

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

BÁO CÁO THỰC HÀNH
IT3103
BÀI THỰC HÀNH 5

Họ và tên sv: Nguyễn Minh Tuấn
Lớp: **K67-CNTT Việt Pháp 01**

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 component

1.1. Lớp AWTAccumulator

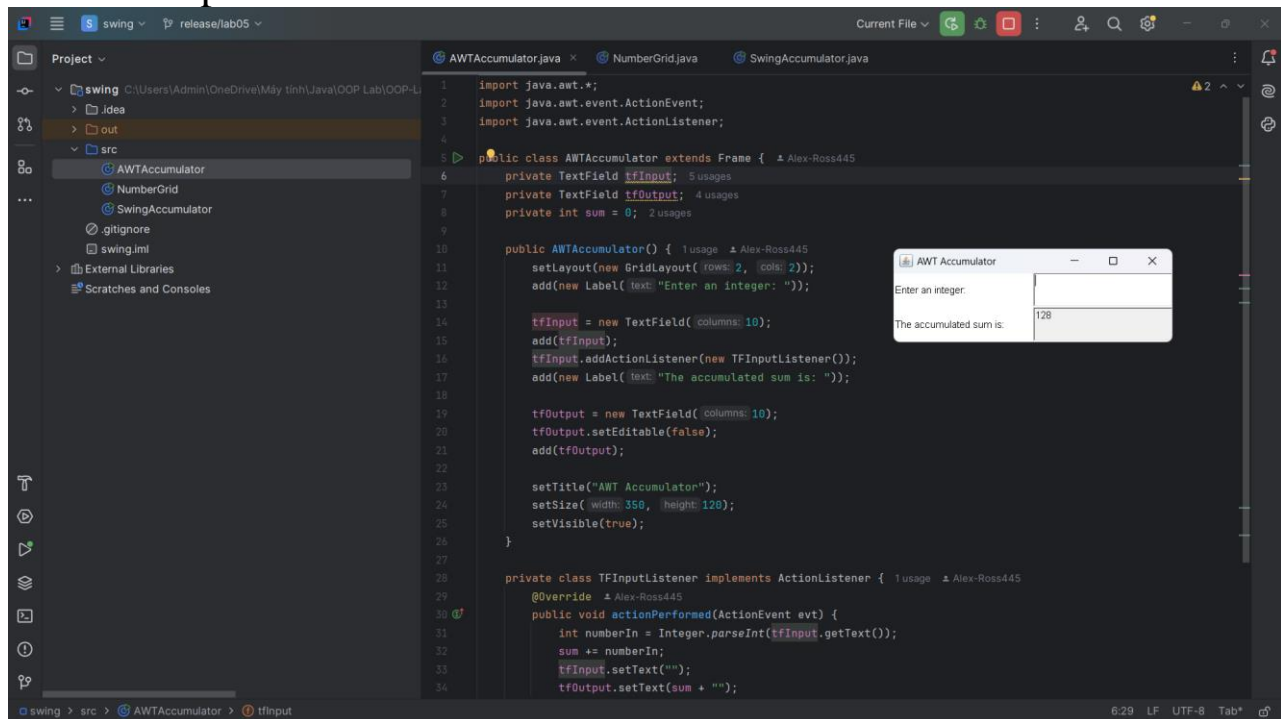


Figure 1: Mã nguồn và giao diện lớp AWTAccumulator

1.2. Lớp SwingAccumulator

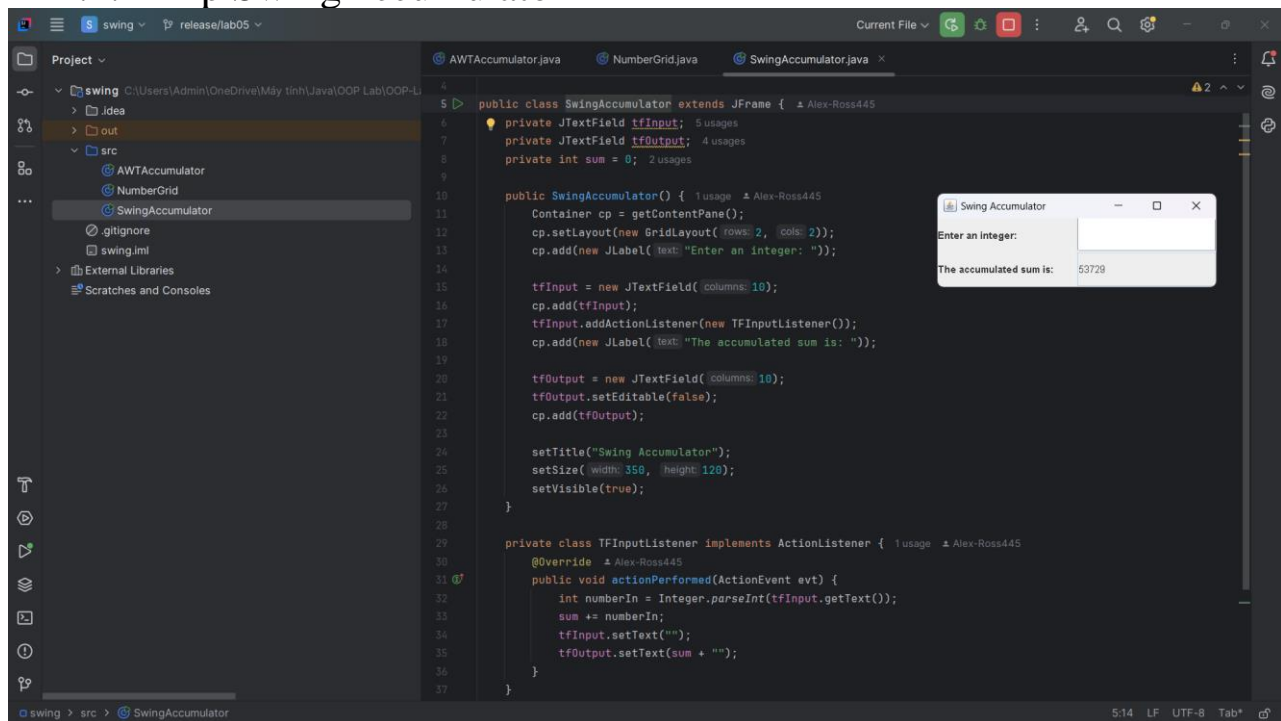


Figure 2: Giao diện và mã nguồn lớp SwingAccumulator

1.3: Compare Swing and AWT elements

- The top-level containers in Swing and AWT:
 - Swing: JFrame, JDialog, JApplet, JWindow
 - AWT: Frame, Dialog, Applet
- The class name of components in AWT and corresponding class's name in Swing

AWT	Swing
Button	JButton
Label	JLabel
TextField	JTextField
TextArea	JTextArea
Checkbox	JCheckBox
Choice	JComboBox
List	JList
ScrollBar	JScrollBar
Panel	JPanel
Canvas	
MenuBar	JMenuBar
Menu	JMenu
MenuItem	JMenuItem

- The main differences between Swing and AWT:
 - AWT uses native components provided by the operating system (heavyweight), whereas Swing is entirely written in Java and rendered on a lightweight layer.
 - Swing offers many extended components that AWT does not, such as JTable, JTree, JTabbedPane, and more.

2. Organizing Swing components with Layout Managers

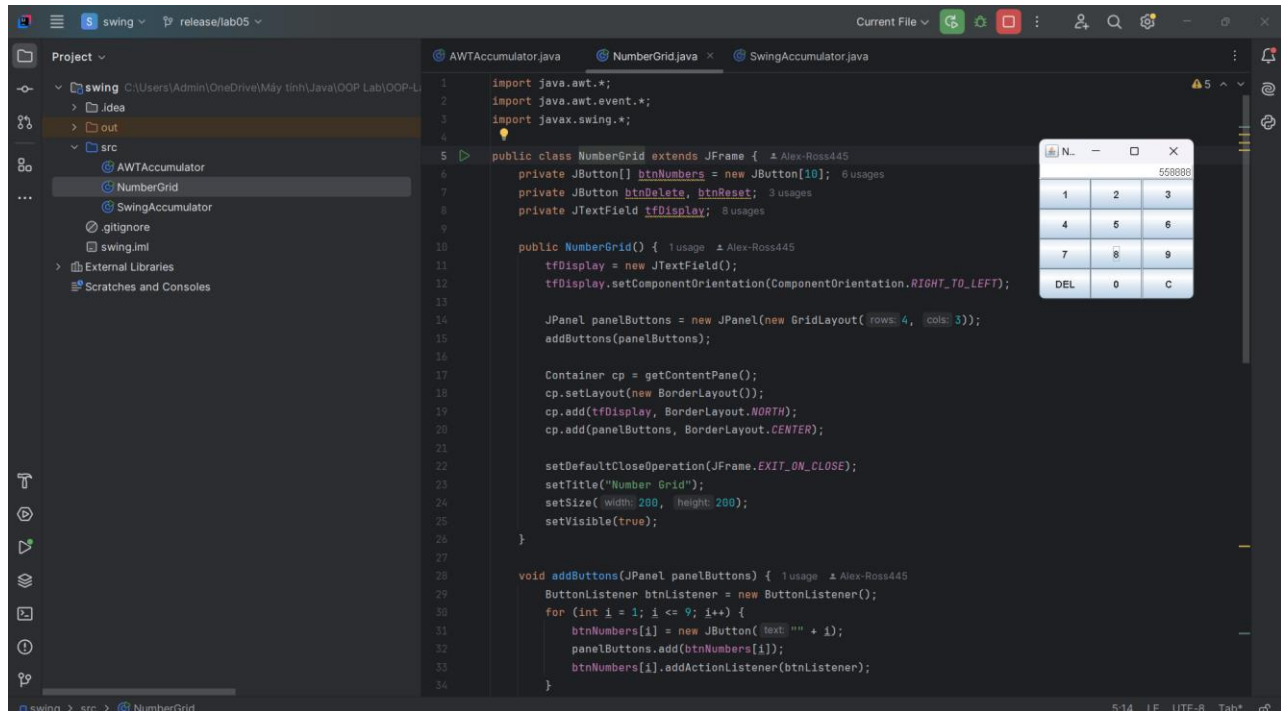


Figure 3: Mã nguồn và giao diện lớp NumberGrid

3. Create a graphical user interface for AIMS with Swing

3.1. Lớp StoreScreen

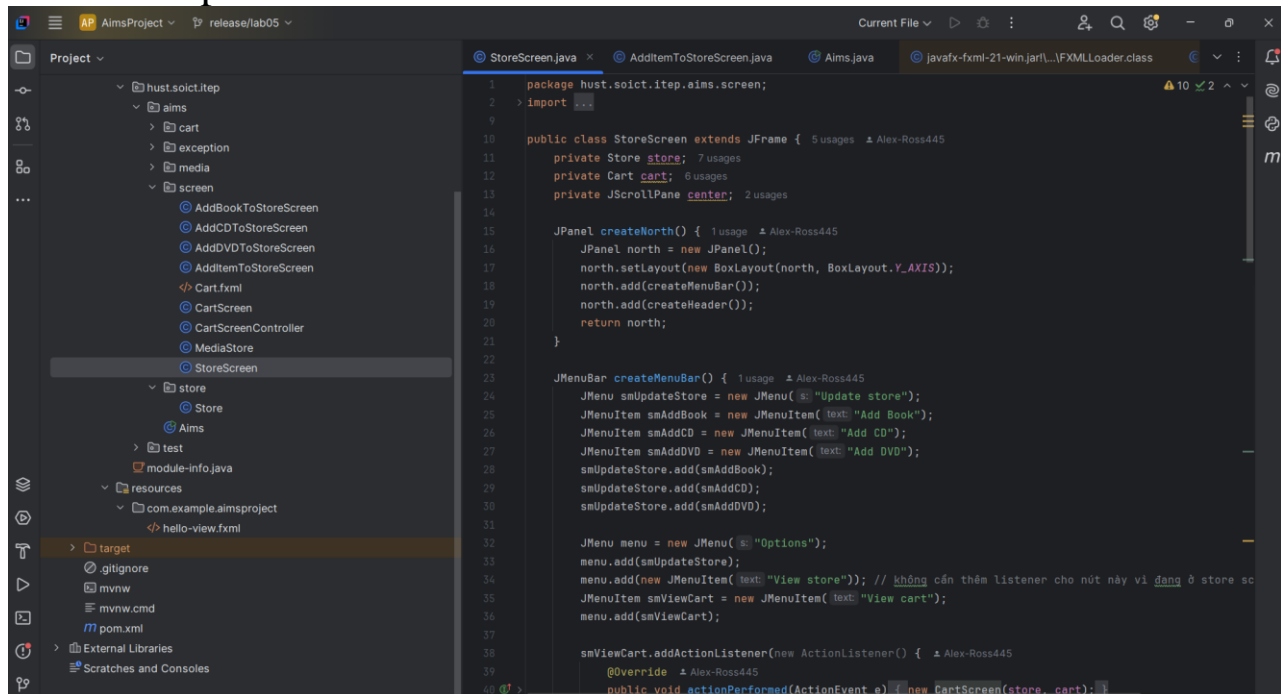


Figure 4: Mã nguồn lớp StoreScreen

3.2. Lớp MediaStore

Thực hành lập trình hướng đối tượng

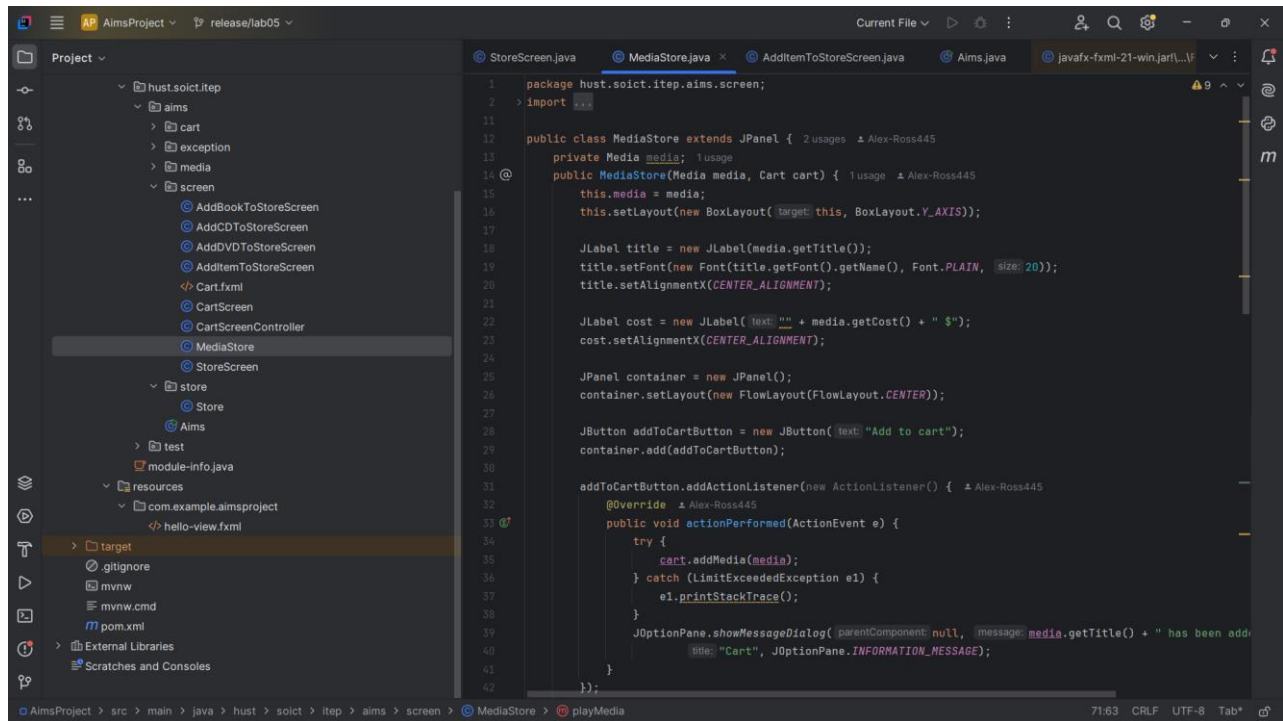


Figure 5: Mã nguồn lớp MediaStore

3.3. Chức năng của nút Play và nút Add to cart

3.3.1. Nút Play

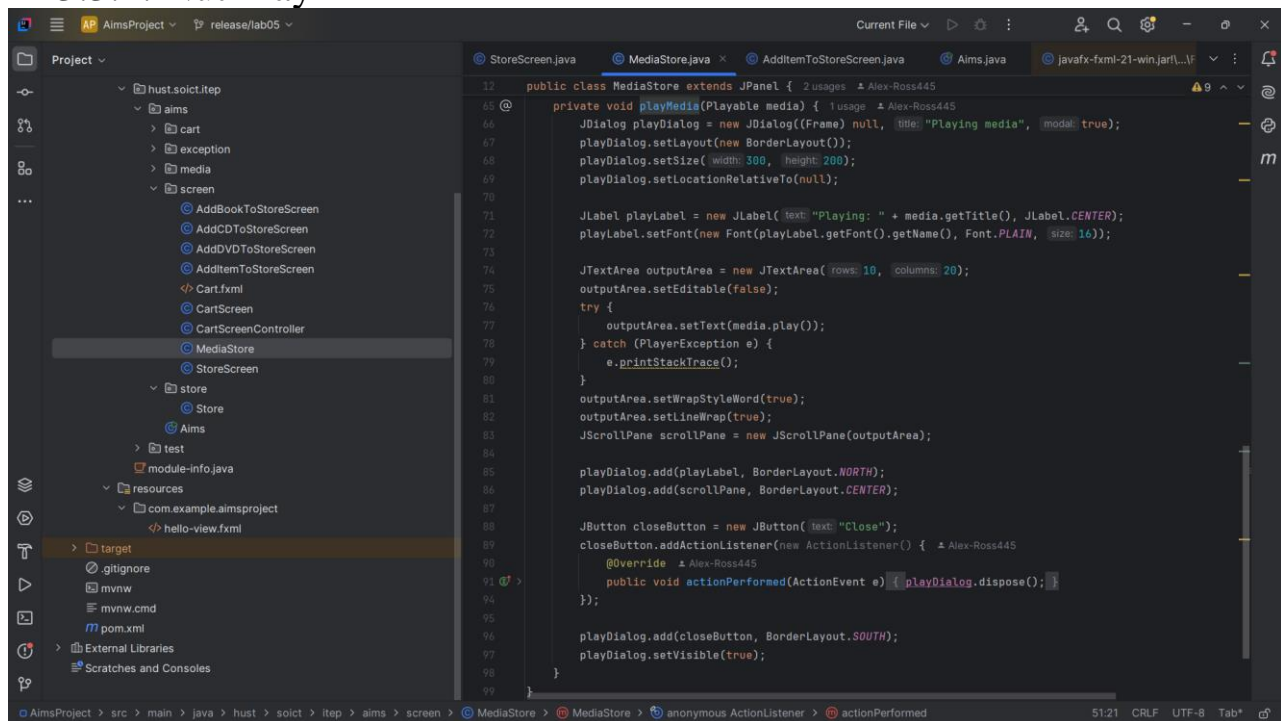


Figure 6: Mã nguồn nút Play của MediaStore

3.3.2. Nút Add to cart

Thực hành lập trình hướng đối tượng


```

JButton addToCartButton = new JButton( text: "Add to cart");
container.add(addToCartButton);

addToCartButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            cart.addMedia(media);
        } catch (LimitExceededException e1) {
            e1.printStackTrace();
        }
        JOptionPane.showMessageDialog( parentComponent: null, message: media.getTitle() + " has been added to cart", title: "Cart", JOptionPane.INFORMATION_MESSAGE);
    }
});

```

Figure 7: Mã nguồn nút Add to cart của MediaStore

3.4. Giao diện

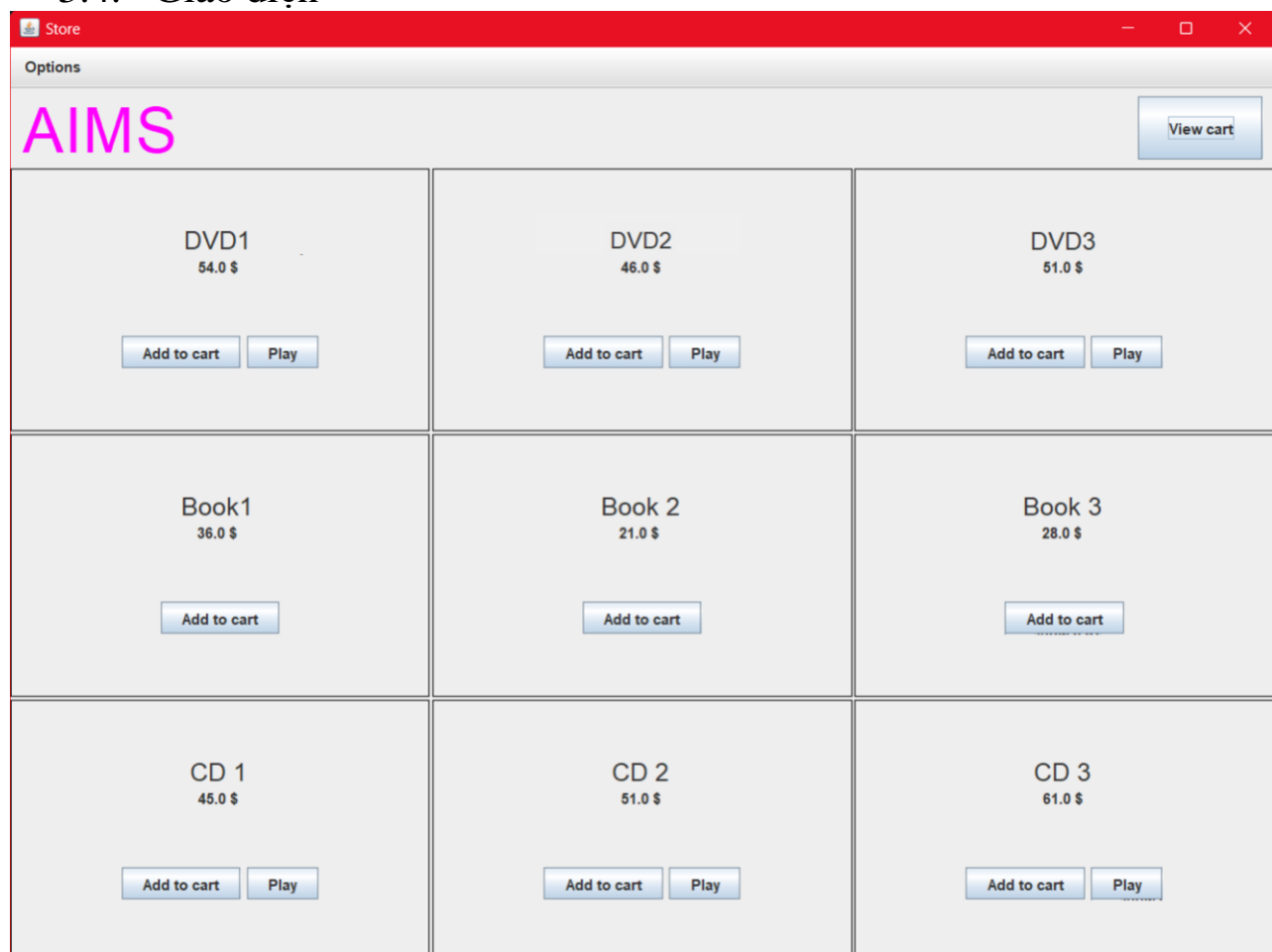


Figure 8: Giao diện Store

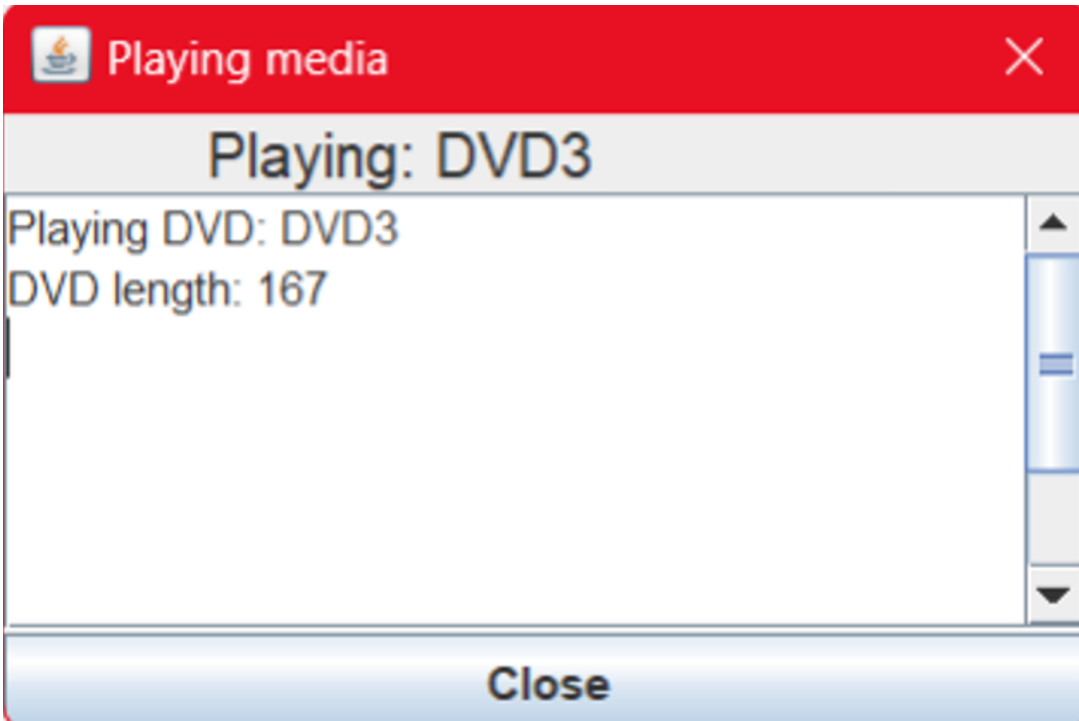


Figure 9: Giao diện Play DVD media trong

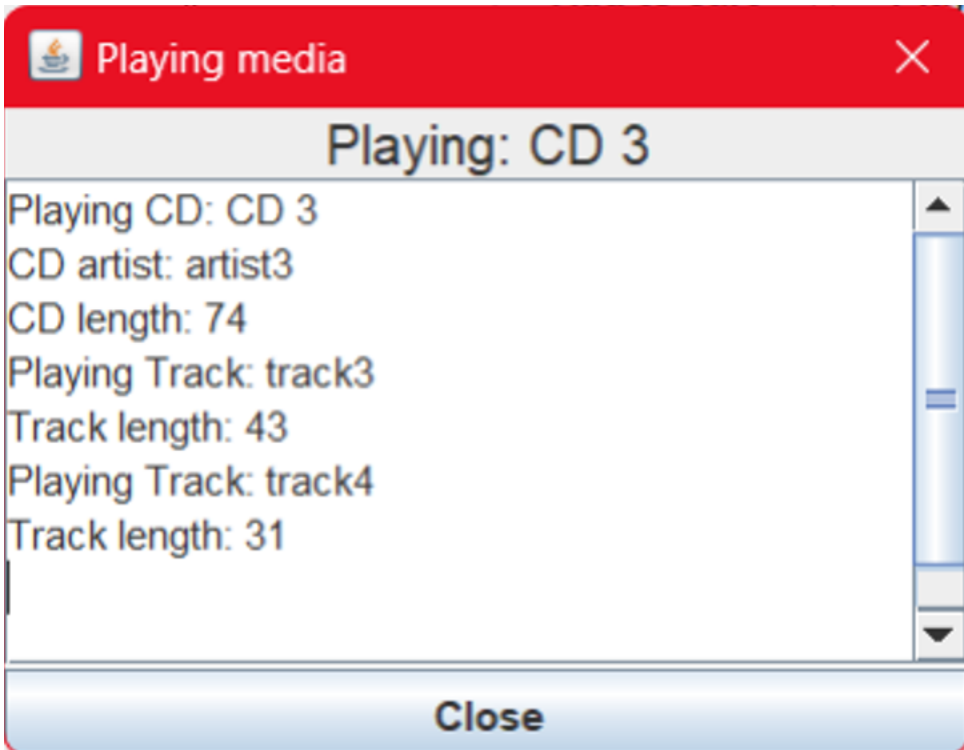


Figure 10: Giao diện Play CD media trong Store

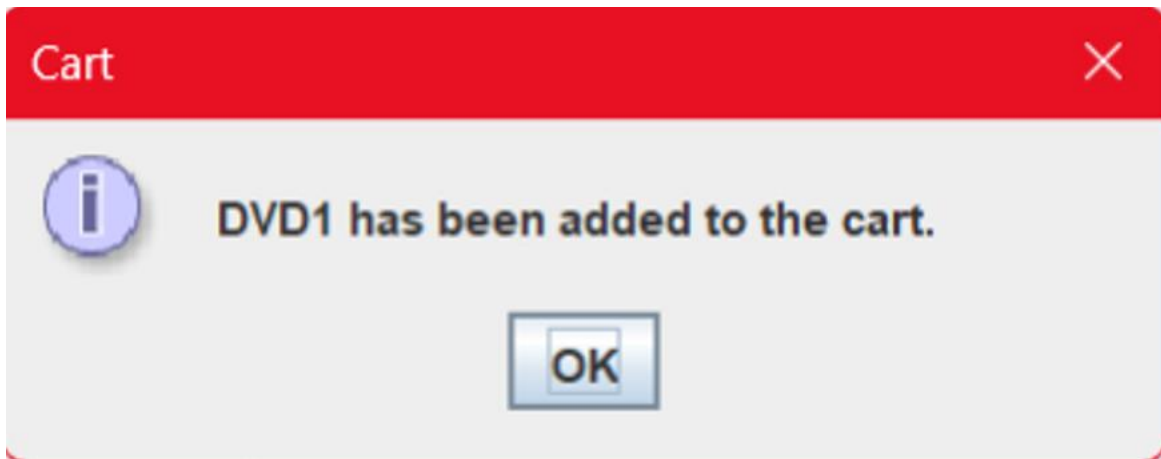


Figure 11: Thông báo thêm media vào cart

4. JavaFX API

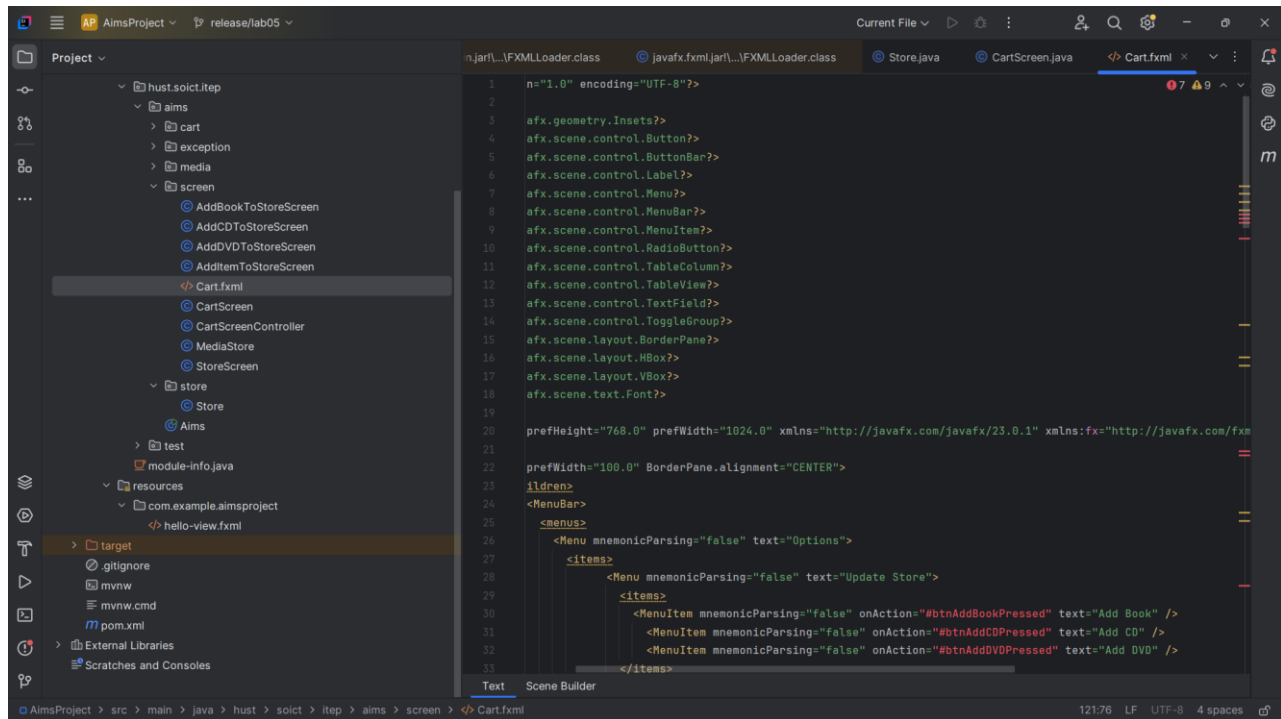


Figure 12: Mã nguồn Painter.fxml

4.2. Painter.java

Thực hành lập trình hướng đối tượng

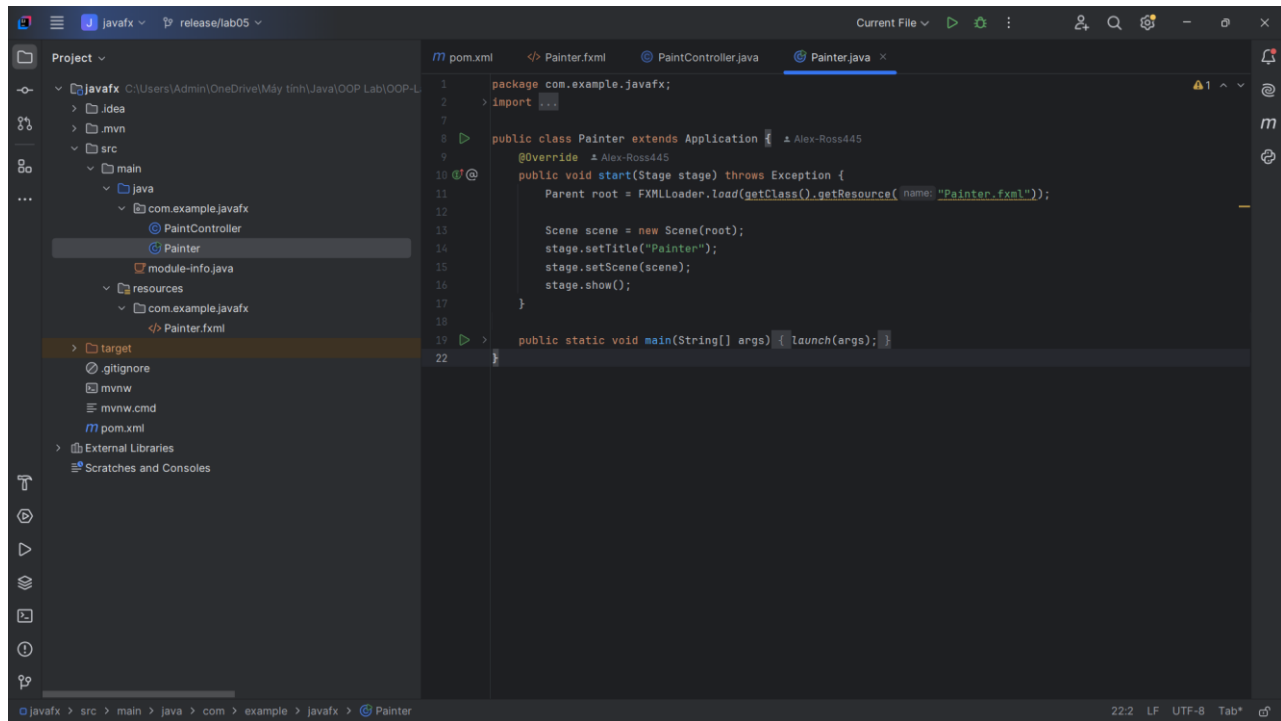


Figure 13: Mã nguồn Painter.java

4.3. PaintController.java

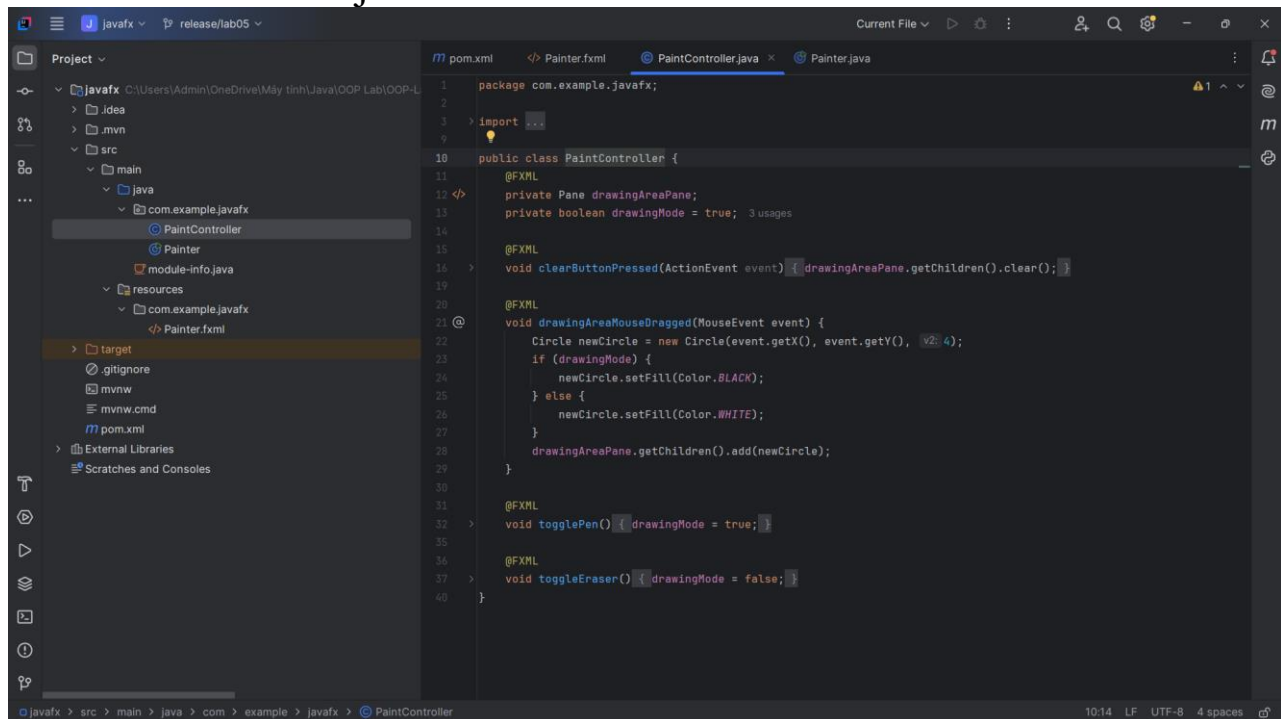


Figure 14: Mã nguồn PaintController.java

4.4. Giao diện

Thực hành lập trình hướng đối tượng

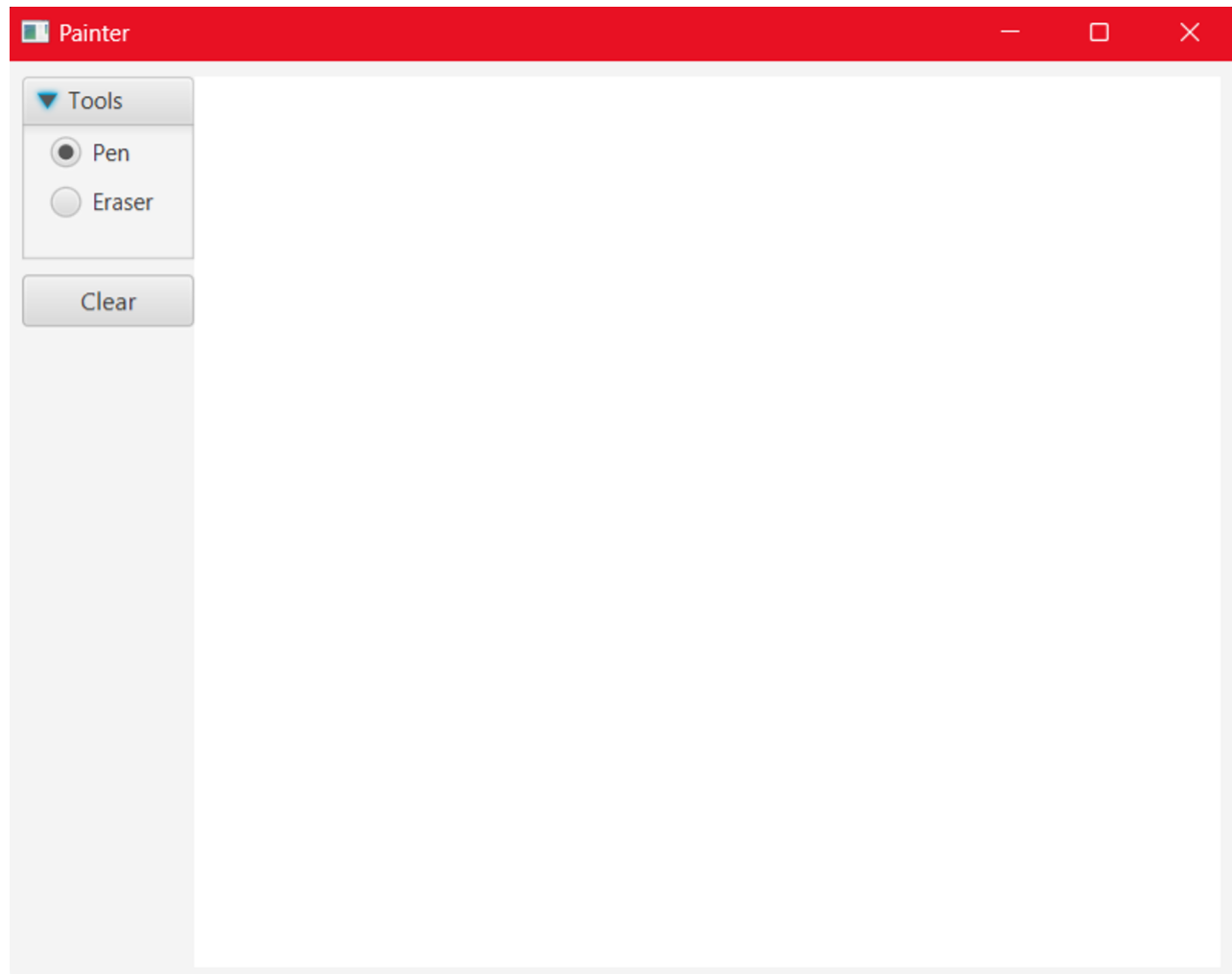


Figure 15: Giao diện Painter

5. Setting up the View Cart Screen with ScreenBuilder

5.1. Cart.fxml

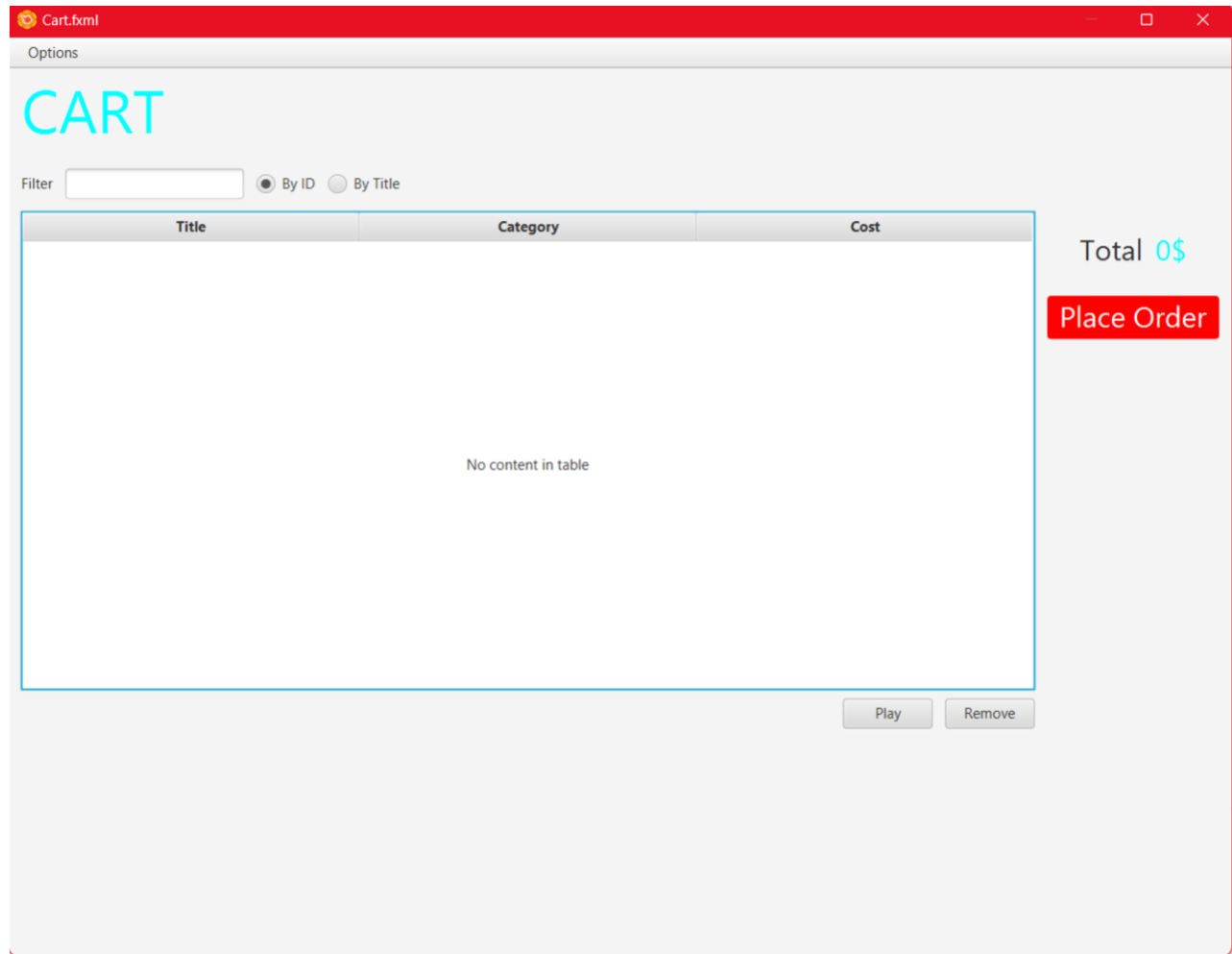


Figure 17: Giao diện Cart Screen

6. Integrating JavaFX into Swing application – The JFXPanel class

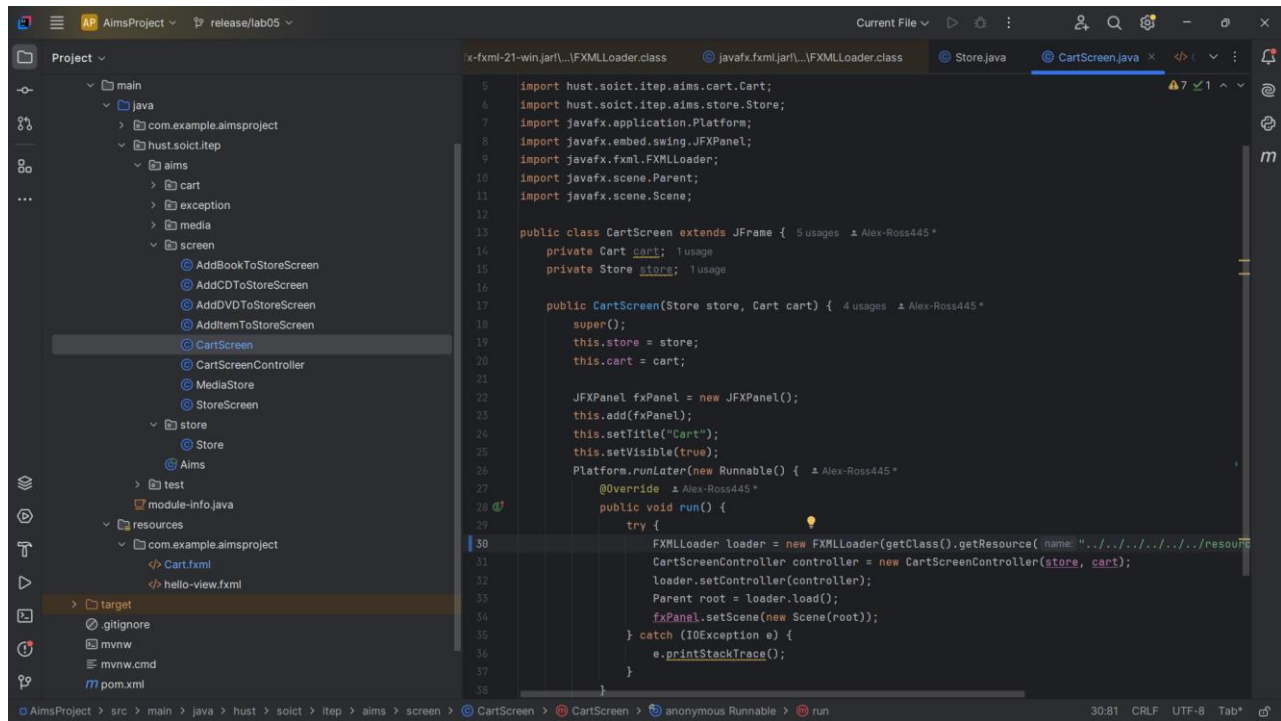


Figure 18: Mã nguồn Cart Screen (nhúng JavaFX vào Swing)

7. View the items in cart – JavaFX’s data-driven UI

7.1. CartScreenController.java

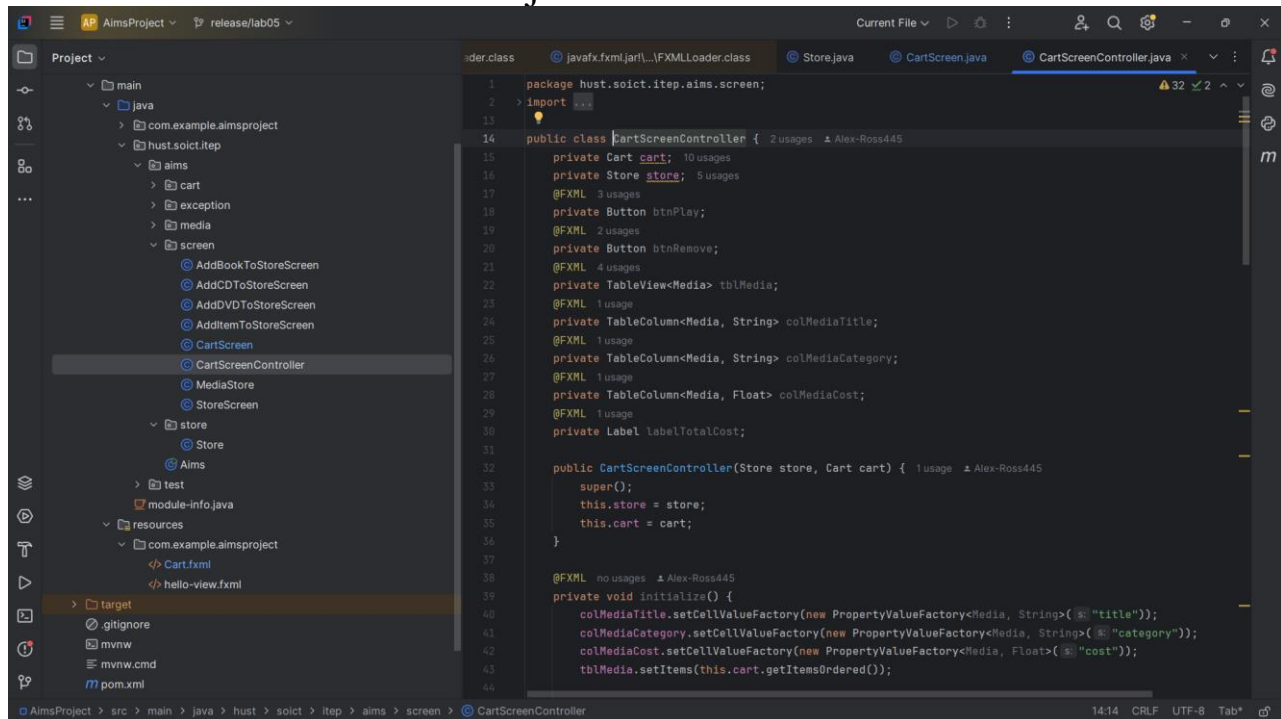


Figure 19: Mã nguồn CartScreenController.java

7.2. Sửa lại Cart.java

Thực hành lập trình hướng đối tượng

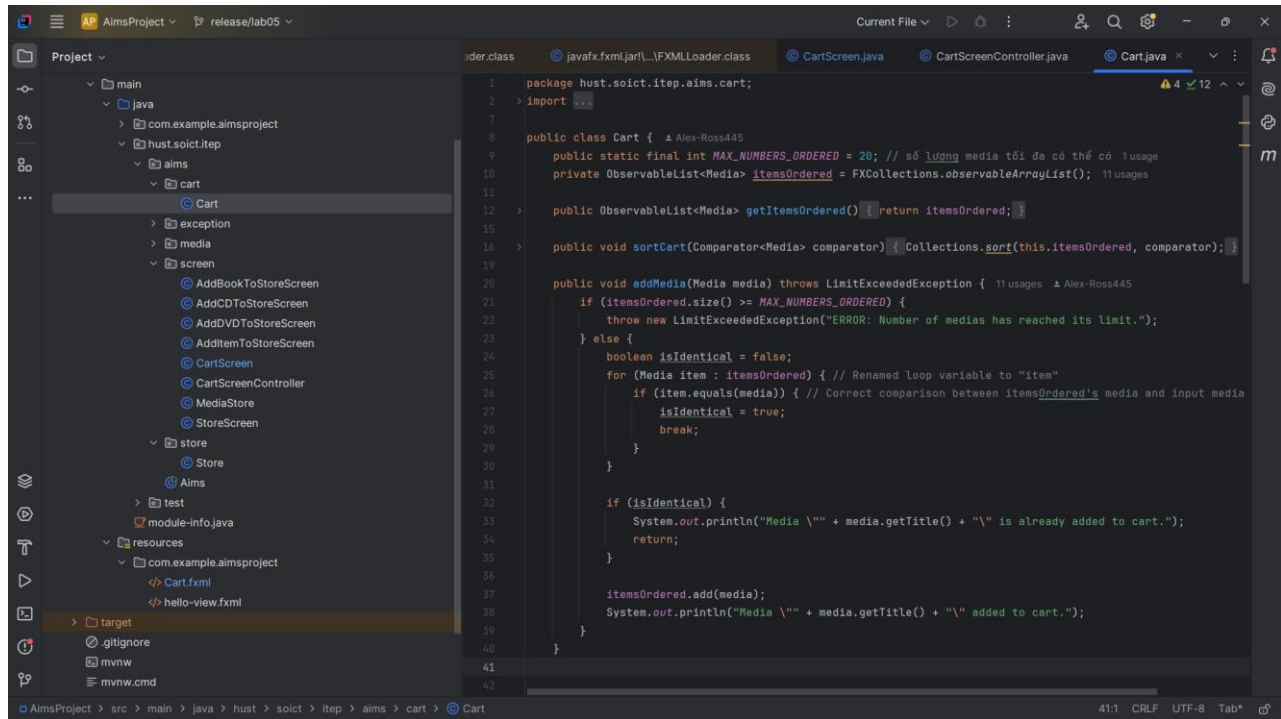


Figure 20: Chỉnh sửa Cart.java để phù hợp với JavaFX

7.3. Giao diện CartScreen

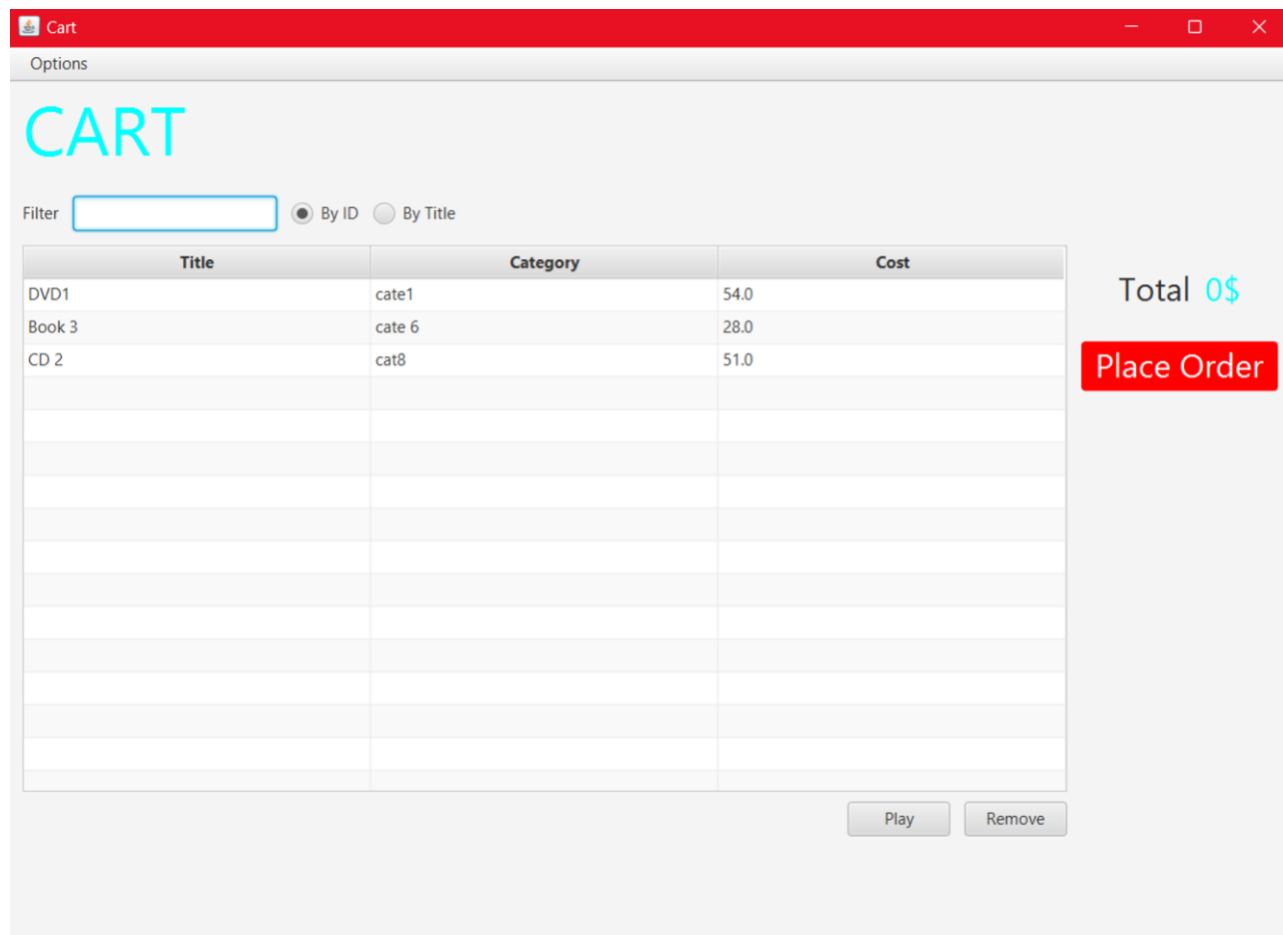


Figure 21: Giao diện Cart (hiển thị sản phẩm trong cart)

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

8.1. Sửa lại CartScreenController.java

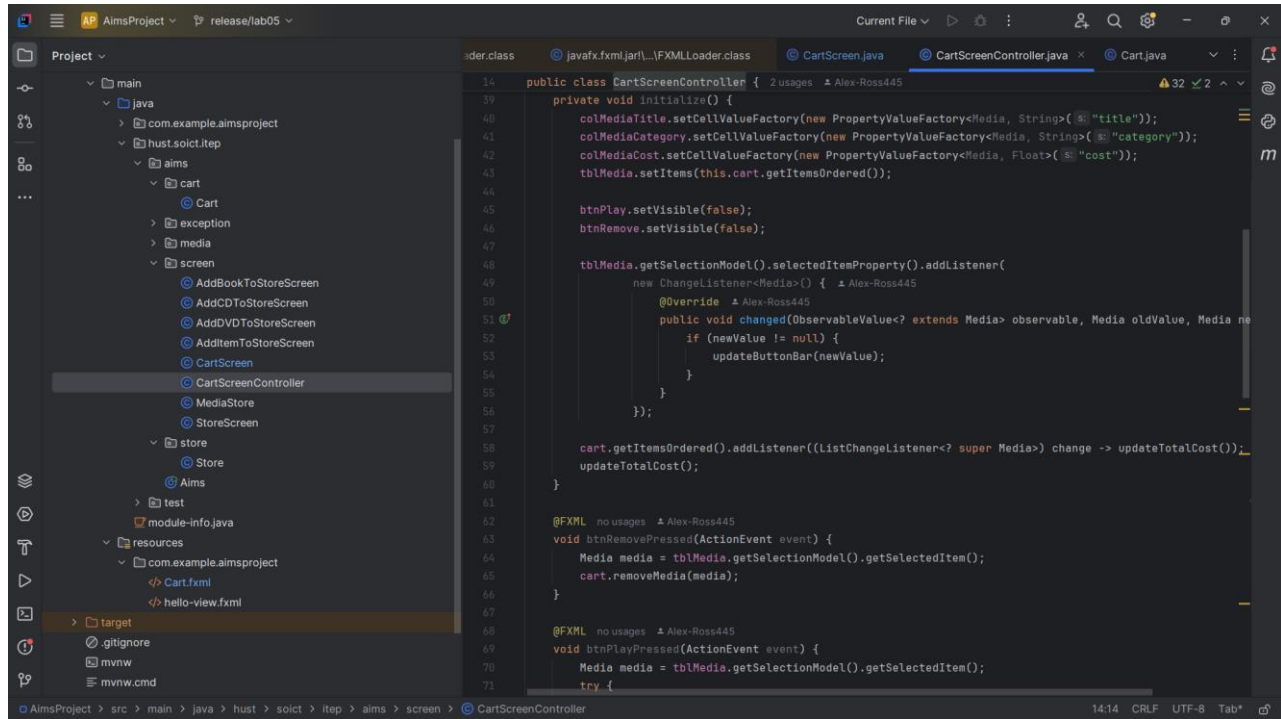


Figure 22: Chỉnh sửa CartScreenController.java

8.2. Giao diện CartScreen

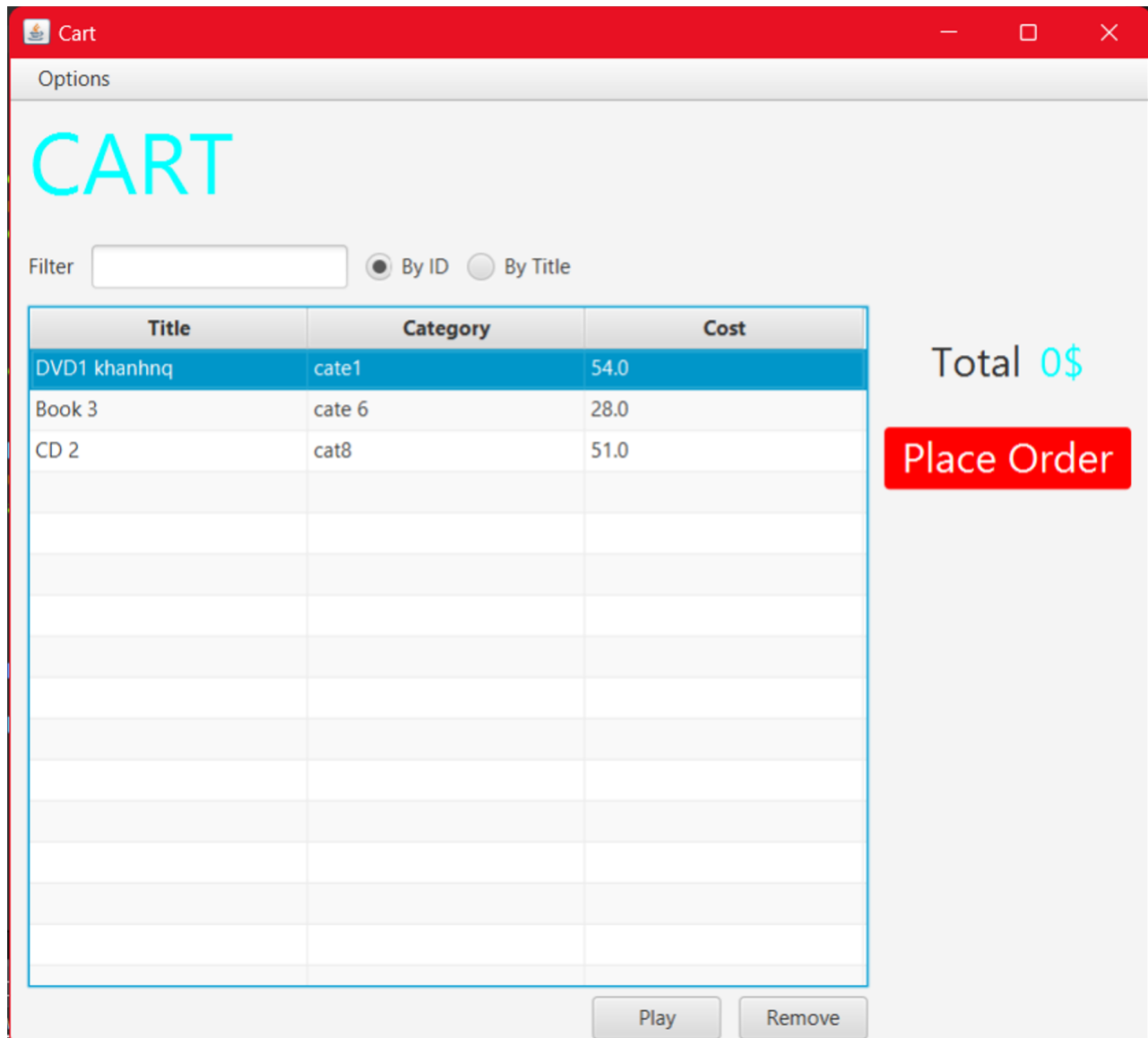


Figure 23: Giao diện Cart (cập nhật nút Play media)

9. Deleting a media

9.1. Thêm phương thức khi ấn nút Remove

```
<buttons>
  <Button fx:id="btnPlay" mnemonicParsing="false" onAction="#btnPlayPressed" text="Play" />
  <Button fx:id="btnRemove" mnemonicParsing="false" onAction="#btnRemovePressed" text="Remove" />
</buttons>
```

Figure 24: Chỉnh sửa Cart.fxml cho chức năng xóa sản phẩm khỏi cart

```
@FXML
void btnRemovePressed(ActionEvent event) {
    Media media = tblMedia.getSelectionModel().getSelectedItem();
    cart.removeMedia(media);
}
```

Figure 25: Thêm phương thức bắt sự kiện nút Remove được click

9.2. Giao diện

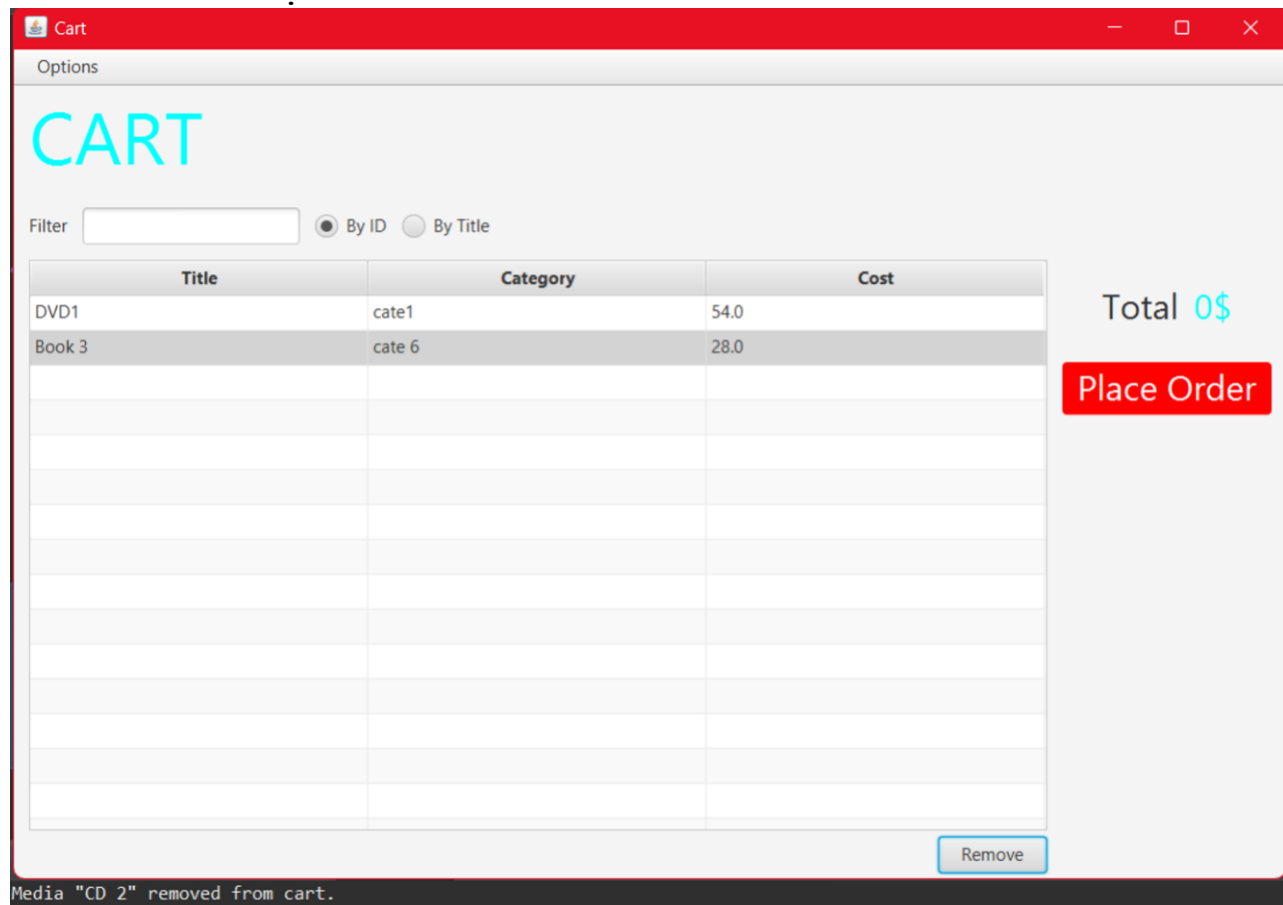


Figure 26: Giao diện Cart sau khi xóa CD2 khỏi cart

10. Filter items in cart – FilteredList (optional)

11. Complete the Aims GUI application

11.1. Cart Screen

11.1.1. “Place order” button

```
@FXML
void btnPlaceOrderPressed(ActionEvent event) {
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("Thông báo ()");
    alert.setHeaderText(null);
    alert.setContentText("Đơn hàng của bạn đã được tạo thành công.");
    alert.showAndWait();

    cart.getItemsOrdered().clear();
}
```

Figure 27: Thêm phương thức bắt sự kiện nút Place order được click



Figure 28: Thông báo đơn hàng đã được tạo thành công

11.1.2. “Play” button

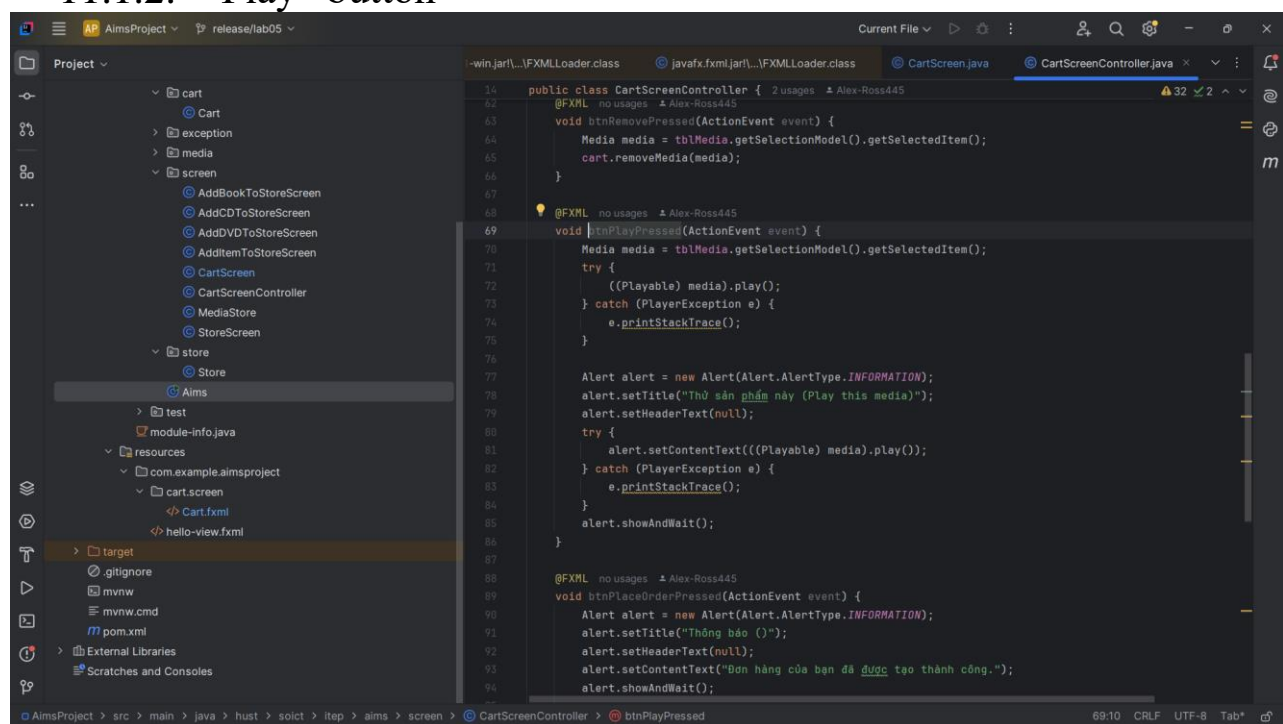


Figure 29: Thêm phương thức bắt sự kiện nút Play được click

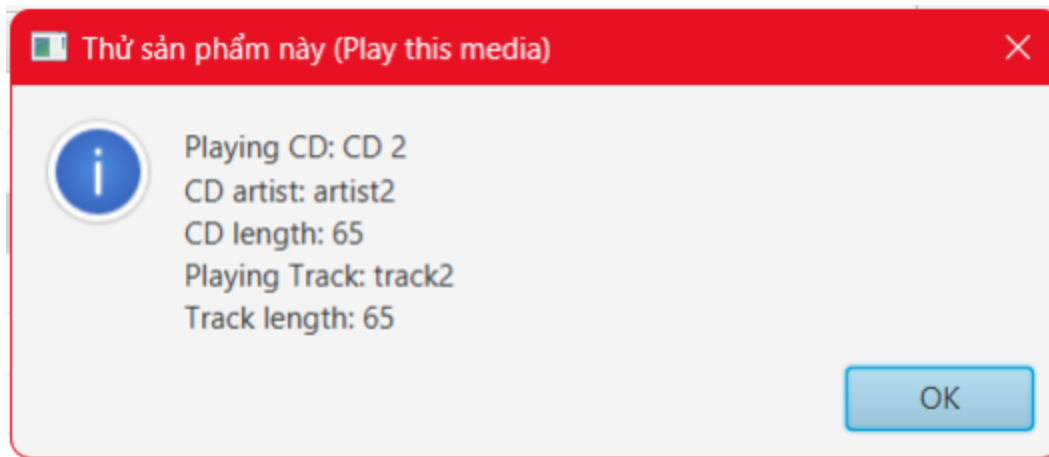


Figure 30: Hộp thoại Play media trong Cart

11.1.3. “Total cost” label

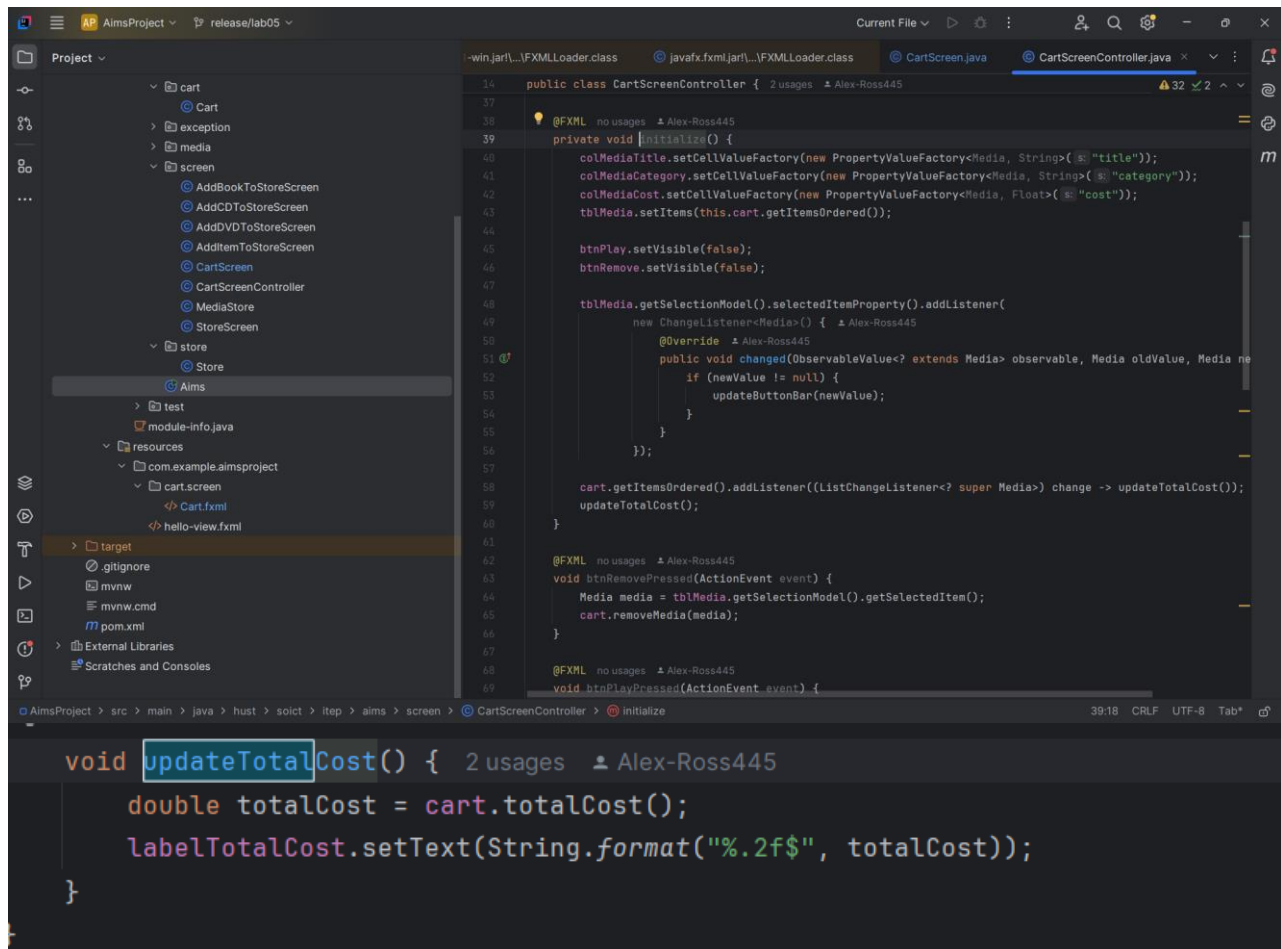


Figure 31: Mã nguồn cập nhật tổng giá tiền của đơn hàng

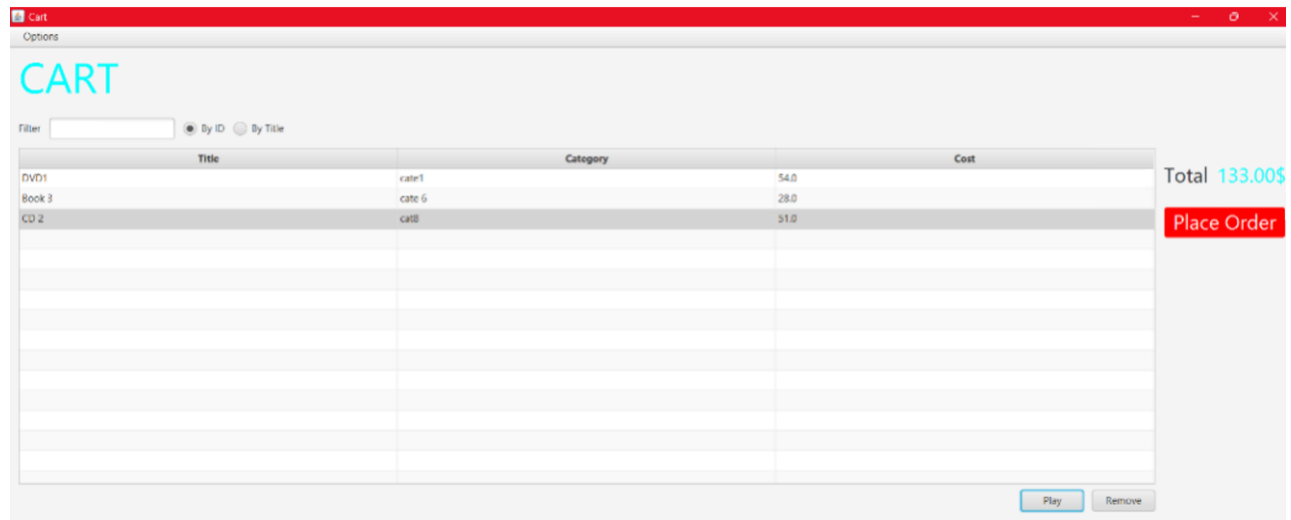


Figure 32: Giao diện Cart (hiển thị tổng giá tiền của đơn hàng)

11.1.4. Menu bar

```
@FXML no usages Alex-Ross445
void menuItemViewStorePressed(ActionEvent event) { new StoreScreen(store, cart); }

@FXML no usages Alex-Ross445
void btnAddBookPressed(ActionEvent event) { new AddBookToStoreScreen(store, cart); }

@FXML no usages Alex-Ross445
void btnAddCDPressed(ActionEvent event) { new AddCDToStoreScreen(store, cart); }

@FXML no usages Alex-Ross445
void btnAddDVDPressed(ActionEvent event) { new AddDVDToStoreScreen(store, cart); }
```

Figure 33: Mã nguồn MenuBar

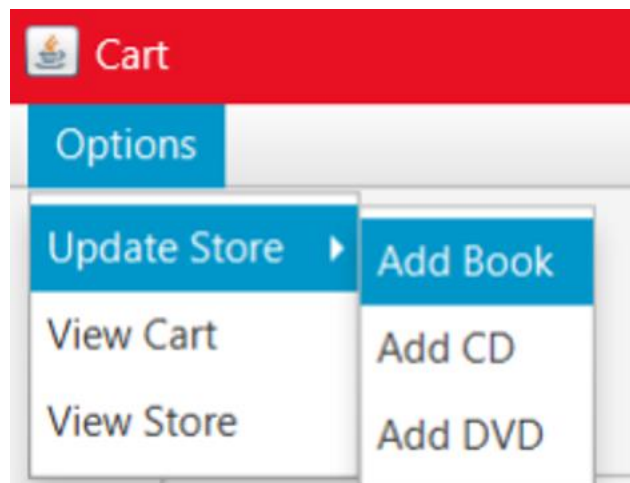


Figure 34: Giao diện MenuBar

11.2. Store Screen

Thực hành lập trình hướng đối tượng


```

addToCartButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            cart.addMedia(media);
        } catch (LimitExceededException e1) {
            e1.printStackTrace();
        }
        JOptionPane.showMessageDialog(parentComponent: null, message: media.getTitle() + " has been added to the cart", title: "Cart", JOptionPane.INFORMATION_MESSAGE);
    }
});

```

Figure 35: Mã nguồn thêm sản phẩm vào cart



Figure 36: Thông báo sản phẩm được thêm vào cart

11.3. Update Store Screen

```

package hust.soict.itcp.aims.screen;
import javax.swing.*;

public class AddItemToStoreScreen extends JFrame {
    protected Store store;
    protected Cart cart;

    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 smUpdateStore = new JMenu("Update store");
        JMenuItem smAddBook = new JMenuItem("Add Book");
        JMenuItem smAddCD = new JMenuItem("Add CD");
        JMenuItem smAddDVD = new JMenuItem("Add DVD");
        smUpdateStore.add(smAddBook);
        smUpdateStore.add(smAddCD);
        smUpdateStore.add(smAddDVD);

        JMenu menu = new JMenu("Options");
        menu.add(smUpdateStore);
        JMenuItem smViewStore = new JMenuItem("View store");
        menu.add(smViewStore);
        JMenuItem smViewCart = new JMenuItem("View cart");
        menu.add(smViewCart);

        smAddBook.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                new AddBookToStoreScreen(store, cart);
            }
        });
    }
}

```

Figure 37: Mã nguồn lớp AddItemToStoreScreen

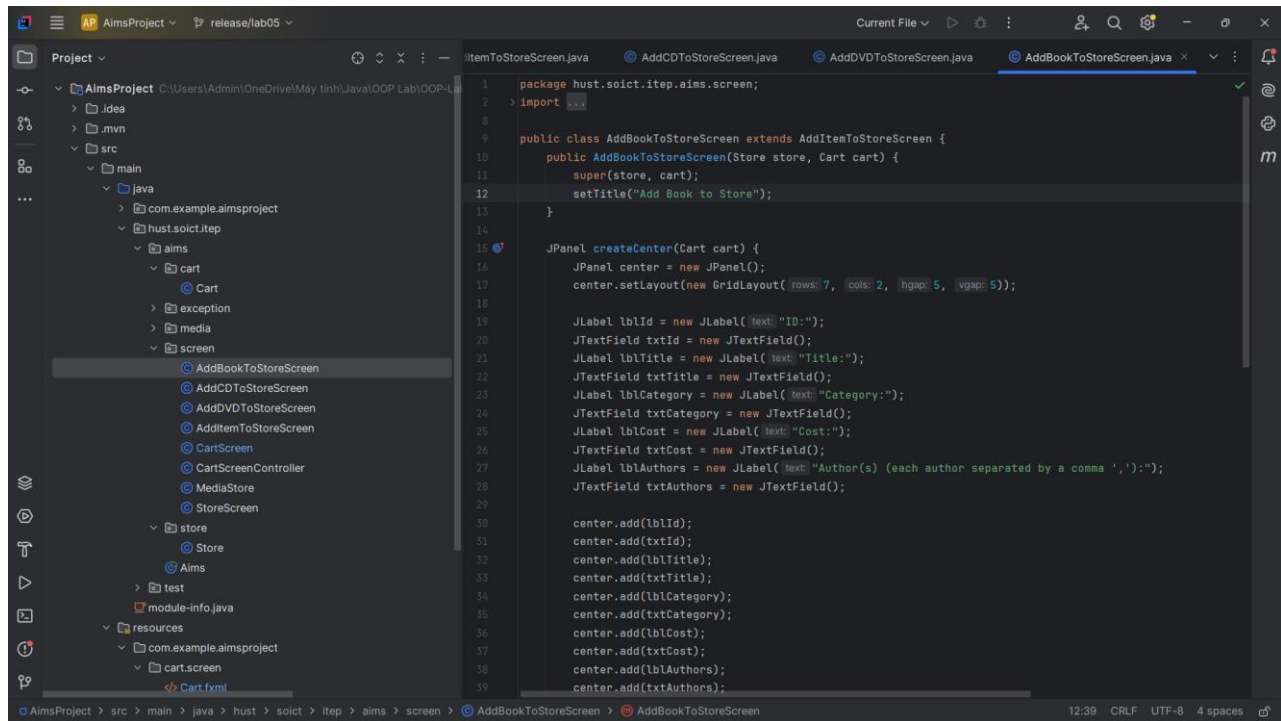


Figure 38: Mã nguồn lớp AddBookToStoreScreen

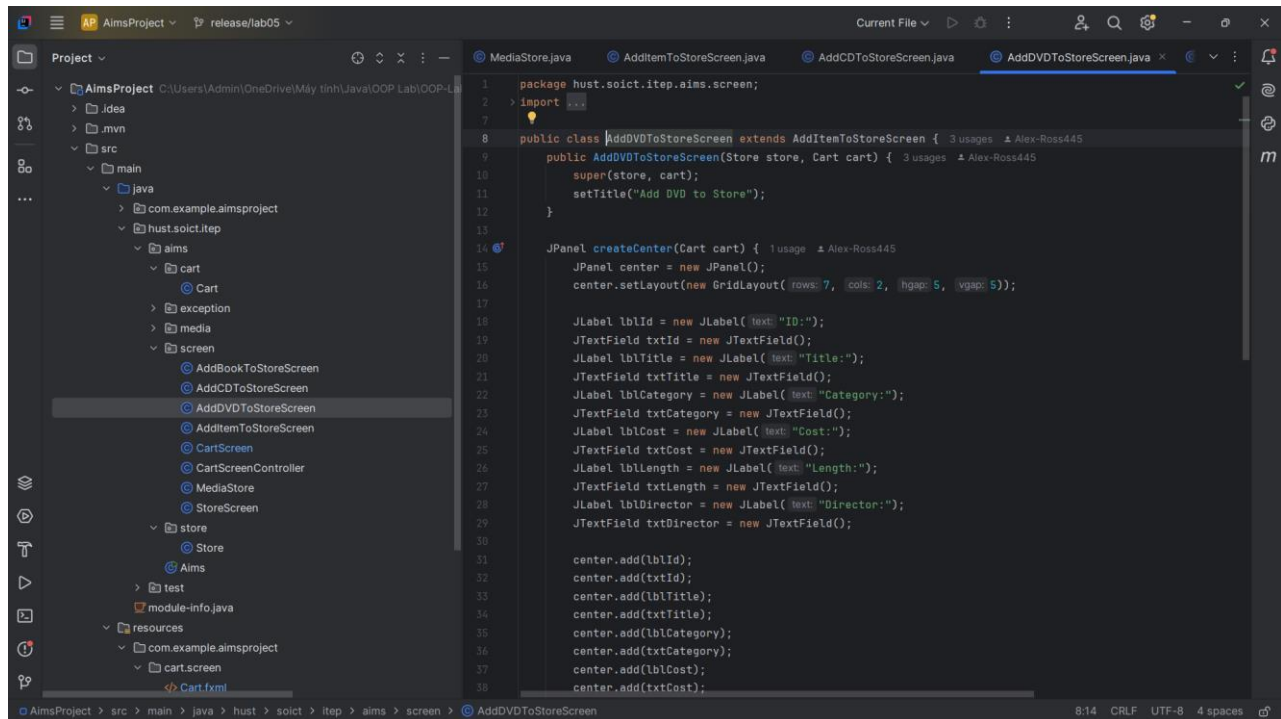


Figure 39: Mã nguồn lớp AddDVDToStoreScreen

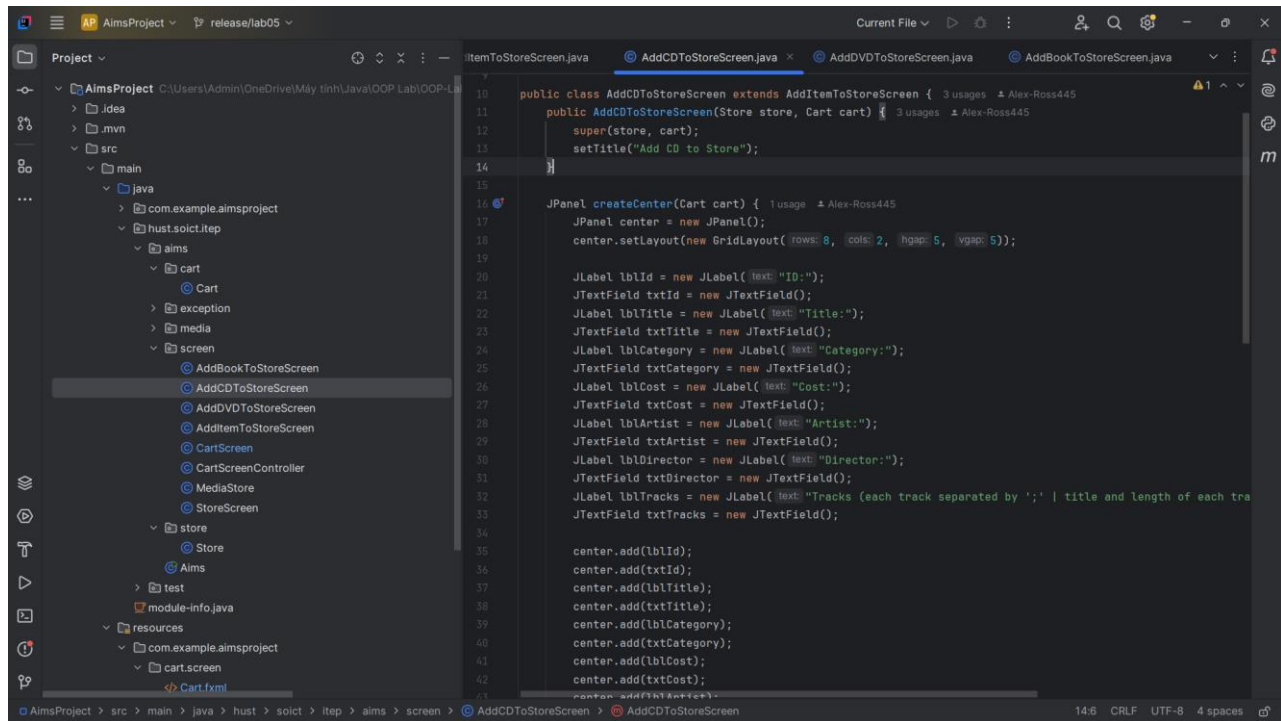


Figure 40: Mã nguồn lớp AddCDToStoreScreen

Add Book to Store

Options

AIMS

ID:

Title:

Category:

Cost:

Author(s) (each author separated by a comma):

Add Book

Figure 41: Giao diện AddBookToStore

Figure 42: Giao diện AddCDToStore

Figure 43: Giao diện AddDVDToStore

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

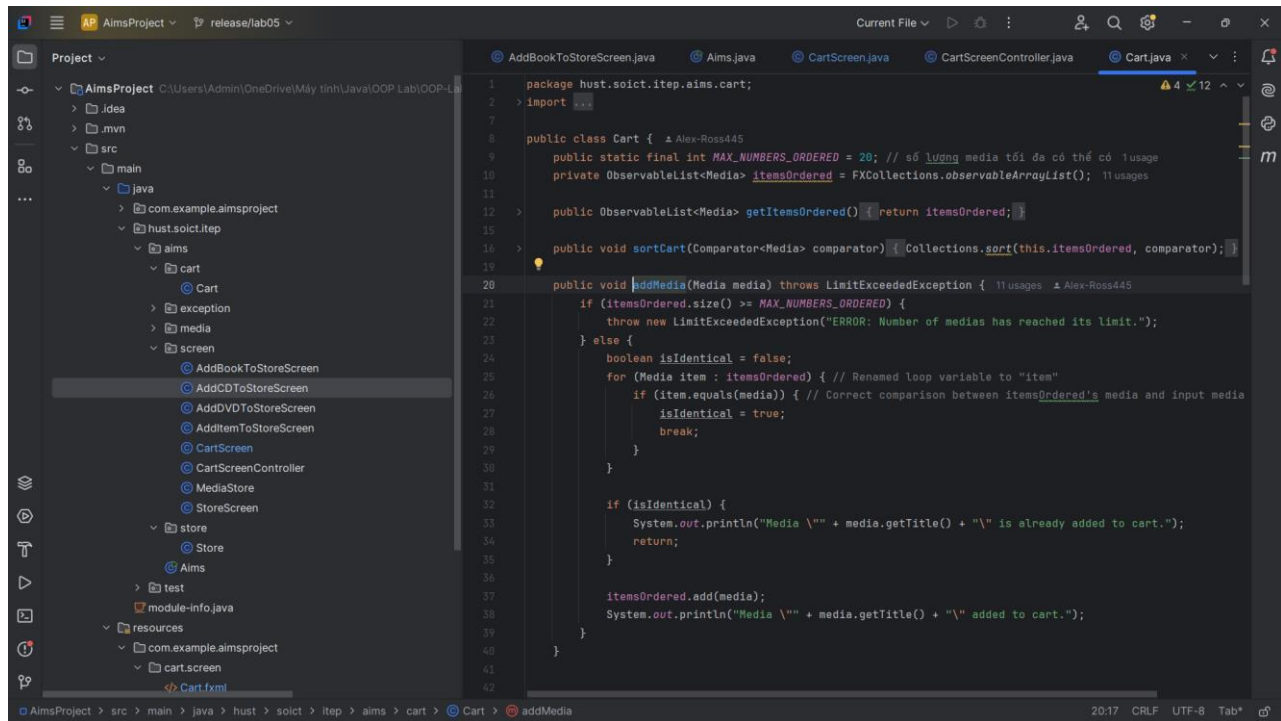


Figure 44: Mã nguồn ngoại lệ vượt quá số lượng sản phẩm tối đa trong cart

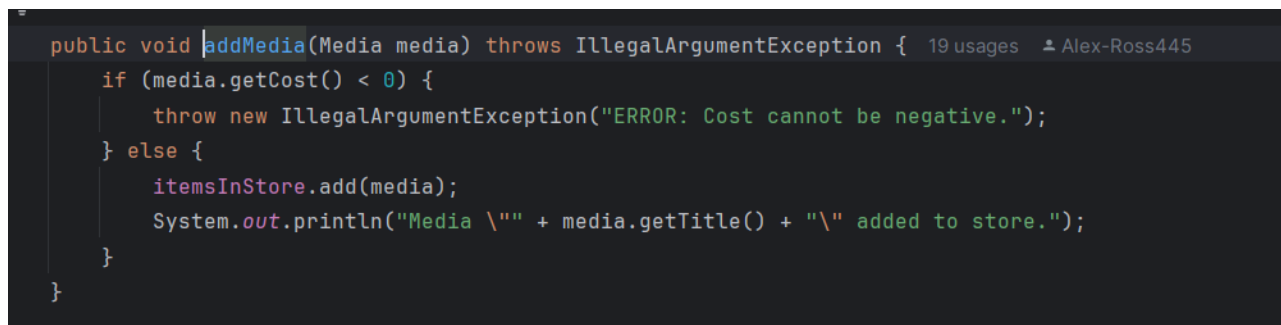


Figure 45: Mã nguồn ngoại lệ giá sản phẩm nhỏ hơn 0

13. Create a class which inherits from Exception

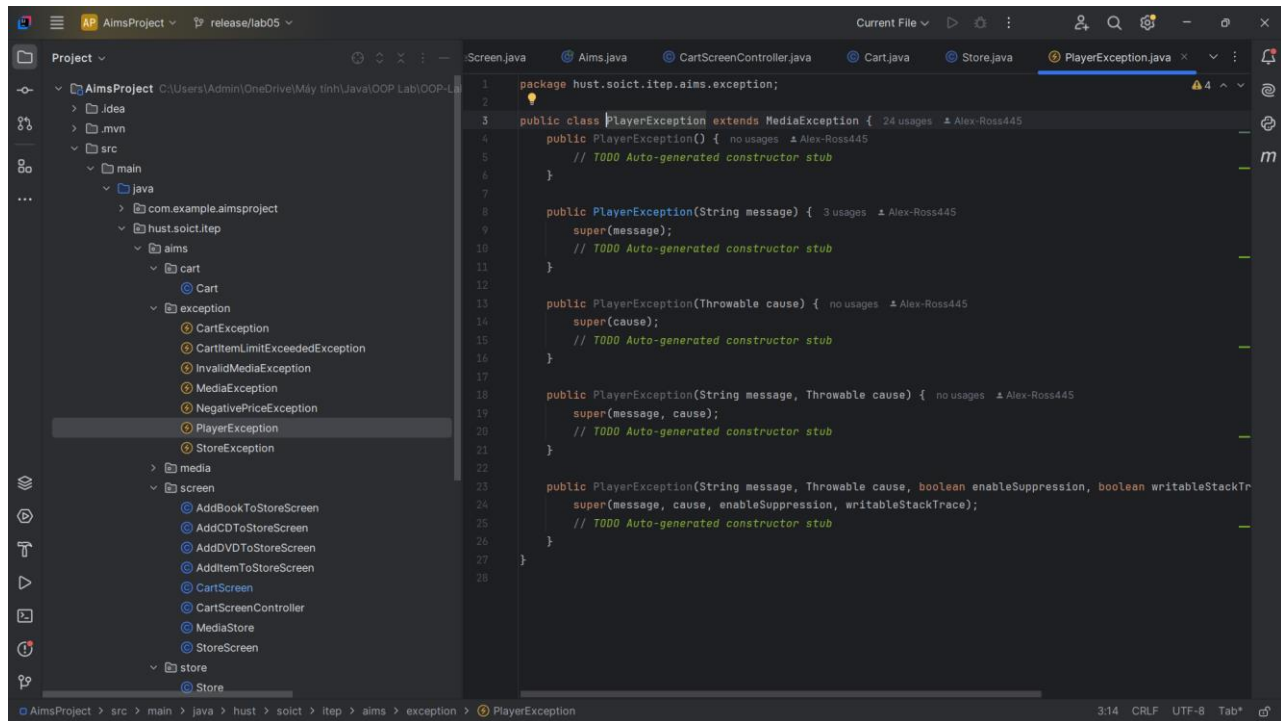


Figure 46: Chỉnh sửa interface Playable

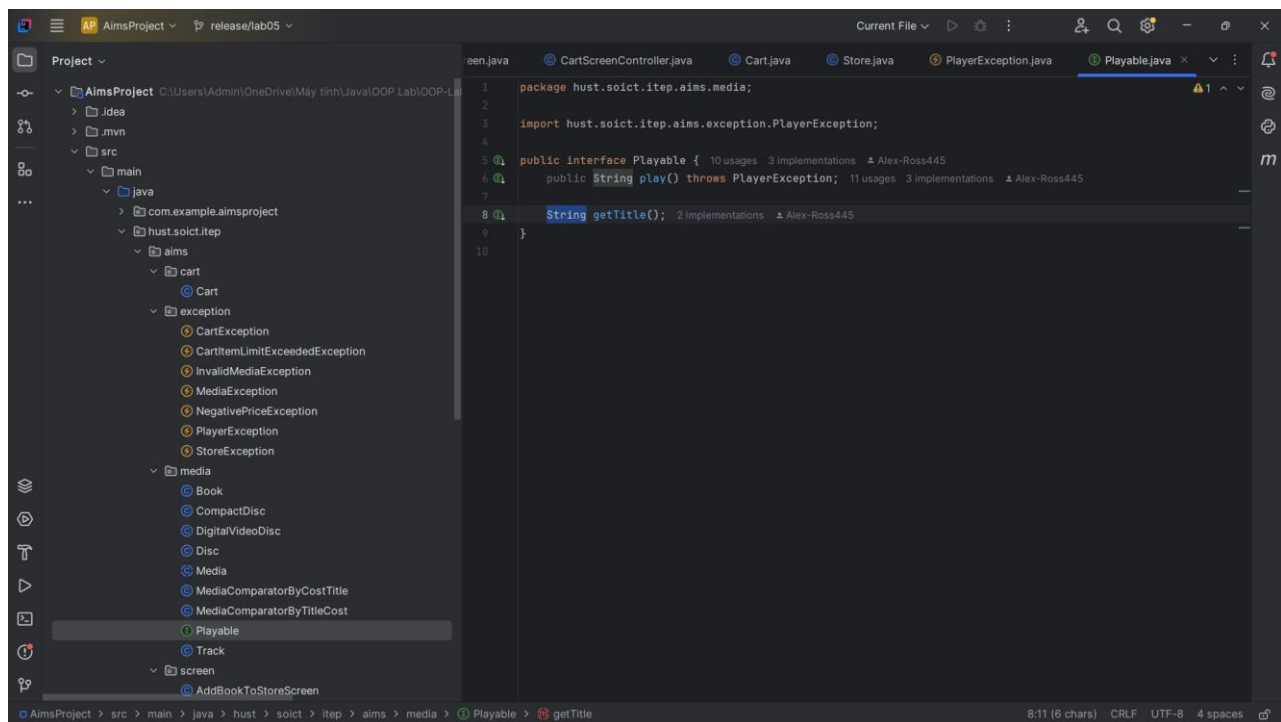


Figure 47: Chỉnh sửa phương thức play() của DigitalVideoDisc

```

public String play() throws PlayerException { 11 usages  Alex-Ross445
    if (this.getLength() > 0) {
        System.out.println("Playing DVD: " + this.getTitle());
        System.out.println("DVD length: " + this.getLength());

        return "Playing DVD: " + this.getTitle() + "\n" +
            "DVD length: " + this.getLength() + "\n";
    } else {
        throw new PlayerException("ERROR: DVD length is non-positive.");
    }
}

```

Figure 48: Chỉnh sửa phương thức play() của Track

```

public String play() throws PlayerException { 11 usages  Alex-Ross445
    if (this.getLength() > 0) {
        System.out.println("Playing CD: " + this.getTitle()); // thông tin về CD
        System.out.println("CD artist: " + this.getArtist());
        System.out.println("CD length: " + this.getLength());
        for (Track track : tracks) {
            try {
                track.play(); // play từng track trong CD
            } catch (PlayerException e) {
                throw e;
            }
        }
    }
}

```

Figure 49: Chỉnh sửa phương thức play() của CompactDisc

14. Update the Aims class

14.1. Answer the question

Question: What happens if the Aims class is not updated to handle exceptions when the play() method is called?

Answer: The exception will propagate through the call stack until it encounters a try-catch block or reaches the main() method. If the main() method does not catch the exception, the program will terminate immediately, and the console will display the stack trace. This makes the program less robust and harder to use.

14.2. Update the Aims.java code


```

    try {
        cart.addMedia(dvd1);
    } catch (LimitExceededException e) {
        e.printStackTrace();
    }
    try {
        cart.addMedia(book3);
    } catch (LimitExceededException e) {
        e.printStackTrace();
    }
    try {
        cart.addMedia(cd2);
    } catch (LimitExceededException e) {
        e.printStackTrace();
    }

    if (media instanceof DigitalVideoDisc) {
        DigitalVideoDisc mediadvd = (DigitalVideoDisc) media;
        try {
            mediadvd.play();
        } catch (PlayerException e) {
            e.printStackTrace();
        }
    } else if (media instanceof CompactDisc) {
        CompactDisc mediacd = (CompactDisc) media;
        try {
            mediacd.play();
        } catch (PlayerException e) {
            e.printStackTrace();
        }
    } else {
        System.out.println("Media \"" + media.getTitle() + "\" is unplayable.");
    }
}

```

Figure 50: Chỉnh sửa lớp Aims

15. Modify the equals() method of Media class


```

@Override
public boolean equals(Object o) {
    try {
        if (this == o) return true;

        if (o == null || getClass() != o.getClass()) return false;

        Media media = (Media) o;
        return this.title != null && this.title.equals(media.title);
    } catch (NullPointerException e) {
        System.err.println("NullPointerException: Title của đối tượng hoặc đối tượng là null.");
        return false;
    } catch (ClassCastException e) {
        System.err.println("ClassCastException: Không thể ép kiểu đối tượng so sánh.");
        return false;
    }
}

```

Figure 51: Chỉnh sửa phương thức equals() của lớp Media

16. Update Aims class diagram

The inheritance of exception classes is clearly demonstrated in the source code of the package hust.soict.dsai.aims.exception, as shown in the diagram below.

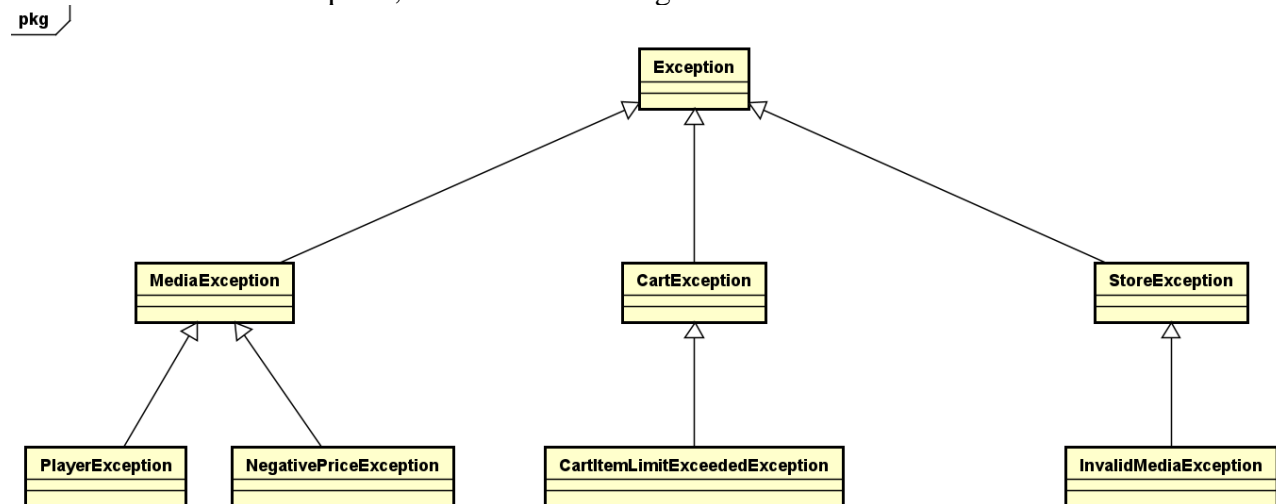


Figure 52: Cây phân cấp ngoại lệ