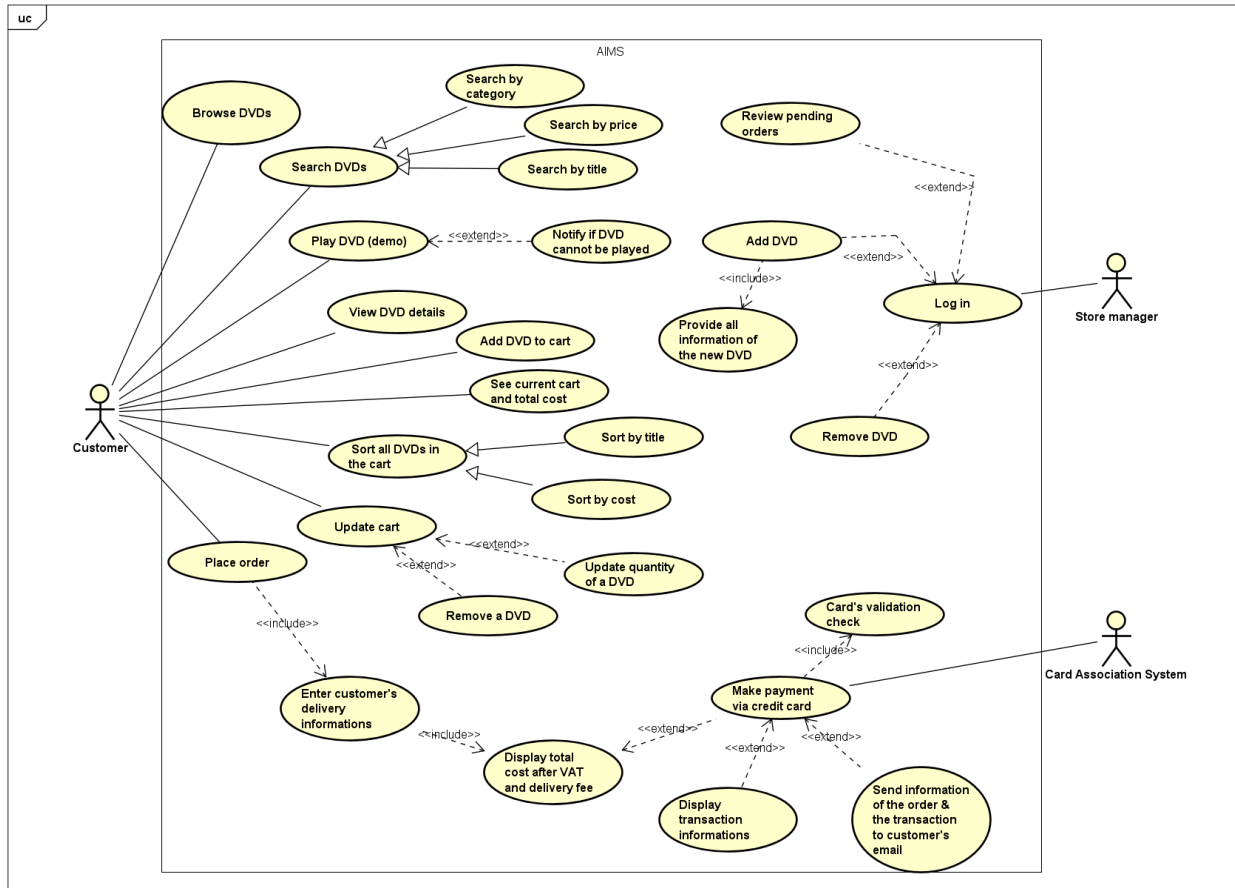


```

classDiagram
    class Media {
        +id int
        +title String
        +category String
        +cost float
        +MediaId() int title String category String cost float
        +getId() int
        +setId(int) void
        +getTitle() String
        +getCategory() String
        +getCost() float
        +setCategory(category String) void
        +getCost() float
        +setCost(cost float) void
        +equals(media Media) boolean
        +compareTo() int
    }
    class Book {
        +authors List<String>
        +BookId() int title String category String cost float
        +BookId() int title String category String cost float authors List<String>
        +getAuthors() List<String>
        +setAuthors(authors List<String>) void
        +addAuthor(authorName String) void
        +removeAuthor(authorName String) void
        +toString() String
    }
    class Disc {
        +length int
        +director String
        +DiscId() int title String category String cost float
        +DiscId() int title String category String cost float length int director String
        +getLength() int
        +setLength(length int) void
        +getDirector() String
        +setDirector(director String) void
    }
    class CompactDisc {
        +artist String
        +CompactDiscId() int title String category String cost float
        +CompactDiscId() int title String category String cost float artist String tracks Array<List<Track>>
        +play() void
        +addTrack(track Track) void
        +removeTrack(track Track) void
        +getLength() int
        +getArtist() String
        +setArtist(artist String) void
        +toString() String
    }
    class DigitalVideoDisc {
        +DigitalVideoDiscId() int title String category String cost float
        +play() void
        +toString() String
    }
    class Track {
        +titleTrack String
        +length int
        +TrackId() int title String category String cost float titleTrack String length int
        +play() void
        +getTitle() String
        +setTitle(title String) void
        +getLength() int
        +setLength(length int) void
        +equals(track Track) boolean
    }
    class ArrayList
    class ArrayTrack
    class ArrayMedia
    class Playable
    class Comparable
    class Comparator
    class List
    class ListString
    class ComparatorMedia

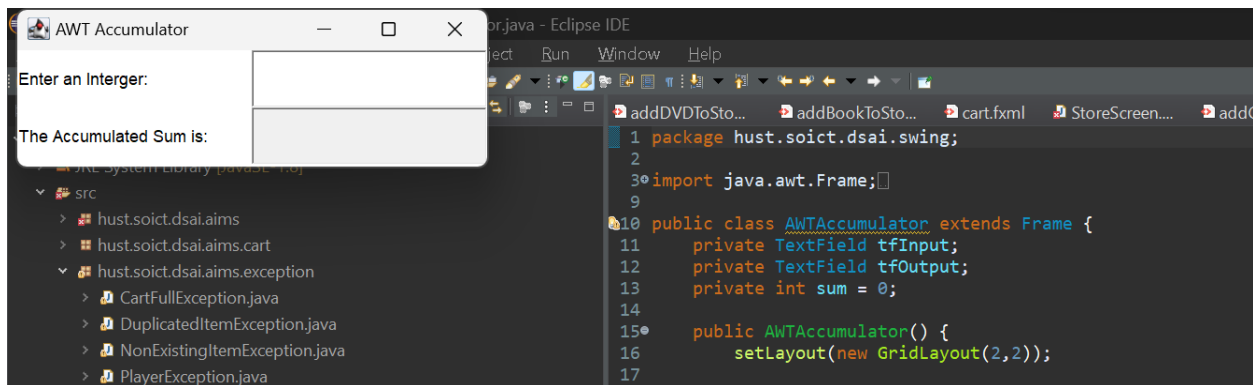
    Media <|-- Book
    Media <|-- Disc
    Disc <|-- CompactDisc
    Disc <|-- DigitalVideoDisc
    Media "1" -- "*" Track : tracks
    Media "1" -- "*" Book : books
    Media "1" -- "*" Disc : discs
    Media "1" -- "*" CompactDisc : compactDiscs
    Media "1" -- "*" DigitalVideoDisc : digitalVideoDiscs
    Media "1" -- "*" ArrayList : arrayList
    Media "1" -- "*" ArrayTrack : arrayTrack
    Media "1" -- "*" ArrayMedia : arrayMedia
    Media "1" -- "*" Playable : playable
    Media "1" -- "*" Comparable : comparable
    Media "1" -- "*" Comparator : comparator
    Media "1" -- "*" List : list
    Media "1" -- "*" ListString : listString
    Media "1" -- "*" ComparatorMedia : comparatorMedia
  
```



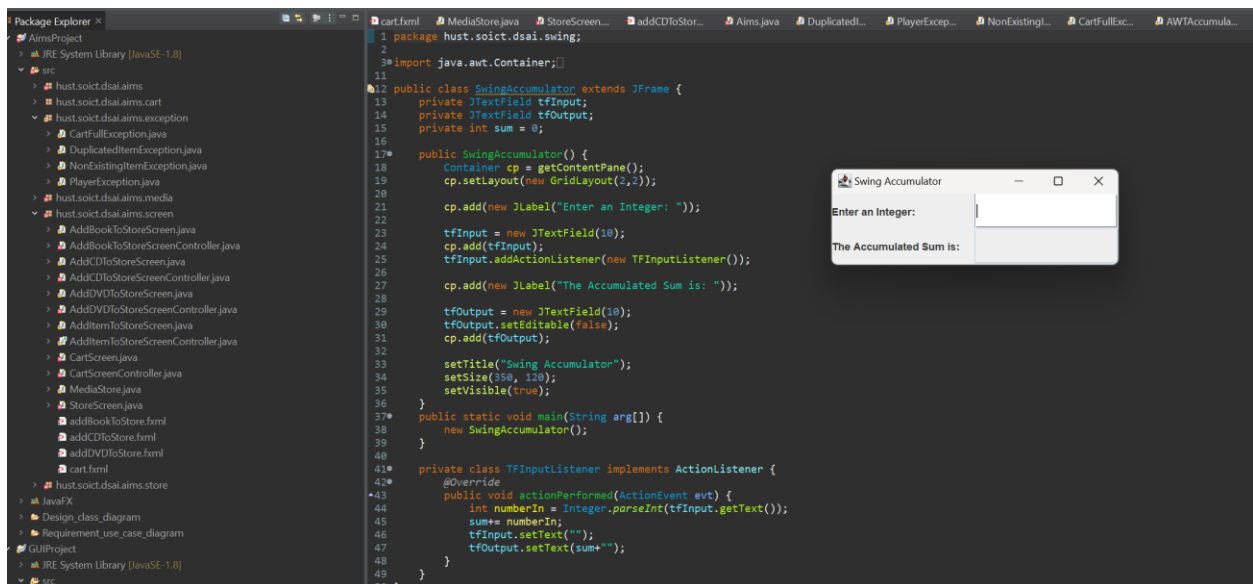
1. Swing Components

1.1. AWT Accumulator

```
addDVDtoSto... addBooktoSto... cart.xml StoreScreen.... addCDtoSto/... Aims.java
1 package hust.soict.dsai.swing;
2
3 import java.awt.Frame;
4
5
6
7
8
9
10 public class AWTAccumulator extends Frame {
11     private TextField tfInput;
12     private TextField tfOutput;
13     private int sum = 0;
14
15     public AWTAccumulator() {
16         setLayout(new GridLayout(2,2));
17
18         add(new Label("Enter an Interger: "));
19
20         tfInput = new TextField(10);
21         add(tfInput);
22         tfInput.addActionListener(new TFInputListener());
23
24         add(new Label("The Accumulated Sum is: "));
25
26         tfOutput = new TextField(10);
27         tfOutput.setEditable(false);
28         add(tfOutput);
29
30         setTitle("AWT Accumulator");
31         setSize(350,120);
32         setVisible(true);
33     }
34
35     public static void main (String arg[]) {
36         new AWTAccumulator();
37     }
38
39     private class TFInputListener implements ActionListener {
40         @Override
41         public void actionPerformed(ActionEvent evt) {
42             int numberIn = Integer.parseInt(tfInput.getText());
43             sum+= numberIn;
44             tfInput.setText("");
45             tfOutput.setText(sum + "");
46         }
47     }
48 }
49
```



1.2. Swing Accumulator



Compare Swing and AWT elements

The Top-Level Containers:

- AWT: The top-level container in AWT is called "Frame".
- Swing: The top-level container in Swing is called "JFrame". It is an extension of Frame and provides additional features and functionality.

The Class Names of Components:

- AWT: The class names of AWT components usually start with a capital letter, such as Button, TextField, and Label.
- Swing: The Swing components generally have the same names as their AWT counterparts but start with a "J" prefix. For example, the equivalent Swing components would be JButton, JTextField, and JLabel.

1.3. NumberGrid

```

1 package hust.soict.dsai.swing;
2
3 import java.awt.BorderLayout;
4
15 public class NumberGrid extends JFrame {
16     private JButton[] btnNumbers = new JButton[10];
17     private JButton btnDelete, btnReset;
18     private JTextField tfDisplay;
19
20     public NumberGrid() {
21         tfDisplay = new JTextField();
22         tfDisplay.setComponentOrientation(
23             ComponentOrientation.RIGHT_TO_LEFT);
24
25         JPanel panelButtons = new JPanel(new GridLayout(4,3));
26         addButtons(panelButtons);
27
28         Container cp = getContentPane();
29         cp.setLayout(new BorderLayout());
30         cp.add(tfDisplay, BorderLayout.NORTH);
31         cp.add(panelButtons, BorderLayout.CENTER);
32
33         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
34         setTitle("Number Grid");
35         setSize(200,200);
36         setVisible(true);
37     }
38
39     void addButtons(JPanel panelButtons) {
40         ButtonListener btnListener = new ButtonListener();
41         for(int i = 1; i<=9;i++) {
42             btnNumbers[i]= new JButton(""+i);
43             panelButtons.add(btnNumbers[i]);
44             btnNumbers[i].addActionListener(btnListener);
45         }
46         btnDelete = new JButton("DEL");
47         panelButtons.add(btnDelete);
48         btnDelete.addActionListener(btnListener);
49
50         btnNumbers[0] = new JButton("0");
51         panelButtons.add(btnNumbers[0]);
52         btnNumbers[0].addActionListener(btnListener);
53
54         btnReset = new JButton("C");
55         panelButtons.add(btnReset);
56         btnReset.addActionListener(btnListener);
57     }
58

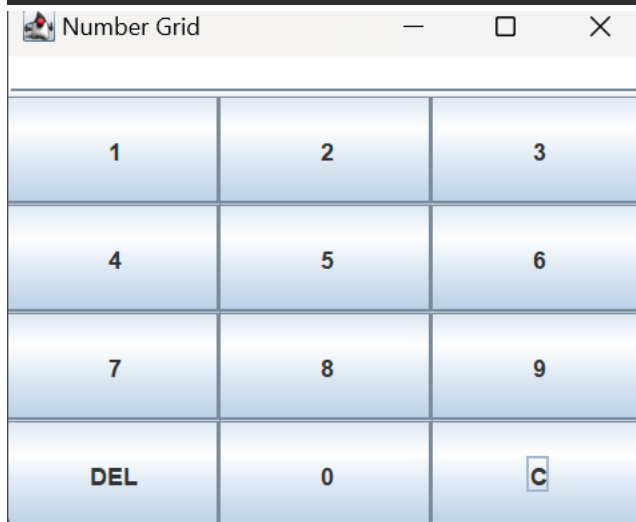
```

```

private class ButtonListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        String button = e.getActionCommand();
        if(button.charAt(0)>= '0' && button.charAt(0)<= '9') {
            tfDisplay.setText(tfDisplay.getText()+button);
        }
        else if (button.equals("DEL")) {
            //handles the "DEL" button
            String text = tfDisplay.getText();
            if (text.length() > 0) {
                tfDisplay.setText(text.substring(0, text.length() - 1));
            }
        }
        else {
            //handles the "C" button
            tfDisplay.setText("");
        }
    }
}

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

```



2. Create a graphical user interface for AIMS with Swing

2.1. View Store Screen

2.1.1. Create the StoreScreen class

```

public class StoreScreen extends JFrame {
    private Store store;
    private Cart cart;

    public static void main(String[] args) throws Exception {
        // Test
        Store myStore = new Store();
        Cart myCart = new Cart();
    }
}

```

2.1.2. The NORTH Component

```

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");
    addBook.addActionListener(new AddBookListener());
    smUpdateStore.add(addBook);
    JMenuItem addCD = new JMenuItem("Add CD");
    addCD.addActionListener(new AddCDListener());
    smUpdateStore.add(addCD);
    JMenuItem addDVD = new JMenuItem("Add DVD");
    addDVD.addActionListener(new AddDVDListener());
    smUpdateStore.add(addDVD);

    menu.add(smUpdateStore);
    menu.add(new JMenuItem("View store"));
    JMenuItem cart = new JMenuItem("View cart");
    cart.addActionListener(new ViewCartListener());
    menu.add(cart);

    JMenuBar menuBar = new JMenuBar();
    menuBar.setLayout(new FlowLayout(FlowLayout.LEFT));
    menuBar.add(menu);

    return menuBar;
}

```

```

• JPanel createHeader() {
    JPanel header = new JPanel();
    header.setLayout(new BoxLayout(header, BoxLayout.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));
    cart.addActionListener(new ViewCartListener());

    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;
}

```

2.1.3. The CENTER Component

```

• JPanel createCenter() {
    JPanel center = new JPanel();
    center.setLayout(new GridLayout(5, 5, 2, 2));

    ArrayList<Media> mediaInStore = store.getItemsInStore();
    for (Media media : mediaInStore) {
        MediaStore cell = new MediaStore(media, cart);
        center.add(cell);
    }
}

```

2.1.4. The MediaStore Class


```

1 package hust.soict.dsai.aims.screen;
2
3 import hust.soict.dsai.aims.media.*;
4
5 public class MediaStore extends JPanel {
6     private Media media;
7     private Cart cart;
8
9     public MediaStore(Media media, Cart cart) {
10         this.media = media;
11         this.cart = cart;
12         this.setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
13
14         JLabel title = new JLabel(media.getTitle());
15         title.setFont(new Font(title.getFont().getName(), Font.PLAIN, 20));
16         title.setAlignmentX(CENTER_ALIGNMENT);
17
18         JLabel cost = new JLabel("" + media.getCost() + " $");
19         cost.setAlignmentX(CENTER_ALIGNMENT);
20
21         JPanel container = new JPanel();
22         container.setLayout(new FlowLayout(FlowLayout.CENTER));
23
24         JButton addToCartButton = new JButton("Add to cart");
25         addToCartButton.addActionListener(new AddToCartListener());
26         container.add(addToCartButton);
27
28         JButton detailsButton = new JButton("View details");
29         detailsButton.addActionListener(new DetailsListener());
30         container.add(detailsButton);
31
32         if (media instanceof Playable) {
33             JButton playButton = new JButton("Play");
34             playButton.addActionListener(new PlayButtonListener());
35             container.add(playButton);
36         }
37
38         this.add(Box.createVerticalGlue());
39         this.add(title);
40         this.add(cost);
41         this.add(Box.createVerticalGlue());
42         this.add(container);
43
44         this.setBorder(BorderFactory.createLineBorder(Color.BLACK));
45     }
46 }

```

2.1.5. Putting it all together

```
public StoreScreen(Store store, Cart cart) {
    this.store = store;
    this.cart = cart;
    Container cp = getContentPane();
    cp.setLayout(new BorderLayout());

    cp.add(createNorth(), BorderLayout.NORTH);
    cp.add(createCenter(), BorderLayout.CENTER);

    setVisible(true);
    setTitle("Store");
    setSize(1024, 768);

    Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();
    int w = getSize().width;
    int h = getSize().height;
    int x = (dim.width - w) / 2;
    int y = (dim.height - h) / 2;
    setLocation(x, y);
}
```

2.2. *Adding more user interaction*

3. *JavaFX API*



4.

3.2. Creater the Controller Class

```

1 package hust.soict.dsai.javaafx;
2 import javafx.event.ActionEvent;
9
10 public class PainterController {
11
12     boolean isEraserMode = false;
13
14     @FXML
15     private ToggleGroup identical;
16
17     @FXML
18     private Pane drawingAreaPane;
19
20     @FXML
21     void clearButtonPressed(ActionEvent event) {
22         drawingAreaPane.getChildren().clear();
23     }
24
25     @FXML
26     void drawingAreaMouseDragged(MouseEvent event) {
27         // Check if the target is the drawing area
28         if (event.getTarget() == drawingAreaPane) {
29             if (isEraserMode) {
30                 Circle eraser = new Circle(event.getX(), event.getY(), 4, Color.WHITE);
31                 drawingAreaPane.getChildren().add(eraser);
32             } else {
33                 Circle pen = new Circle(event.getX(), event.getY(), 4, Color.BLACK);
34                 drawingAreaPane.getChildren().add(pen);
35             }
36         }
37     }
38
39     @FXML
40     void penMode(ActionEvent event) {
41         isEraserMode = false;
42     }
43
44     @FXML
45     void eraserMode (ActionEvent event) {
46         isEraserMode = true;
47     }
48
49 }
50

```

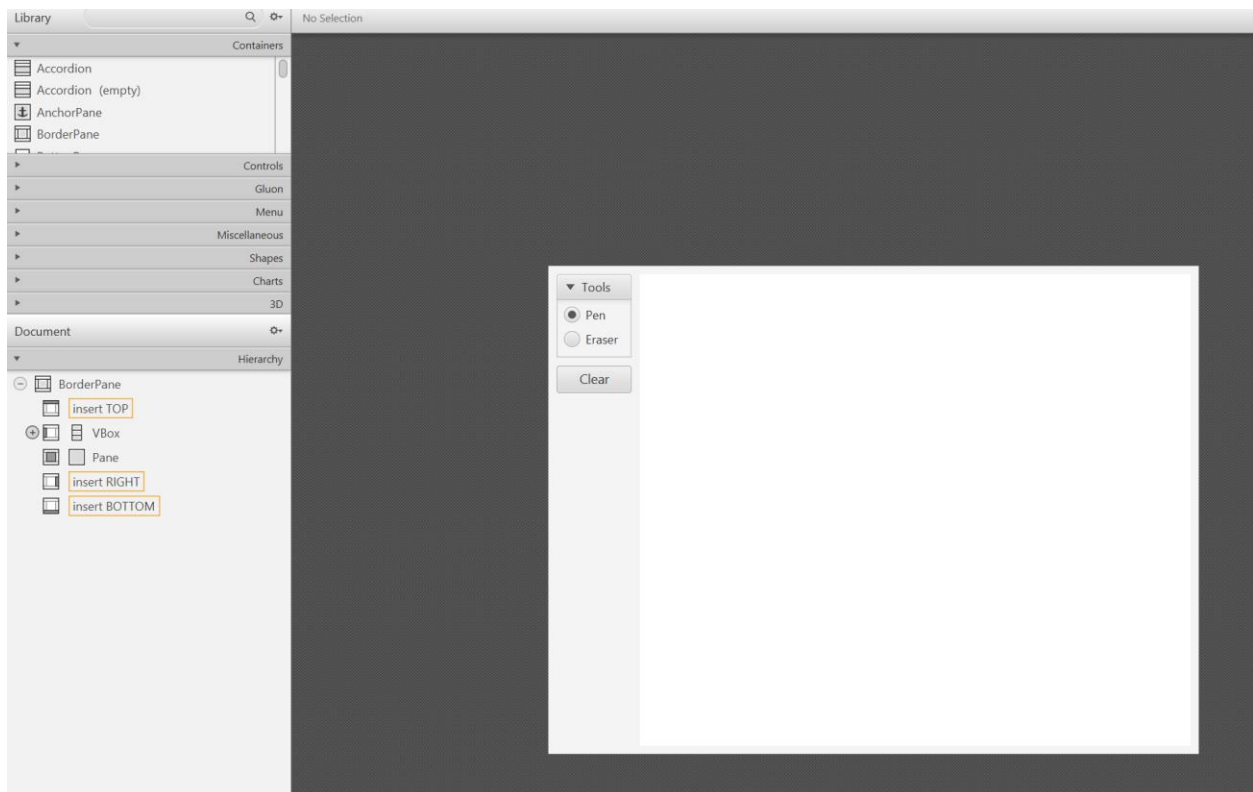
3.3. Create the application

```

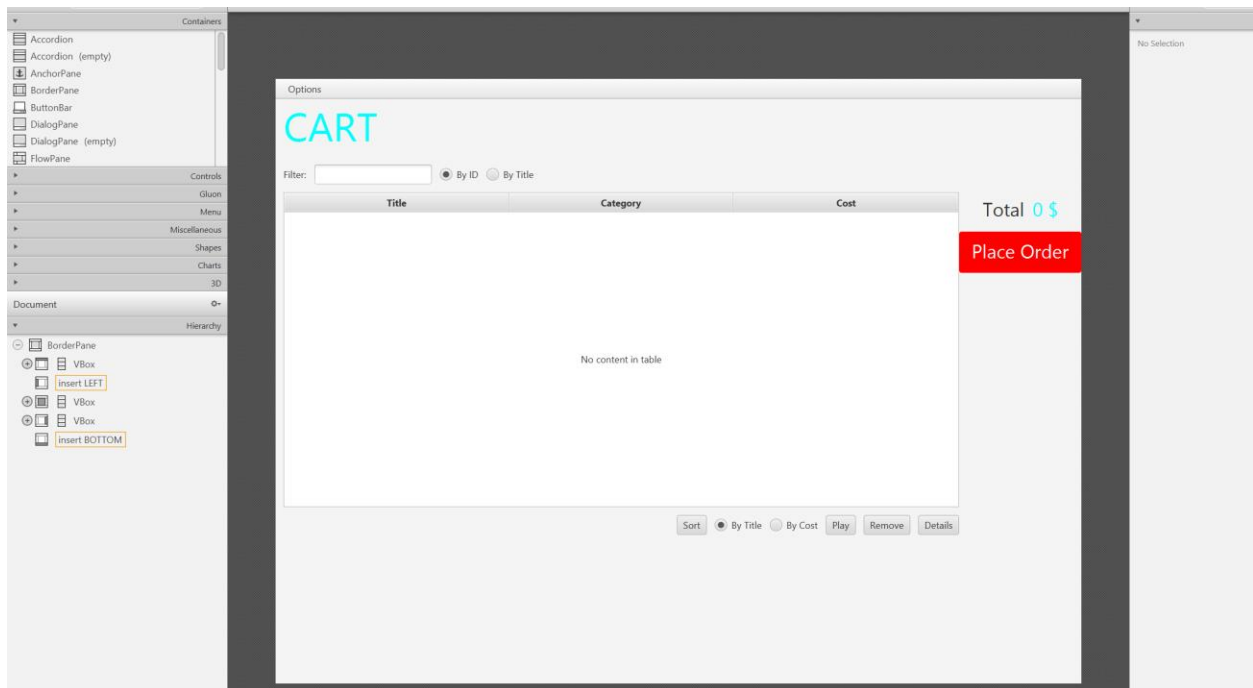
1 package hust.soict.dsai.javaafx;
2
3 import javafx.application.Application;
8
9 public class Painter extends Application {
10
11     @Override
12     public void start(Stage stage) throws Exception {
13         Parent root = FXMLLoader.load(getClass().
14             getResource("GUIProject/src/hust/soict/dsai/javaafx/Painter.fxml"));
15
16         Scene scene = new Scene(root);
17         stage.setTitle("Painter");
18         stage.setScene(scene);
19         stage.show();
20
21     }
22     public static void main(String[] args) {
23         launch(args);
24     }
25 }
26

```

3.4 Practice Exercise



4. Setting up the View Cart Screen with ScreenBuilder



5. Integrating JavaFX into Swing Application

```

1 package hust.soict.dsai.aims.screen;
2
3 import java.awt.event.WindowAdapter;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21 public class CartScreen extends JFrame {
22     private Store store;
23     private Cart cart;
24
25     public CartScreen(Store store, Cart cart) {
26         super();
27
28         this.store = store;
29         this.cart = cart;
30         this.setSize(1024, 768);
31         JFXPanel fxPanel = new JFXPanel();
32         this.add(fxPanel);
33
34         this.setTitle("Cart");
35         this.setVisible(true);
36         JFrame frame = this;
37
38         this.addWindowListener(new WindowAdapter() {
39             @Override
40             public void windowClosing(WindowEvent e) {
41                 new StoreScreen(store, cart);
42                 dispose();
43             }
44         });
45
46         Platform.runLater(new Runnable() {
47
48             @Override
49             public void run() {
50
51                 try {
52                     FXMLLoader loader = new FXMLLoader(
53                         getClass().getResource("cart.fxml"));
54                     CartScreenController controller = new CartScreenController(store, cart, frame);
55                     loader.setController(controller);
56                     Parent root = loader.load();
57                     fxPanel.setScene(new Scene(root));
58                 } catch (IOException e) {
59                     e.printStackTrace();
60                 }
61             }
62         });
63     }
64 }

```

6. View the items in Cart JavaFX data-driven UI

```

1 package hust.soict.dsai.aims.screen;
2
3 import javax.swing.JFrame;
4
5
6
7
8
9
10
11
12
13
14 public class CartScreenController {
15     private Store store;
16     private Cart cart;
17     private boolean filterByID = true;
18     private boolean sortByTitle = true;
19     private FilteredList<Media> filteredCart;
20     private JFrame stage;
21
22
23
24     @FXML
25     private TableView<Media> tblMedia;
26
27
28     @FXML
29     private TableColumn<Media, String> colMediaTitle;
30
31
32     @FXML
33     private TableColumn<Media, String> colMediaCategory;
34
35
36     @FXML
37     private TableColumn<Media, String> colMediaCost;
38
39
40     @FXML
41     private Button btnPlay;
42
43
44     @FXML
45     private Button btnRemove;
46
47
48     @FXML
49     private Button btnDetails;
50
51
52     @FXML
53     private TextField tfFilter;
54
55
56     @FXML
57     private Label costLabel;
58
59
60     public CartScreenController(Store store, Cart cart, JFrame stage) {
61         super();
62         this.store = store;
63         this.cart = cart;
64         this.stage = stage;
65     }
66

```

```

66
67• @FXML
68 public void initialize() {
69     filteredCart = new FilteredList<Media>(this.cart.getItemsOrdered(), s -> true);
70
71     colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>("title"));
72     colMediaCategory.setCellValueFactory(new PropertyValueFactory<Media, String>("category"));
73     colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, String>("cost"));
74     tblMedia.setItems(filteredCart);
75
76     btnPlay.setVisible(false);
77     btnRemove.setVisible(false);
78     btnDetails.setVisible(false);
79
80     costLabel.setText(String.valueOf(this.cart.totalCost()));
81
82•     tblMedia.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Media>() {
83
84•         @Override
85         public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {
86             updateButtonBar(newValue);
87         }
88     });
89
90•     tfFilter.textProperty().addListener(new ChangeListener<String> () {
91
92•         @Override
93         public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
94             showFilteredMedia(newValue);
95         }
96     });
97
98 }
99

```

7. Updating buttons based on selected item in TableView ChangeListener


```

68     public void initialize() {
69         filteredCart = new FilteredList<Media>(this.cart.getItemsOrdered(), s -> true);
70
71         colMediaTitle.setCellValueFactory(new PropertyValueFactory<Media, String>("title"));
72         colMediaCategory.setCellValueFactory(new PropertyValueFactory<Media, String>("category"));
73         colMediaCost.setCellValueFactory(new PropertyValueFactory<Media, String>("cost"));
74         tblMedia.setItems(filteredCart);
75
76         btnPlay.setVisible(false);
77         btnRemove.setVisible(false);
78         btnDetails.setVisible(false);
79
80         costLabel.setText(String.valueOf(this.cart.totalCost()));
81
82         tblMedia.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Media>() {
83
84             @Override
85             public void changed(ObservableValue<? extends Media> observable, Media oldValue, Media newValue) {
86                 updateButtonBar(newValue);
87             }
88         });
89
90         tffilter.textProperty().addListener(new ChangeListener<String>() {
91
92             @Override
93             public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
94                 showFilteredMedia(newValue);
95             }
96         });
97
98
99
100     private void updateButtonBar(Media media) {
101         if (media == null) {
102             btnRemove.setVisible(false);
103             btnDetails.setVisible(false);
104             btnPlay.setVisible(false);
105         } else {
106             btnRemove.setVisible(true);
107             btnDetails.setVisible(true);
108             if (media instanceof Playable) {
109                 btnPlay.setVisible(true);
110             } else {
111                 btnPlay.setVisible(false);
112             }
113         }
114     }
115
116     private void showFilteredMedia(String filter) {
117         if (filter == null || filter.length() == 0) {
118             filteredCart.setPredicate(s -> true);
119         } else {
120             if (filterByID) {
121                 try {
122                     filteredCart.setPredicate(s -> s.getID() == Integer.parseInt(filter));
123                 } catch (NumberFormatException e) {}
124             } else {
125                 filteredCart.setPredicate(s -> s.getTitle().toLowerCase().contains(filter));
126             }
127         }
128     }
129

```

8. Deleting a media

```

@FXML
private void removeButtonPressed(ActionEvent event) {
    Media media = tblMedia.getSelectionModel().getSelectedItem();
    try {
        this.cart.removeMedia(media);
    } catch (NonExistingItemException e) {
        Alert alert = new Alert(AlertType.ERROR);
        alert.setTitle("Notification");
        alert.setHeaderText("Failed to remove");
        alert.setContentText("Media not in cart");
        alert.showAndWait();
    }
    costLabel.setText(String.valueOf(this.cart.totalCost()));
}

```

9. Filter items in cart FilteredList

```

//
•   tfFilter.textProperty().addListener(new ChangeListener<String> () {
•
•       @Override
        public void changed(ObservableValue<? extends String> observable, String oldValue, String newValue) {
            showFilteredMedia(newValue);
        }
    });

```

10. Complete the Aims GUI Application

Cart Screen

```

0   private void placeOrderPressed(ActionEvent event) {
1       if (this.cart.getSize() > 0) {
2           Alert alert = new Alert(AlertType.INFORMATION);
3           alert.setTitle("Notification");
4           alert.setHeaderText("Success!");
5           alert.setContentText("Your order has been placed.");
6           alert.showAndWait();
7           this.cart.empty();
8           costLabel.setText(String.valueOf(this.cart.totalCost()));
9       } else {
10          Alert alert = new Alert(AlertType.ERROR);
11          alert.setTitle("Notification");
12          alert.setHeaderText("ERROR: Failed to place order.");
13          alert.setContentText("Your cart is empty");
14          alert.showAndWait();
15      }
16  }
17  }
18

```

```

@FXML
private void playButtonPressed(ActionEvent event) {
    Media media = this.tblMedia.getSelectionModel().getSelectedItem();
    try {
        ((Playable)media).play();
    } catch (PlayerException 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();
    }
}

```

```

@FXML
private void detailsButtonPressed(ActionEvent event) {
    Media media = this.tblMedia.getSelectionModel().getSelectedItem();
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Detail information");
    alert.setHeaderText("Viewing " + media.getTitle() + " detail information.");
    alert.setContentText(media.getDetails());
    alert.showAndWait();
}

```

Store Screen

Update Store Screen

```

private class AddDVDListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        new AddDVDToStoreScreen(store, cart);
        dispose();
    }
}

private class AddBookListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        new AddBookToStoreScreen(store, cart);
        dispose();
    }
}

private class AddCDListener implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        new AddCDToStoreScreen(store, cart);
        dispose();
    }
}

```

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

- hust.soict.dsai.aims.exception
 - CartFullException.java
 - DuplicatedItemException.java
 - NonExistingItemException.java
 - PlayerException.java

```
addDVDtoSto... addBooktoSto... cart.xml MediaStore.java StoreScreen... AddDVDtoSto... AddItemtoSto... addCDtoSto... Aims.java
1 package hust.soict.dsai.aims.exception;
2
3 public class PlayerException extends Exception {
4
5     public PlayerException() {
6         // TODO Auto-generated constructor stub
7     }
8
9     public PlayerException(String message) {
10         super(message);
11         // TODO Auto-generated constructor stub
12     }
13
14     public PlayerException(Throwable cause) {
15         super(cause);
16         // TODO Auto-generated constructor stub
17     }
18
19     public PlayerException(String message, Throwable cause) {
20         super(message, cause);
21         // TODO Auto-generated constructor stub
22     }
23
24     public PlayerException(String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace) {
25         super(message, cause, enableSuppression, writableStackTrace);
26         // TODO Auto-generated constructor stub
27     }
28
29 }
```

```

1 package hust.soict.dsai.aims.exception;
2
3 public class CartFullException extends Exception {
4
5     public CartFullException() {
6         // TODO Auto-generated constructor stub
7     }
8
9     public CartFullException(String message) {
10         super(message);
11         // TODO Auto-generated constructor stub
12     }
13
14     public CartFullException(Throwable cause) {
15         super(cause);
16         // TODO Auto-generated constructor stub
17     }
18
19     public CartFullException(String message, Throwable cause) {
20         super(message, cause);
21         // TODO Auto-generated constructor stub
22     }
23
24     public CartFullException(String message, Throwable cause, boolean enableSuppression, boolean writableStackTrace) {
25         super(message, cause, enableSuppression, writableStackTrace);
26         // TODO Auto-generated constructor stub
27     }
28
29 }

```

```

1 package hust.soict.dsai.aims.exception;
2
3 public class DuplicatedItemException extends Exception {
4
5     public DuplicatedItemException() {
6         // TODO Auto-generated constructor stub
7     }
8
9     public DuplicatedItemException(String message) {
10         super(message);
11         // TODO Auto-generated constructor stub
12     }
13
14     public DuplicatedItemException(Throwable cause) {
15         super(cause);
16         // TODO Auto-generated constructor stub
17     }
18
19     public DuplicatedItemException(String message, Throwable cause) {
20         super(message, cause);
21         // TODO Auto-generated constructor stub
22     }
23
24     public DuplicatedItemException(String message, Throwable cause, boolean enableSuppression,
25         boolean writableStackTrace) {
26         super(message, cause, enableSuppression, writableStackTrace);
27         // TODO Auto-generated constructor stub
28     }
29
30 }

```

```
1 package hust.soict.dsai.aims.exception;
2
3 public class NonExistingItemException extends Exception {
4
5     public NonExistingItemException() {
6         // TODO Auto-generated constructor stub
7     }
8
9     public NonExistingItemException(String message) {
10         super(message);
11         // TODO Auto-generated constructor stub
12     }
13
14     public NonExistingItemException(Throwable cause) {
15         super(cause);
16         // TODO Auto-generated constructor stub
17     }
18
19     public NonExistingItemException(String message, Throwable cause) {
20         super(message, cause);
21         // TODO Auto-generated constructor stub
22     }
23
24     public NonExistingItemException(String message, Throwable cause, boolean enableSuppression,
25         boolean writableStackTrace) {
26         super(message, cause, enableSuppression, writableStackTrace);
27         // TODO Auto-generated constructor stub
28     }
29
30 }
```