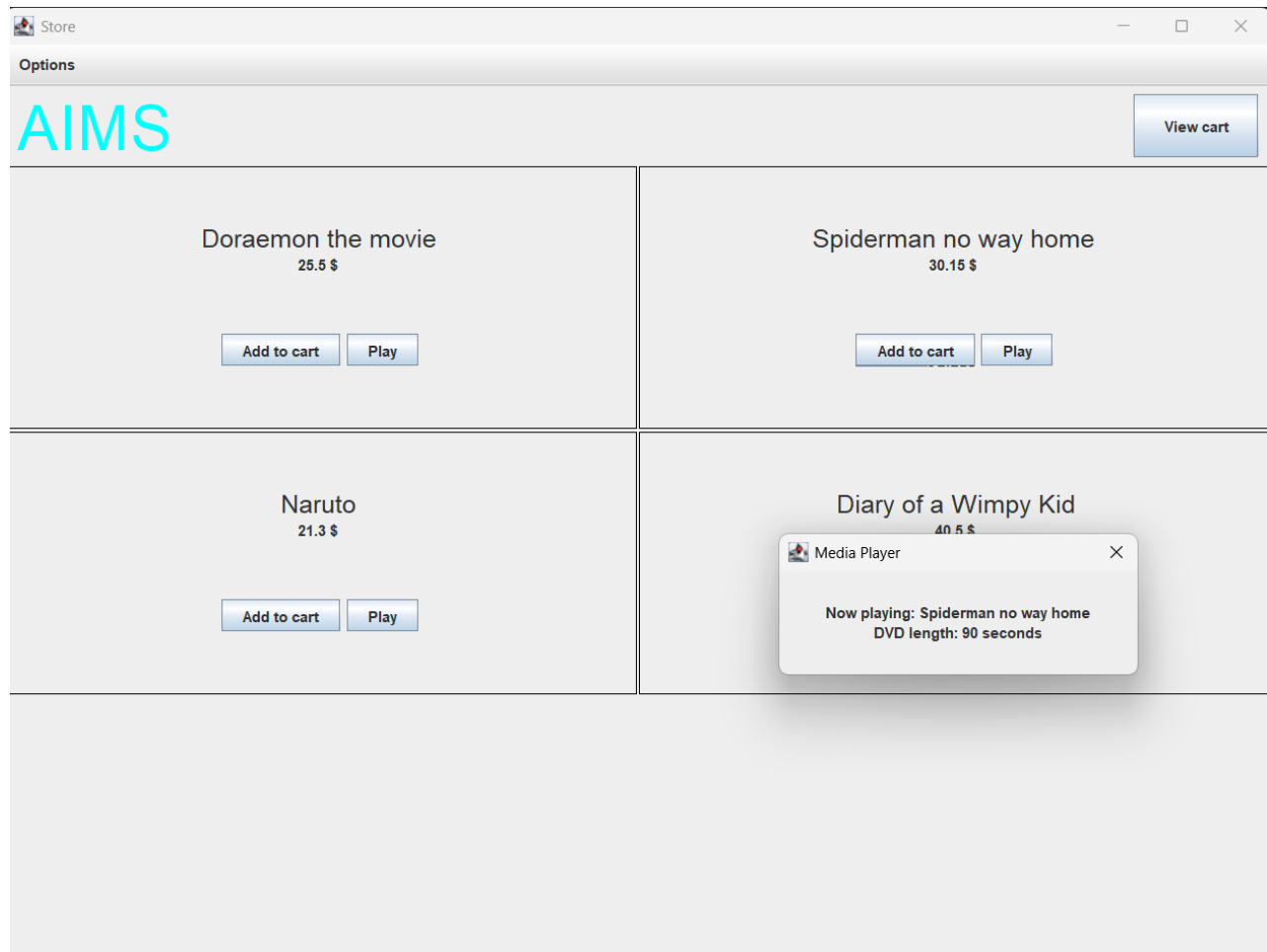


Figure 3.10 Demo Play button

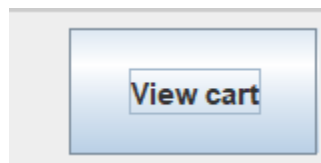


Figure 3.11 Demo View cart button

## 4 JavaFX API

### 4.1 Create class Painter

```
//DOAN THANH TUNG-20225946|
public class Painter extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("Painter.fxml"));
        Scene scene = new Scene(root);
        stage.setTitle("Painter");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

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 <BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="480.0" prefWidth="640.0"
13     xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="hust.soict.hedspi.javafx.PaintController">
14     <left>
15         <VBox maxHeight="1.7976931348623157E308" prefHeight="200.0" prefWidth="100.0" BorderPane.alignment="CENTER">
16             <BorderPane.margin>
17                 <Insets right="8.0" />
18             </BorderPane.margin>
19             <children>
20                 <TitledPane animated="false" text="Tools">
21                     <content>
22                         <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="102.0">
23                             <children>
24                                 <VBox layoutX="-1.0" layoutY="-19.0" prefHeight="200.0" prefWidth="100.0">
25                                     <children>
26                                         <RadioButton mnemonicParsing="false" text="RadioButton" />
27                                         <RadioButton fx:id="penButton" mnemonicParsing="false" onAction="#penMode" text="Pen">
28                                             <toggleGroup>
29                                                 <ToggleGroup fx:id="toolGroup" />
30                                             </toggleGroup>
31                                         </RadioButton>
32                                         <RadioButton fx:id="eraseButton" mnemonicParsing="false" onAction="#eraserMode" text="Erase">
33                                             <toggleGroup="toolGroup" />
34                                         </children>
35                                     </VBox>
36                                 </children>
37                             </AnchorPane>
38                         </content>
39                     </TitledPane>
40                     <Button maxWidth="1.7976931348623157E308" mnemonicParsing="false" onAction="#clearButtonPressed" text="Clear" />
41                 </children>
42             </VBox>
43         </left>
44         <padding>
```

Figure 4.2: Painter.fxml 1

```
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="480.0" prefWidth="640.0"
xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1" fx:controller="hust.soict.hedspi.javaafx.PaintController">
    <left>
        <VBox maxHeight="1.7976931348623157E308" prefHeight="200.0" prefWidth="100.0" BorderPane.alignment="CENTER">
            <BorderPane.margin>
                <Insets right="8.0" />
            </BorderPane.margin>
            <children>
                <TitledPane animated="false" text="Tools">
                    <content>
                        <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="102.0">
                            <children>
                                <VBox layoutX="-1.0" layoutY="-19.0" prefHeight="200.0" prefWidth="100.0">
                                    <children>
                                        <RadioButton mnemonicParsing="false" text="RadioButton" />
                                        <RadioButton fx:id="penButton" mnemonicParsing="false" onAction="#penMode" text="Pen">
                                            <toggleGroup>
                                                <ToggleGroup fx:id="toolGroup" />
                                            </toggleGroup>
                                        </RadioButton>
                                        <RadioButton fx:id="eraseButton" mnemonicParsing="false" onAction="#eraserMode" text="Erase">
                                            <toggleGroup="<!--
                                </children>
                            </VBox>
                        </children>
                    </AnchorPane>
                </content>
            </TitledPane>
            <Button maxWidth="1.7976931348623157E308" mnemonicParsing="false" onAction="#clearButtonPressed" text="Clear" />
        </children>
    </VBox>
</left>
<padding>
    <Insets bottom="8.0" left="8.0" right="8.0" top="8.0" />
</padding>
<center>
    <Pane fx:id="drawingAreaPane" onMouseDragged="#drawingAreaMouseDragged" prefHeight="200.0" prefWidth="200.0"
style="-fx-background-color: white;" BorderPane.alignment="CENTER" />
</center>
</BorderPane>
```

Figure 4.3: Painter.fxml 2

### 4.3 Create class PainterController

```
//DOAN THANH TUNG - 20225946
public class PaintController {
    boolean Eraser = false;
    private double eraserSize = 10.0;
    @FXML
    private Pane drawingAreaPane;

    @FXML
    void clearButtonPressed(ActionEvent event) {
        drawingAreaPane.getChildren().clear();
    }
    @FXML
    void drawingAreaMouseDragged(MouseEvent event) {
        // Check if the target is the drawing area
        if (event.getTarget() == drawingAreaPane) {
            if (Eraser) {
                Circle eraser = new Circle(event.getX(), event.getY(), eraserSize, Color.WHITE);
                drawingAreaPane.getChildren().add(eraser);
            } else {
                Circle pen = new Circle(event.getX(), event.getY(), 4, Color.AQUA);
                drawingAreaPane.getChildren().add(pen);
            }
        }
    }
    @FXML
    void penMode(ActionEvent event) {
        Eraser = false;
    }
    @FXML
    void eraserMode(ActionEvent event) {
        Eraser = true;
    }
}
```

Figure 4.4: PainterController

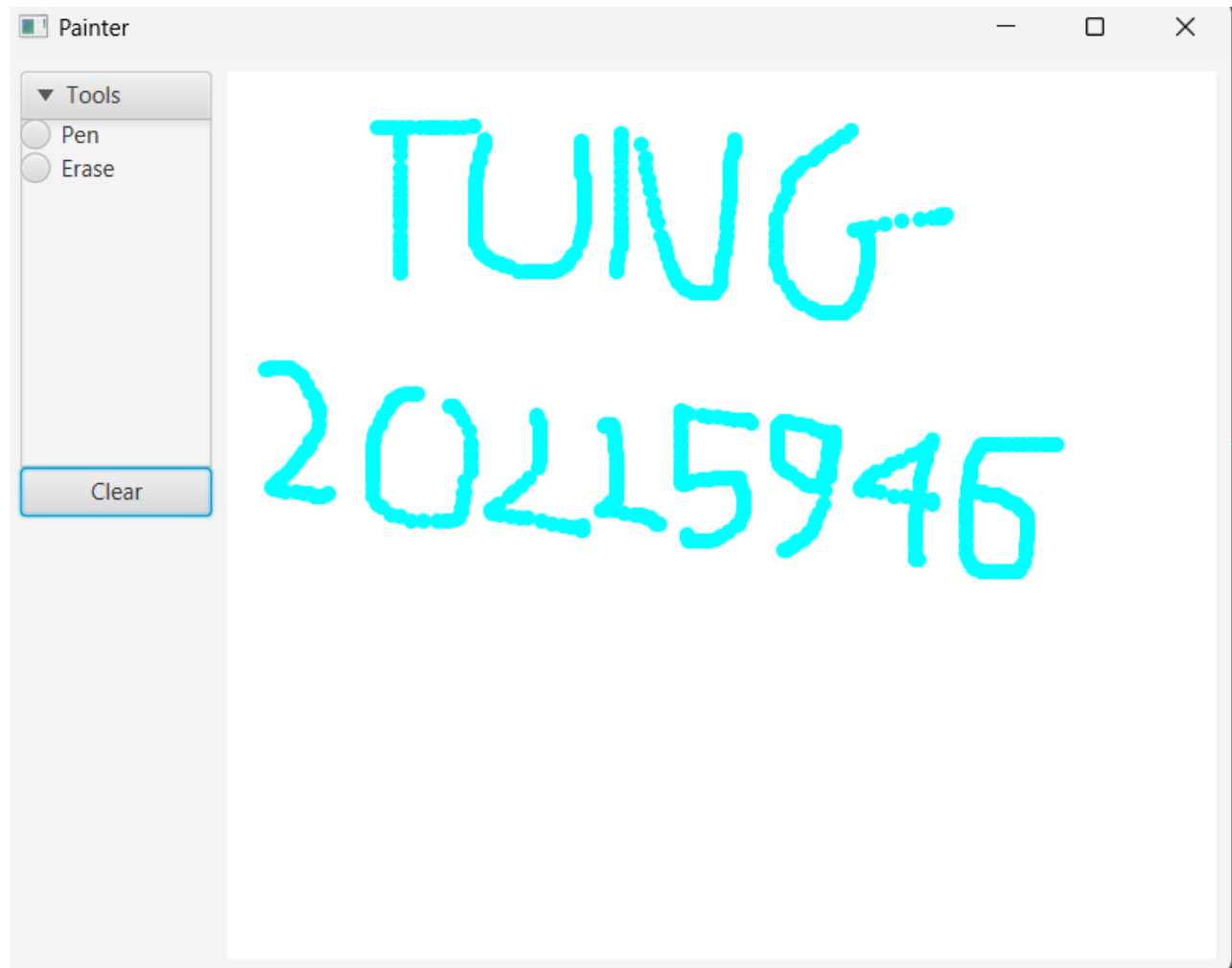


Figure 4.5: Use Pen

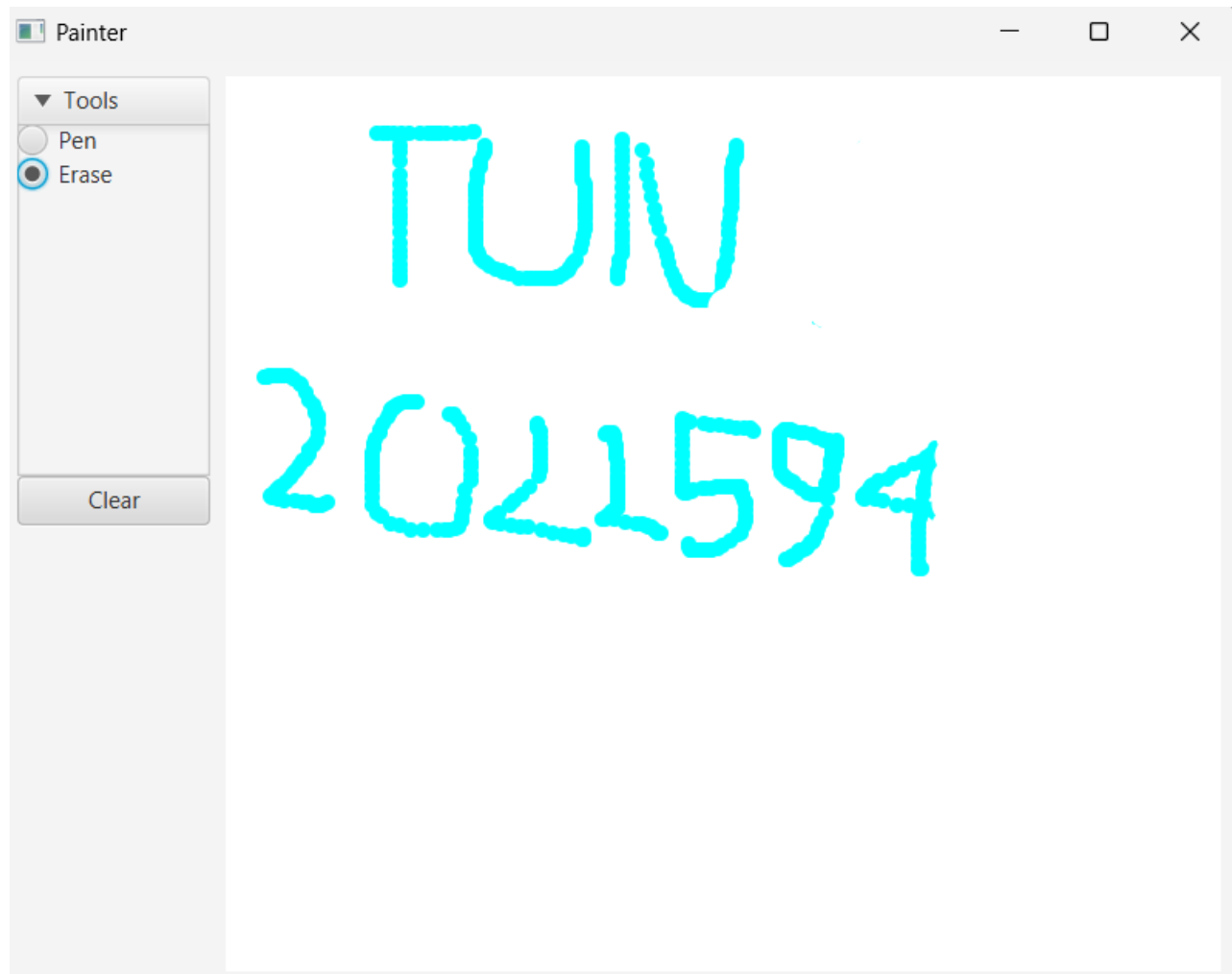


Figure 4.6: Use Eraser

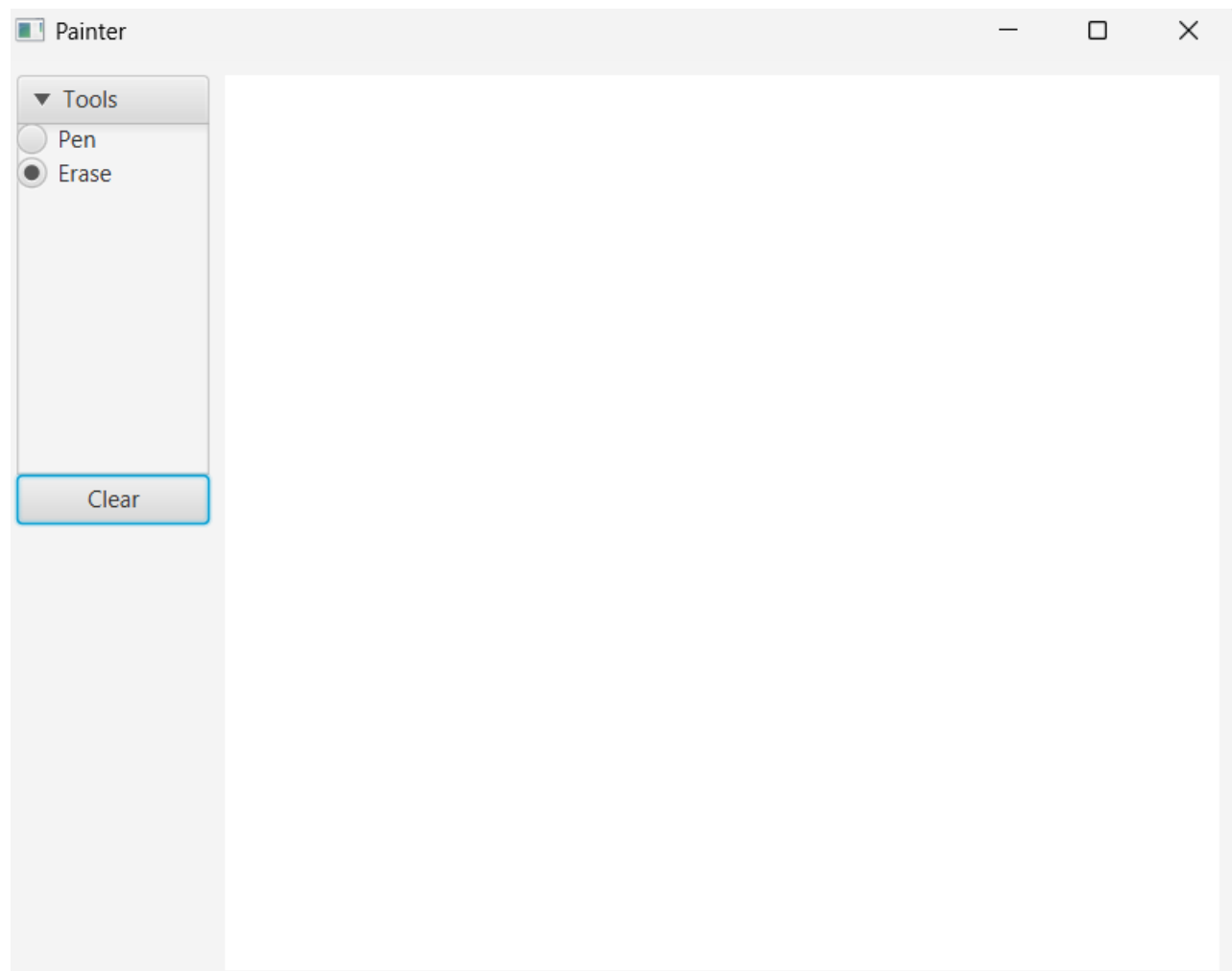


Figure 4.7: Clear button

## 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.text.Font?>
18
19 <BorderPane maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" minHeight="-Infinity" minWidth="-Infinity" prefHeight="768.0"
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" text="Add Book" />
30 <MenuItem mnemonicParsing="false" text="Add CD" />
31 <MenuItem mnemonicParsing="false" text="Add DVD" />

```

Figure 5.1: Cart.fxml 1

```

<padding>
<Insets left="10.0" right="10.0" top="30.0" />
</padding>
<children>
<HBox alignment="CENTER" spacing="10.0">
<children>
<Label text="Total">
<font>
<Font size="24.0" />
</font>
</Label>
<Label fx:id="costLabel" text="0 $" textFill="AQUA">
<font>
<Font size="24.0" />
</font>
</Label>
</children>
</HBox>
<Button mnemonicParsing="false" onAction="#placeOrderPressed" style="-fx-background-color: RED;" text="Place Order" textFill="WHITE">
<font>
<Font size="24.0" />
</font>
</Button>
</children>
<BorderPane.margin>
<Insets />
</BorderPane.margin>
</VBox>
</right>
</BorderPane>

```

Figure 5.2: Cart.fxml 2

```

        <RadioButton mnemonicParsing="false" onAction="#setSortBycost" text="By Cost" toggleGroup="$sortCategory" />
    </children>
</HBox>
<HBox alignment="CENTER_RIGHT" maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" spacing="10.0">
    <children>
        <Button fx:id="btnPlay" mnemonicParsing="false" onAction="#playButtonPressed" text="Play" />
        <Button fx:id="btnRemove" mnemonicParsing="false" onAction="#removeButtonPressed" text="Remove" />
        <Button fx:id="btnDetails" mnemonicParsing="false" onAction="#detailsButtonPressed" text="Details" />
    </children>
</HBox>
</children>
<padding>
    <Insets bottom="40.0" top="10.0" />
</padding>
</HBox>
</children>
</VBox>
</center>
<right>
    <VBox alignment="TOP_CENTER" prefHeight="200.0" spacing="20.0" BorderPane.alignment="CENTER">
        <padding>
            <Insets left="10.0" right="10.0" top="30.0" />
        </padding>
        <children>
            <HBox alignment="CENTER" spacing="10.0">
                <children>
                    <Label text="Total">
                        <font>
                            <Font size="24.0" />
                        </font>
                    </Label>
                </children>
            </HBox>
        </children>
    </VBox>
</right>
</BorderPane>
</GridPane>
</Scene>
</Stage>
</FXML>

```

Figure 5.3: Cart.fxml 3

## 5.2 Create class CartScreen

```

public static void main(String args[]) throws Exception {
    // Test
    DigitalVideoDisc dvd1 = new DigitalVideoDisc(1, "The Goblin King", "Animation", "Someone", 87,
    DigitalVideoDisc dvd2 = new DigitalVideoDisc(2, "Star Wars", "Sci-fi", "George Lucas", 87, 24.
    DigitalVideoDisc dvd3 = new DigitalVideoDisc(3, "Aladin", "Animation", 18.99f);
    Track track1 = new Track(1, "Wei");
    Track track2 = new Track(2, "Shu");
    Track track3 = new Track(3, "Wu");
    CompactDisc cd1 = new CompactDisc(1, "ROTK OST (Part 1)", "Drama", "Various", 30.95f);
    cd1.addTrack(track1);
    cd1.addTrack(track2);
    CompactDisc cd2 = new CompactDisc(2, "ROTK OST (Part 2)", "Drama", "Various", 25.99f);
    cd2.addTrack(track3);
    Cart myCart = new Cart();
    myCart.addMedia(dvd1);
    myCart.addMedia(dvd2);
    myCart.addMedia(dvd3);
    myCart.addMedia(cd1);
    myCart.addMedia(cd2);
}

```

Figure 5.4: CartScreen class



### 5.3 Create class CartScreenController

```
public class CartScreenController {
    private Store store;
    private Cart cart;
    private boolean filterById = true;
    private boolean sortByTitle = true;
    private FilteredList<Media> filteredCart;
    private JFrame stage;
    @FXML
    private TableView<Media> tblMedia;
    @FXML
    private TableColumn<Media, String> colMediaTitle;
    @FXML
    private TableColumn<Media, String> colMediaCategory;
    @FXML
    private TableColumn<Media, String> colMediaCost;
    @FXML
    private Button btnPlay;
    @FXML
    private Button btnRemove;
    @FXML
    private Button btnDetails;
    @FXML
    private TextField tfFilter;
    @FXML
    private Label costLabel;

    public CartScreenController(Store store, Cart cart, JFrame stage) {
        super();
        this.store = store;
    }
}
```

Figure 5.5: CartScreenController 1

```
@FXML
public void initialize() {
    colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>("title"));
    colMediaCategory.setCellValueFactory(new PropertyValueFactory<Media, String>("category"));
    colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, String>("cost"));
    tblMedia.setItems(filteredCart);
    btnPlay.setVisible(false);
    btnRemove.setVisible(false);
    btnDetails.setVisible(false);
    costLabel.setText(String.valueOf(this.cart.totalcost()));
    tblMedia.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Media>() {
        @Override
        public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {
            updateButtonBar(newValue);
        }
    });
    tfFilter.textProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
            showFilteredMedia(newValue);
        }
    });
}

private void updateButtonBar(Media media) {
    if (media == null) {
        btnRemove.setVisible(false);
        btnDetails.setVisible(false);
        btnPlay.setVisible(false);
    } else {
        btnRemove.setVisible(true);
        btnDetails.setVisible(true);
    }
}
```

Figure 5.6: CartScreenController 2

```
private void updateButtonBar(Media media) {
    if (media == null) {
        btnRemove.setVisible(false);
        btnDetails.setVisible(false);
        btnPlay.setVisible(false);
    } else {
        btnRemove.setVisible(true);
        btnDetails.setVisible(true);
        if (media instanceof Playable) {
            btnPlay.setVisible(true);
        } else {
            btnPlay.setVisible(false);
        }
    }
}

private void showFilteredMedia(String filter) {
    if (filter == null || filter.length() == 0) {
        filteredCart.setPredicate(s -> true);
    } else {
        if (filterById) {
            try {
                filteredCart.setPredicate(s -> s.getId() == Integer.parseInt(filter));
            } catch (NumberFormatException e) {
            }
        } else {
            filteredCart.setPredicate(s -> s.getTitle().toLowerCase().contains(filter));
        }
    }
}
```

Figure 5.7: CartScreenController 3

## 5.4 Demo

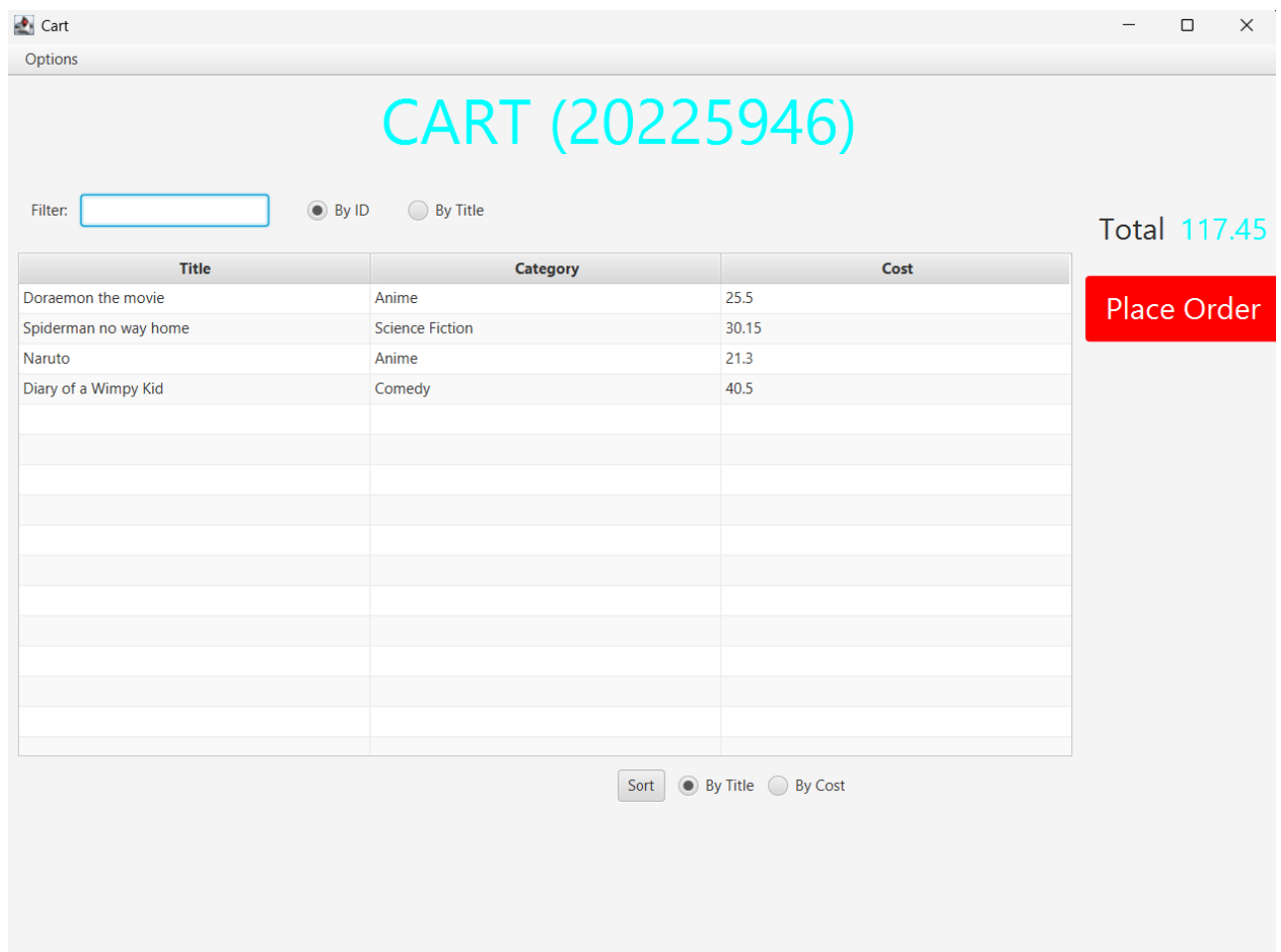


Figure 5.7: Demo CartScreen

## 6 Updating buttons based on selected item in TableView – ChangeListener

### 6.1 Edit class CartScreenController

```
public CartScreenController(Store store, Cart cart, JFrame stage) {
    super();
    this.store = store;
    this.cart = cart;
    this.stage = stage;
}

@FXML
public void initialize() {
    colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>("title"));
    colMediaCategory.setCellValueFactory(new PropertyValueFactory<Media, String>("category"));
    colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, String>("cost"));
    tblMedia.setItems(filteredCart);
    btnPlay.setVisible(false);
    btnRemove.setVisible(false);
    btnDetails.setVisible(false);
    costLabel.setText(String.valueOf(this.cart.totalcost()));
    tblMedia.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Media>() {
        @Override
        public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {
            updateButtonBar(newValue);
        }
    });
    tffilter.textProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
            showFilteredMedia(newValue);
        }
    });
}
```

Figure 6.1: CartScreenController 1

```
private void updateButtonBar(Media media) {  
    if (media == null) {  
        btnRemove.setVisible(false);  
        btnDetails.setVisible(false);  
        btnPlay.setVisible(false);  
    } else {  
        btnRemove.setVisible(true);  
        btnDetails.setVisible(true);  
        if (media instanceof Playable) {  
            btnPlay.setVisible(true);  
        } else {  
            btnPlay.setVisible(false);  
        }  
    }  
}  
  
private void showFilteredMedia(String filter) {  
    if (filter == null || filter.length() == 0) {  
        filteredCart.setPredicate(s -> true);  
    } else {  
        if (filterById) {  
            try {  
                filteredCart.setPredicate(s -> s.getId() == Integer.parseInt(filter));  
            } catch (NumberFormatException e) {  
            }  
        } else {  
            filteredCart.setPredicate(s -> s.getTitle().toLowerCase().contains(filter));  
        }  
    }  
}
```

Figure 6.2: CartScreenController 2

## 6.2 Demo

Cart

Options

CART (20225946)

Filter:

☒ By ID ☐ By Title

Title	Category	Cost
Doraemon the movie	Anime	25.5
Spiderman no way home	Science Fiction	30.15
Naruto	Anime	21.3
Diary of a Wimpy Kid	Comedy	40.5

Sort

☒ By Title ☐ By Cost

Play

Remove

Details

Total 117.45

Place Order

Figure 6.3: Demo media playable

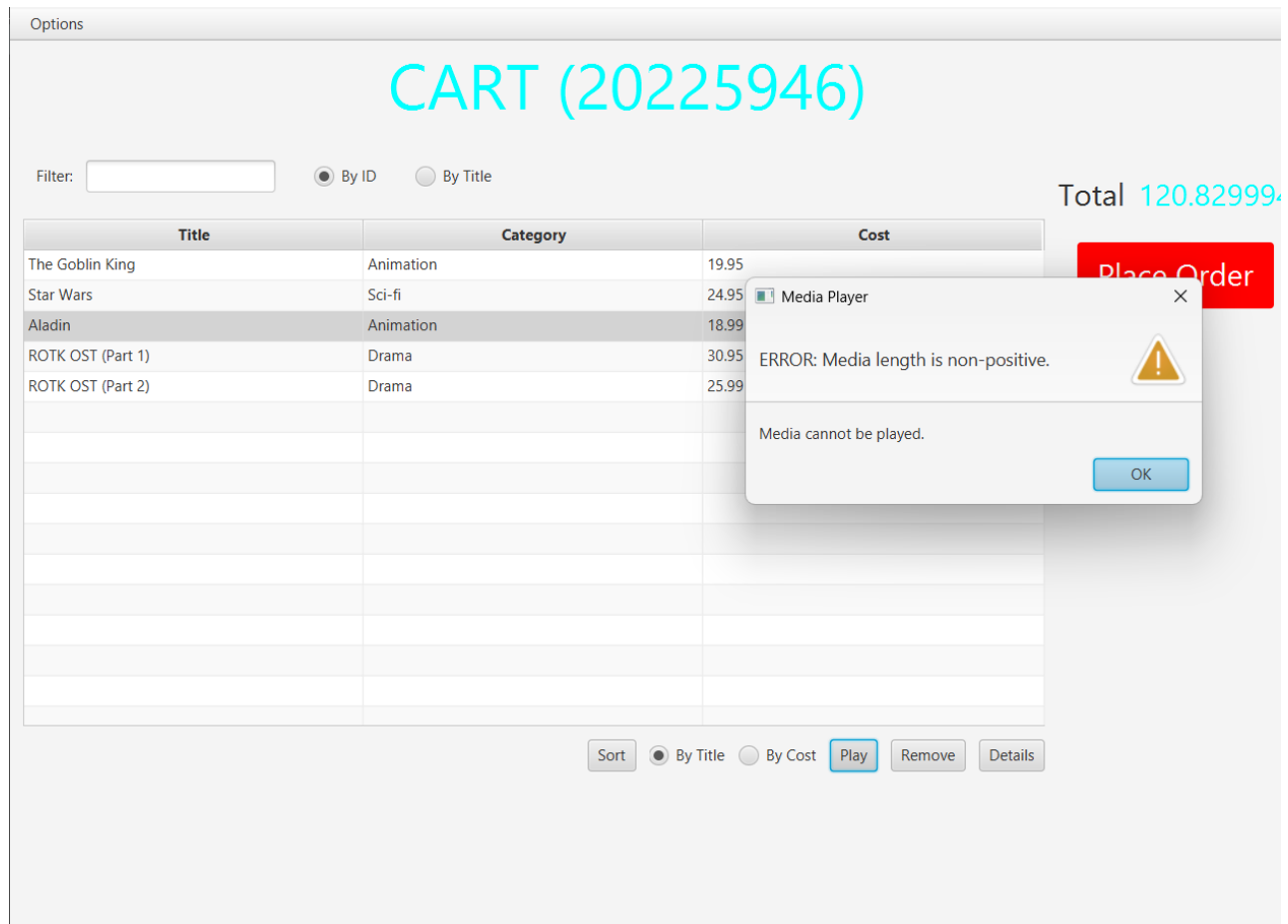


Figure 6.4: Demo media unplayable

## 7 Deleting a media

### 7.1 Code

```
@FXML
public void playButtonPressed(ActionEvent event) {
    Media media = this.tblMedia.getSelectionModel().getSelectedItem();
    try {
        ((Playable) media).play();
    } catch (UserException e) {
        Alert alert = new Alert(AlertType.WARNING);
        alert.setTitle("Media Player");
        alert.setHeaderText("ERROR: Media length is non-positive.");
        alert.setContentText("Media cannot be played.");
        alert.showAndWait();
    }
}
```

Figure 7.1: btnRemovePressed Method

## 7.2 Demo

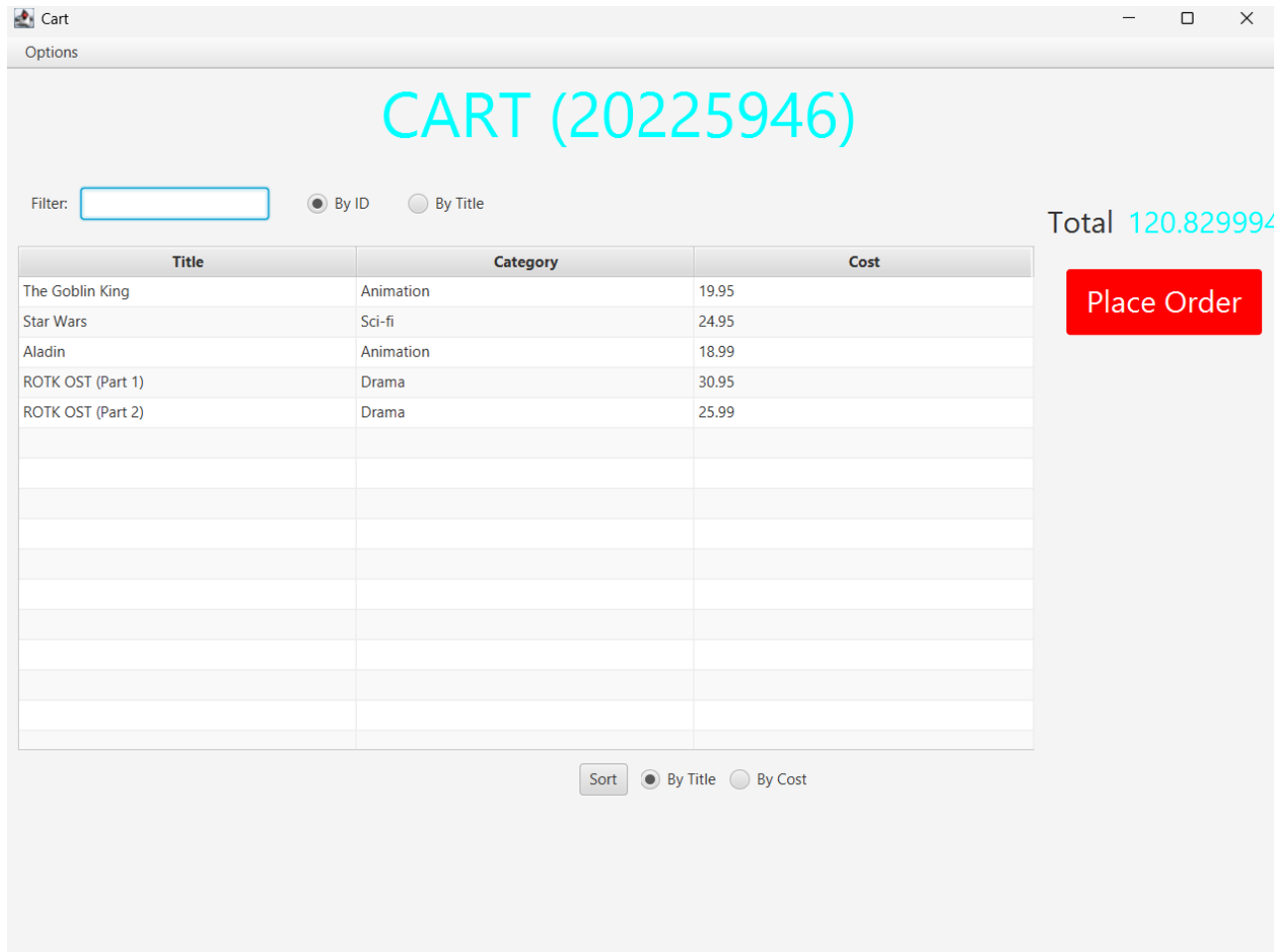


Figure 7.2: button Remove



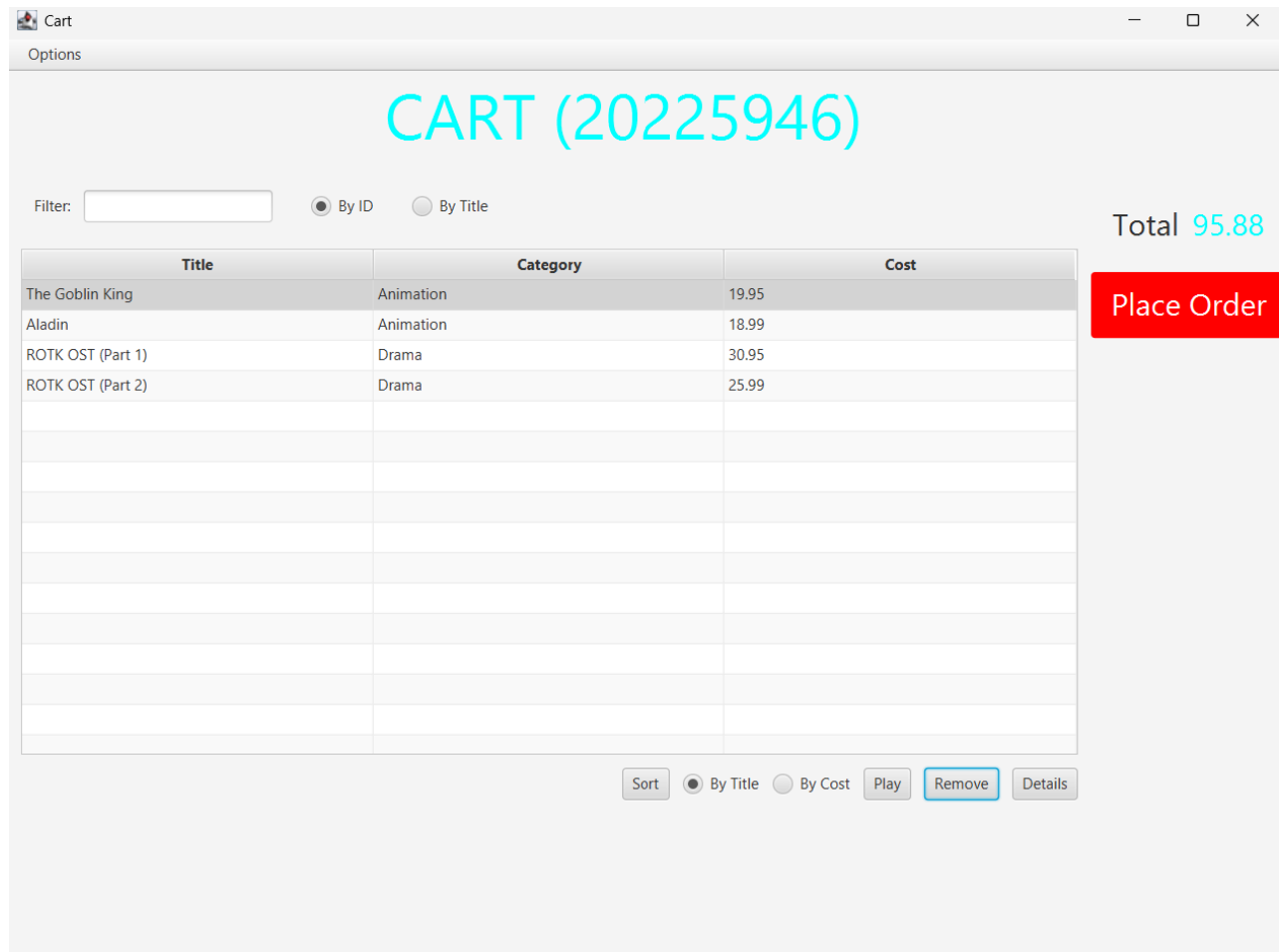


Figure 7.3: after Remove

## 8 Complete the Aims GUI application

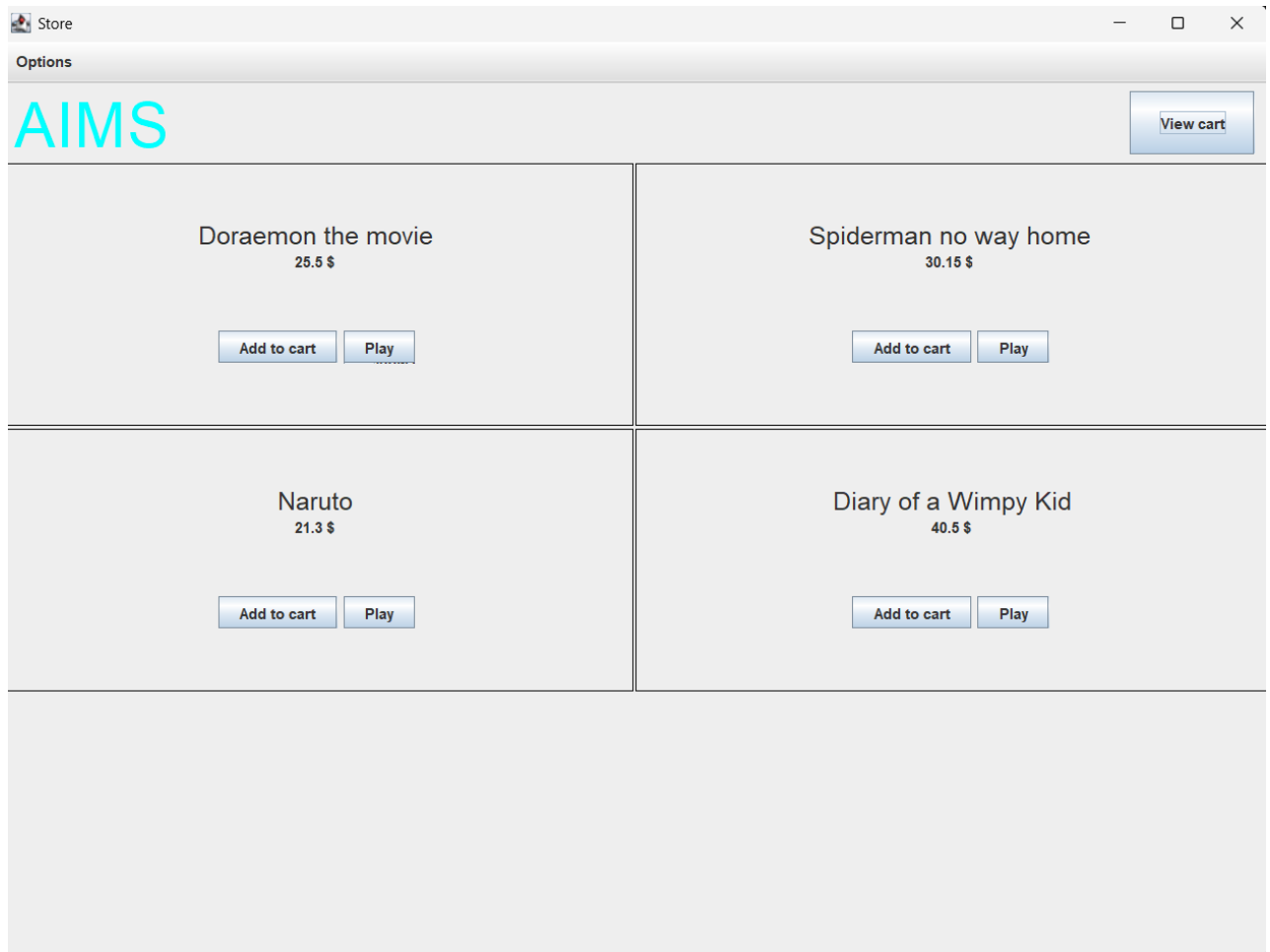
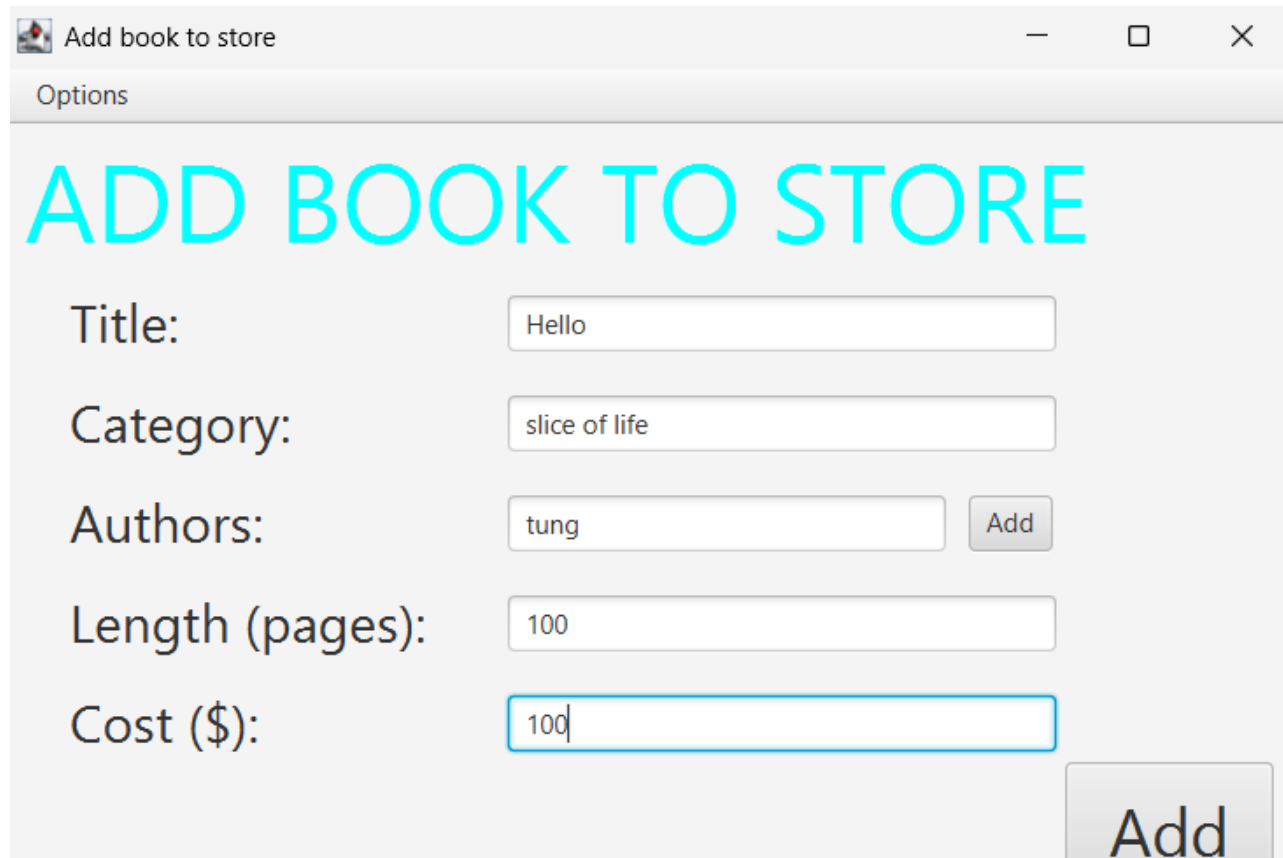


Figure 8.1: Store before add book



The screenshot shows a web browser window with the title "Add book to store". The page has a light gray background and a large cyan heading "ADD BOOK TO STORE". Below the heading, there are five input fields for book information: "Title:" with the value "Hello", "Category:" with the value "slice of life", "Authors:" with the value "tung" and a small "Add" button next to it, "Length (pages):" with the value "100", and "Cost (\$):" with the value "100". A large "Add" button is located at the bottom right of the form area.

Add book to store

Options

# ADD BOOK TO STORE

Title:

Category:

Authors:

Length (pages):

Cost (\$):

Figure 8.2: Add book

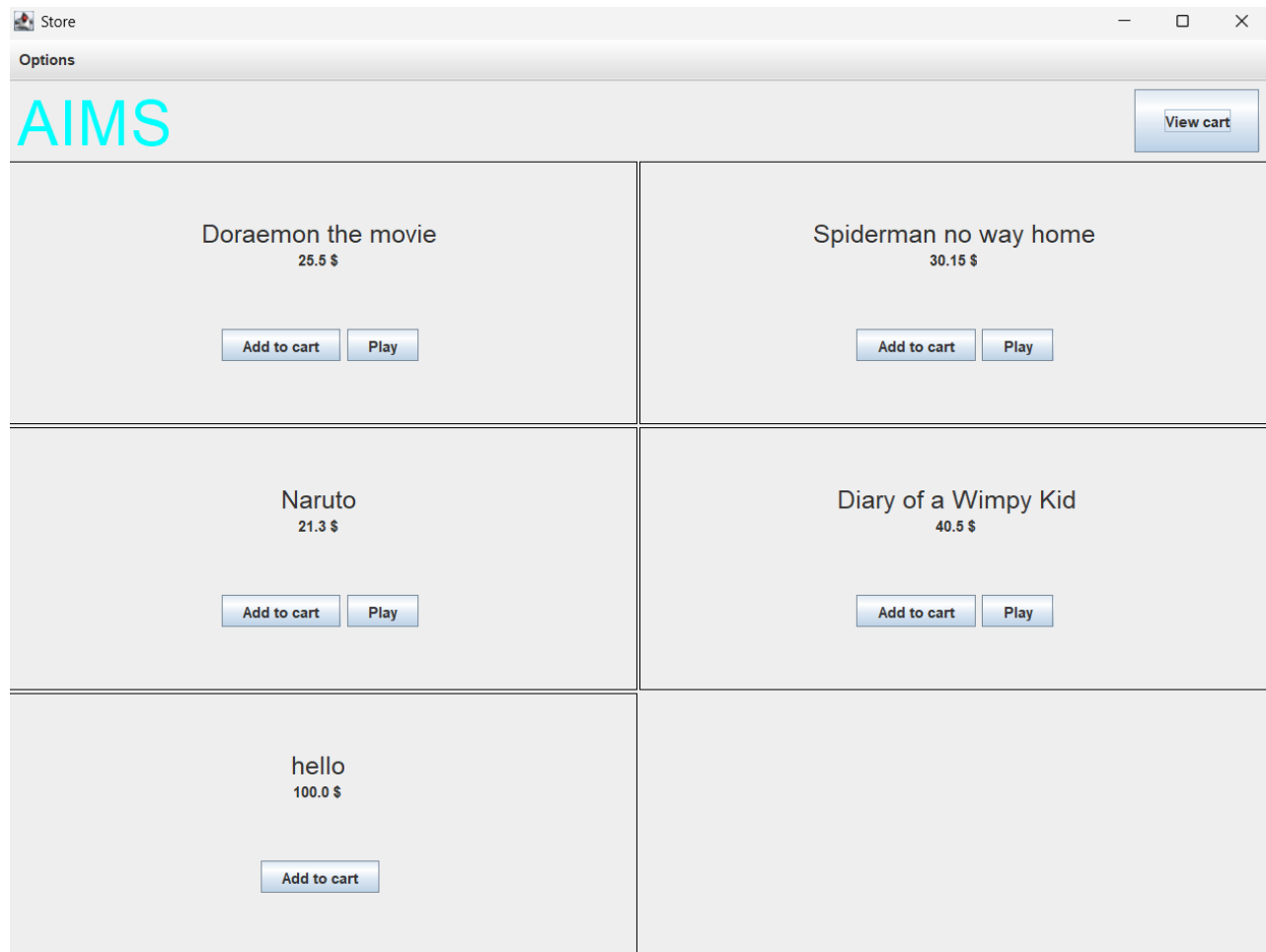
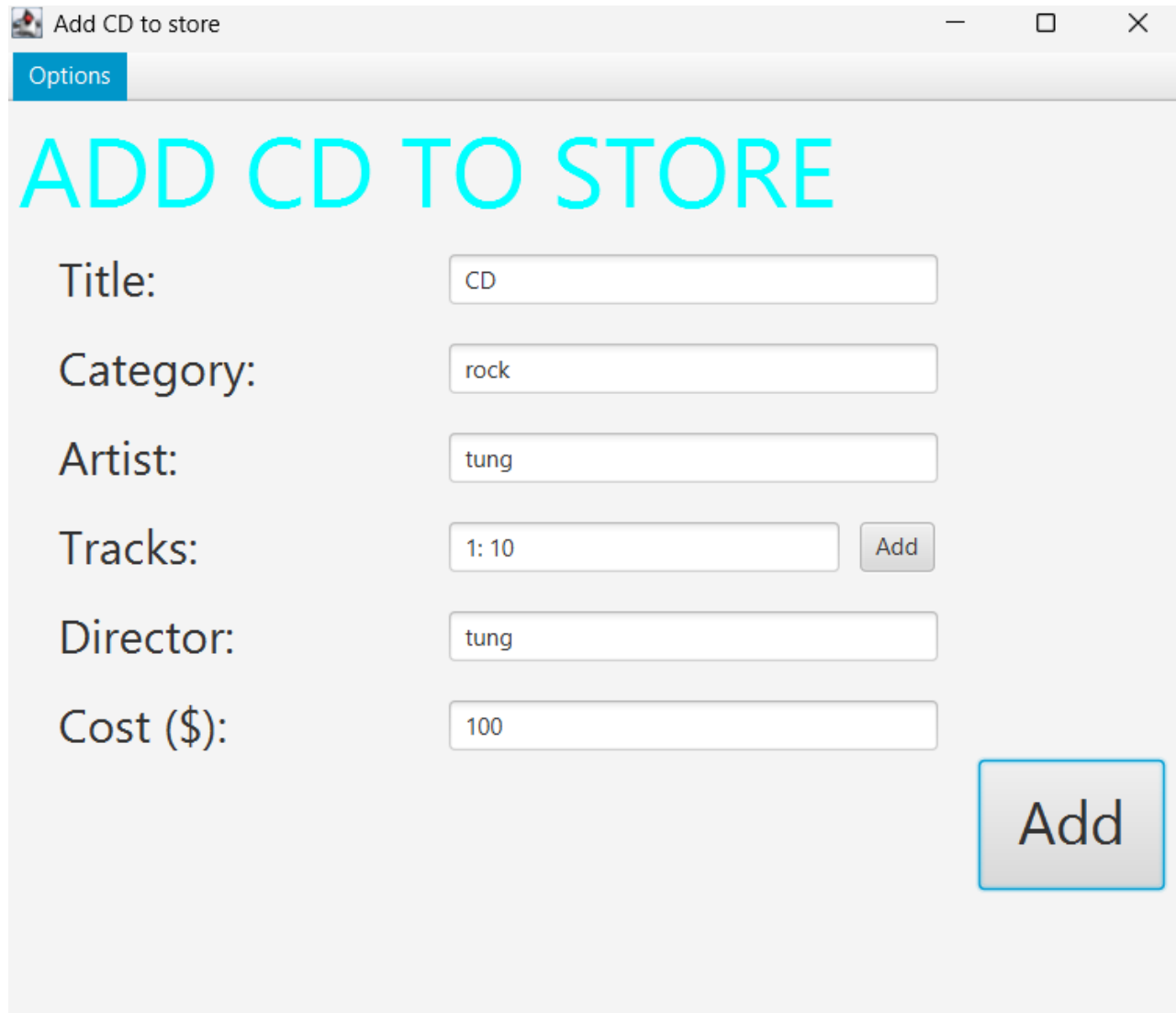


Figure 8.3: Store after add book



Add CD to store

Options

# ADD CD TO STORE

Title:

Category:

Artist:

Tracks:

Director:

Cost (\$):

Figure 8.4: Add CD

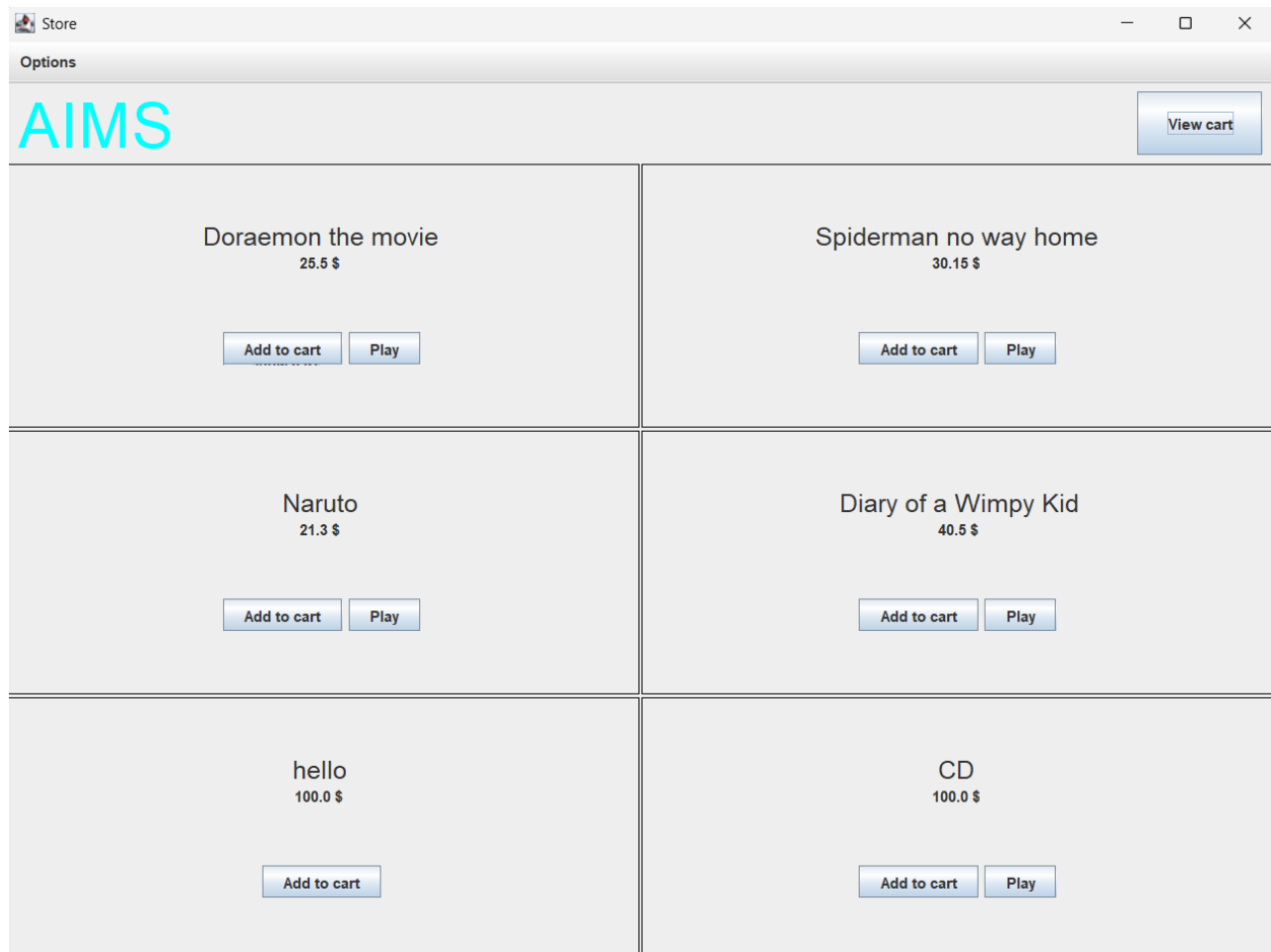
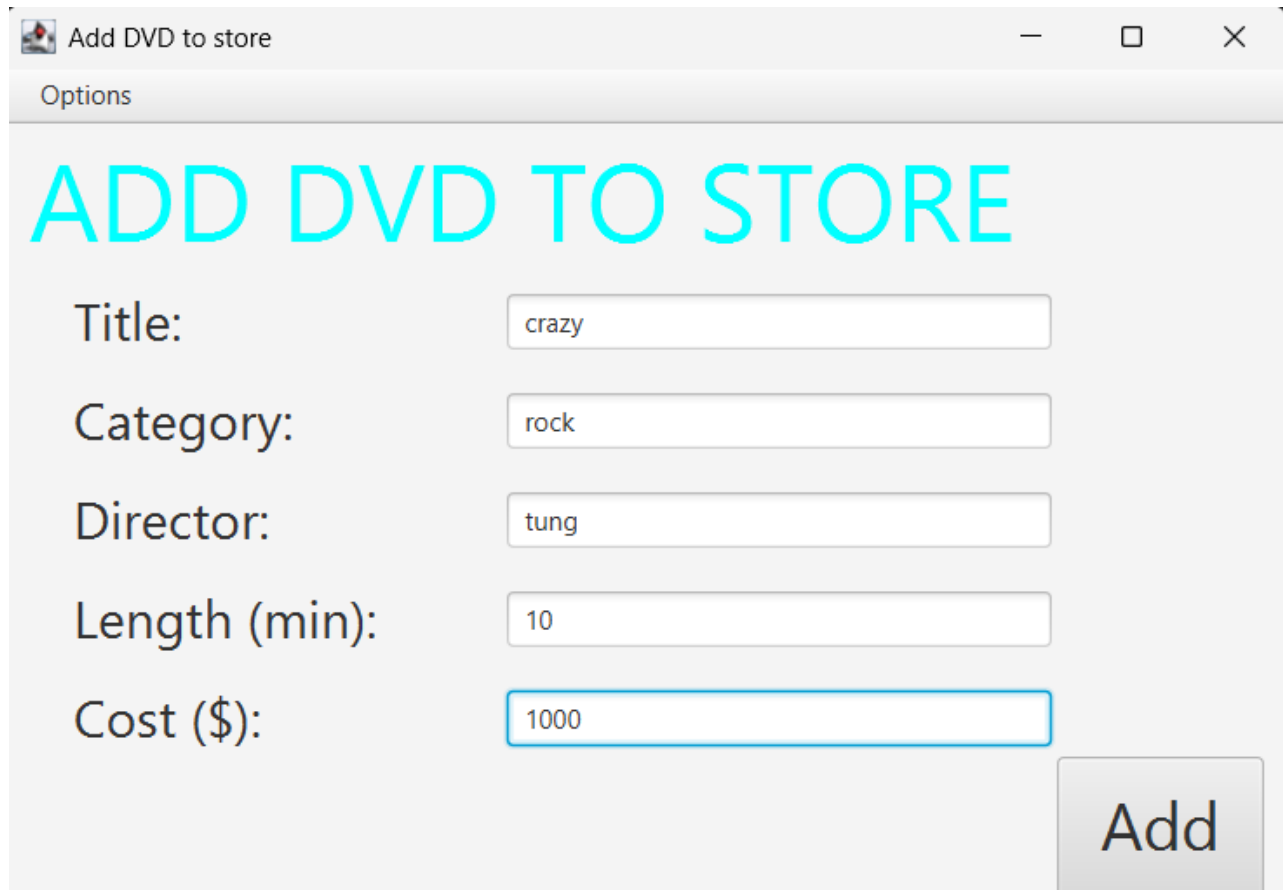


Figure 8.5: Store after add CD



The screenshot shows a web browser window with the title "Add DVD to store". Below the title bar is a header area with the word "Options". The main content area has the heading "ADD DVD TO STORE" in large, bold, cyan letters. Below this heading are five form fields, each with a label and an input box:

- Title: crazy
- Category: rock
- Director: tung
- Length (min): 10
- Cost (\$): 1000

At the bottom right of the form is a button labeled "Add".

Figure 8.6 Add DVD

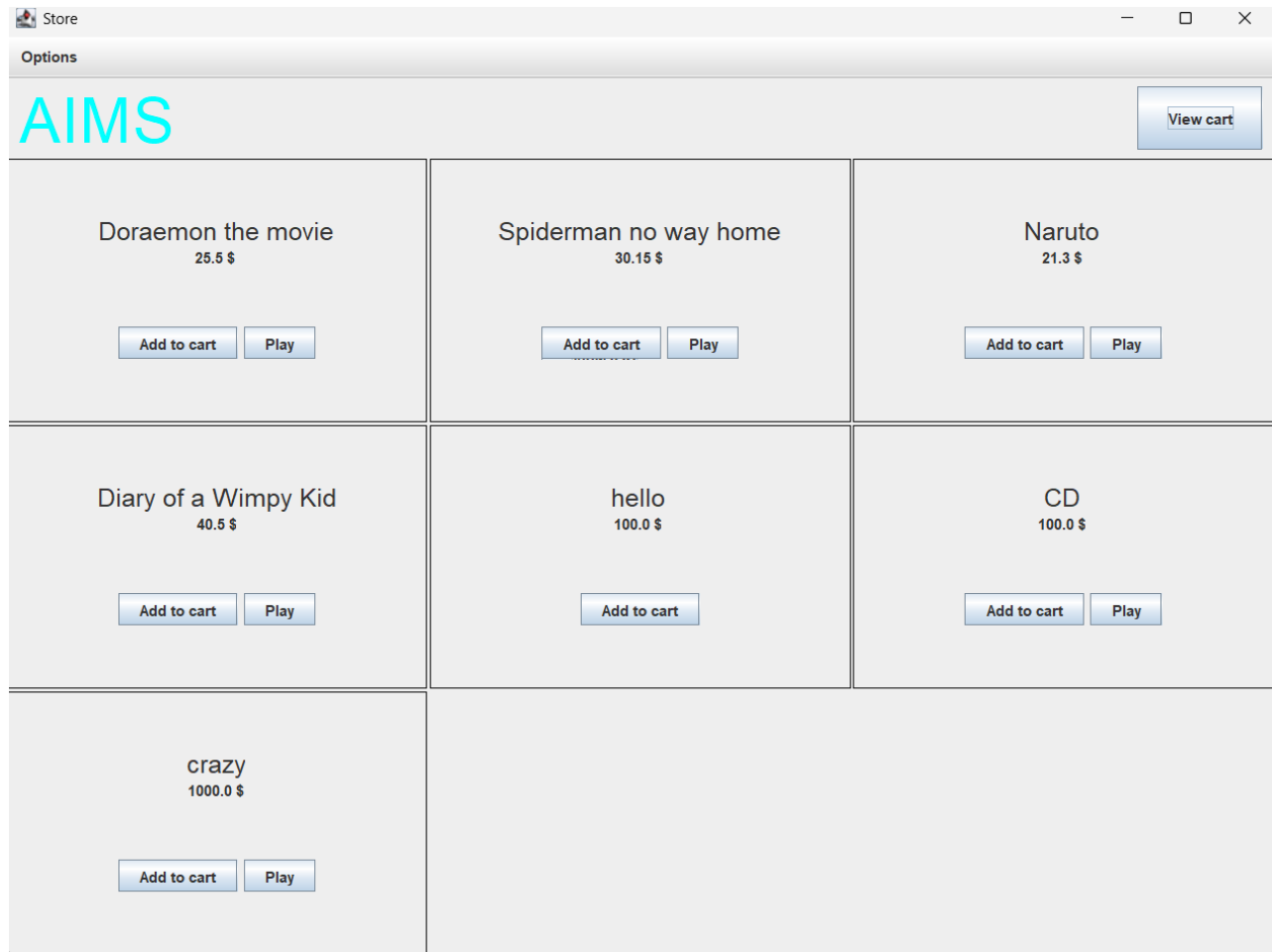



Figure 8.7: Store after add DVD



 Cart

Options

CART (20225946)

Filter:

☒ By ID ☐ By Title

Total 1317.45

Place Order

Title	Category	Cost
Doraemon the movie	Anime	25.5
Spiderman no way home	Science Fiction	30.15
Naruto	Anime	21.3
Diary of a Wimpy Kid	Comedy	40.5
hello	slice of life	100.0
CD	rock	100.0
crazy	rock	1000.0

Sort

☒ By Title ☐ By Cost

Figure 8.8: Cart

```
package hust.soict.hedspi.aims.Exception;

public class UserException extends Exception {

    public UserException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public UserException(String message, Throwable cause) {
        super(message, cause);
        // TODO Auto-generated constructor stub
    }

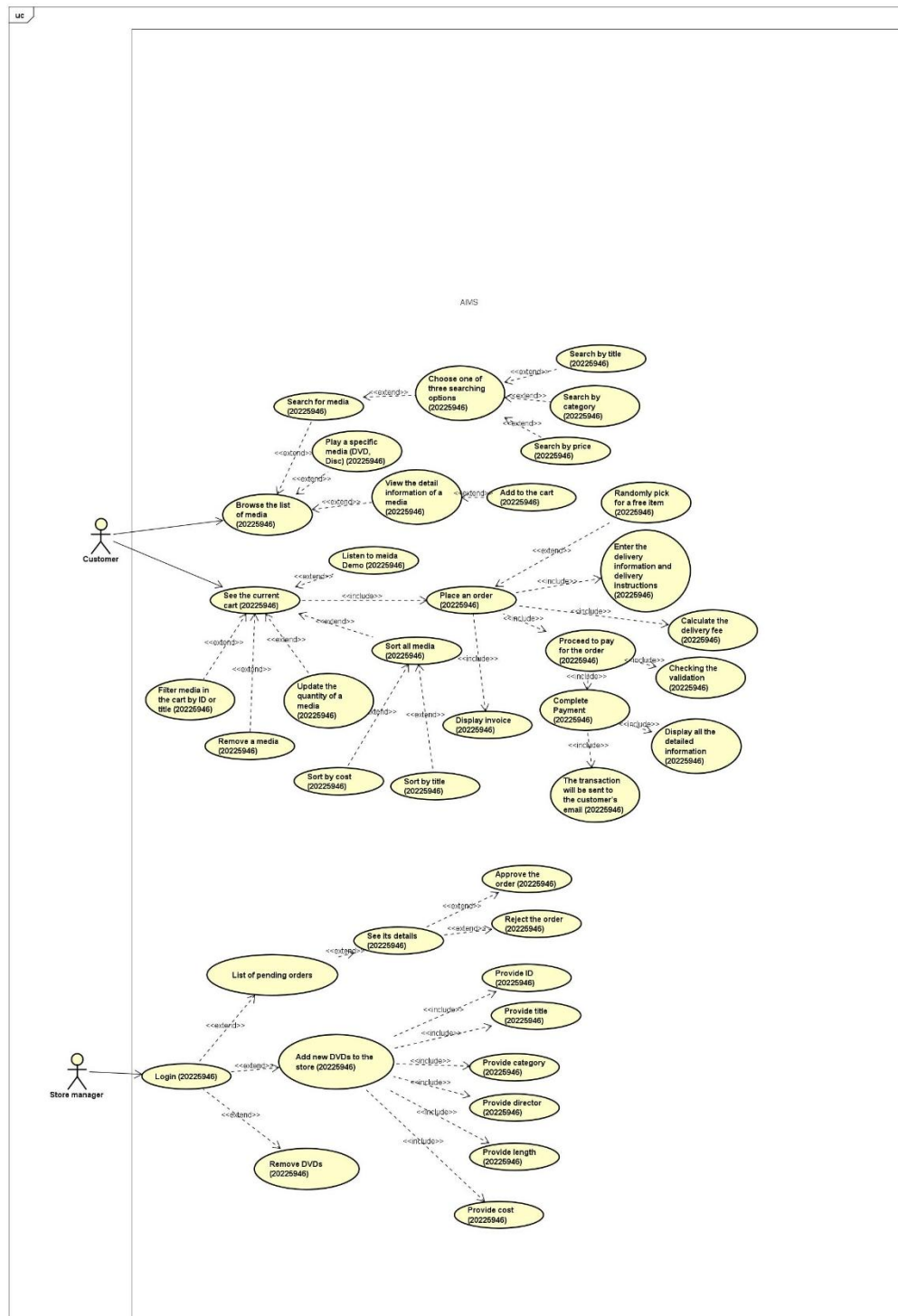
    public UserException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

    public UserException(Throwable cause) {
        super(cause);
        // TODO Auto-generated constructor stub
    }

}
```

Figure 8.9: Exception

## 9 Use case Diagram



## 10 Class Diagram

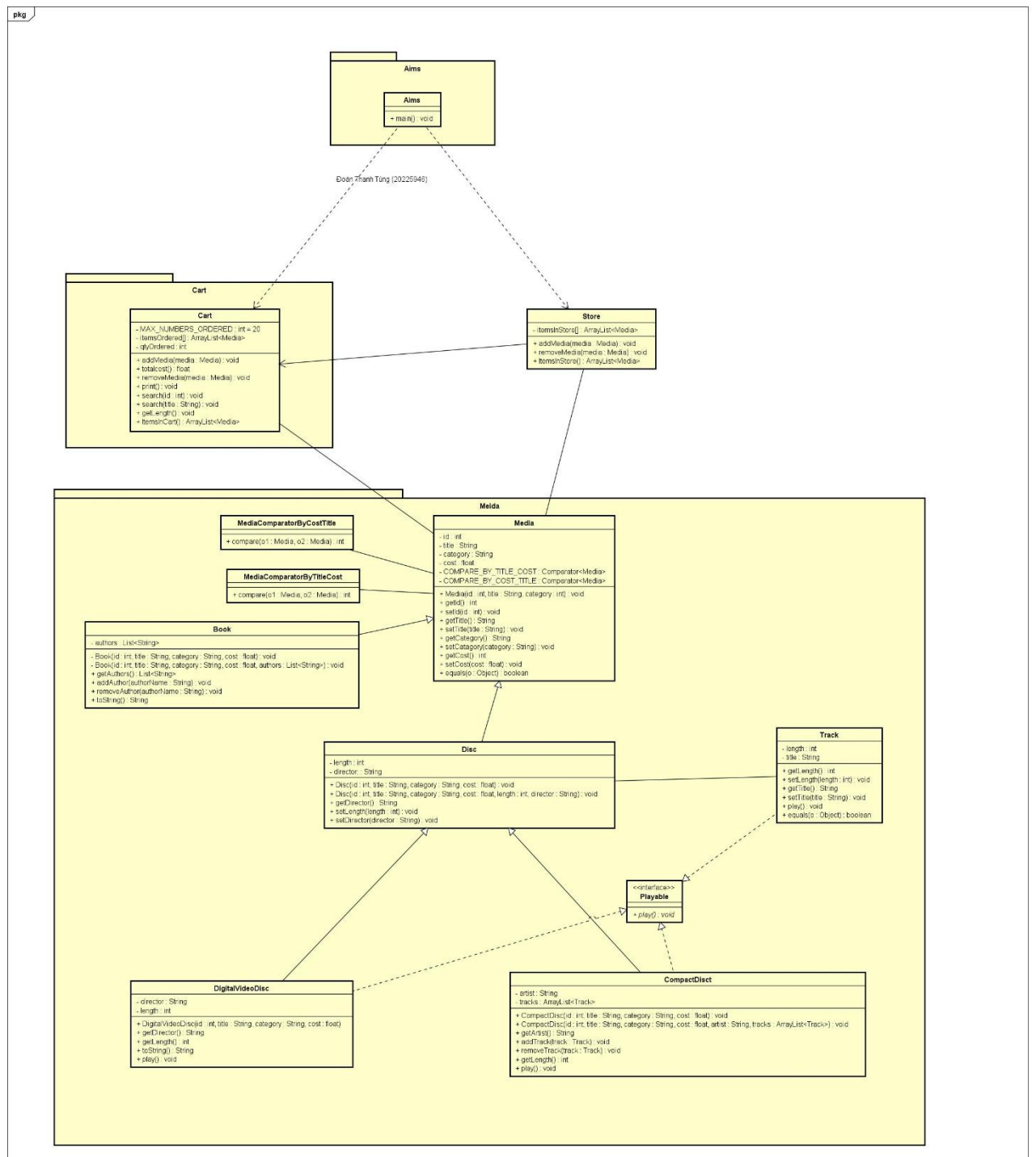


Figure 10.1: Class Diagram