06 - Media Management System

Tyler Scott
Chris Rhoda
Jackson Markowski
Sean Moss

Walkthrough

https://youtu.be/7mBTZAP8LW0

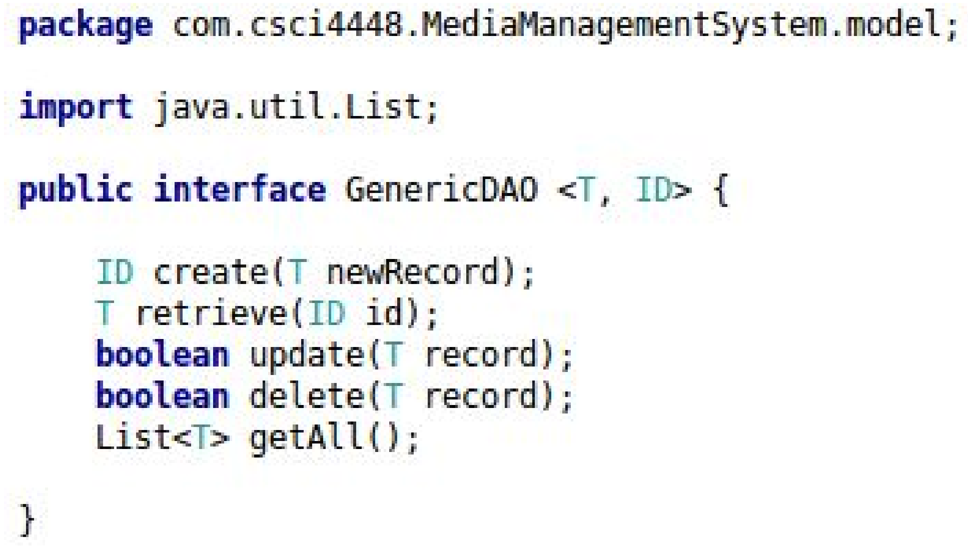
Data Access Object (DAO) Design Pattern

- Architectural design pattern, not GoF
- Used to remove the details of persistence from business logic
- Supports the Single Responsibility principle
- Increases the simplicity of accessing the database
- Minimizes duplication of CRUD operations in Entity classes

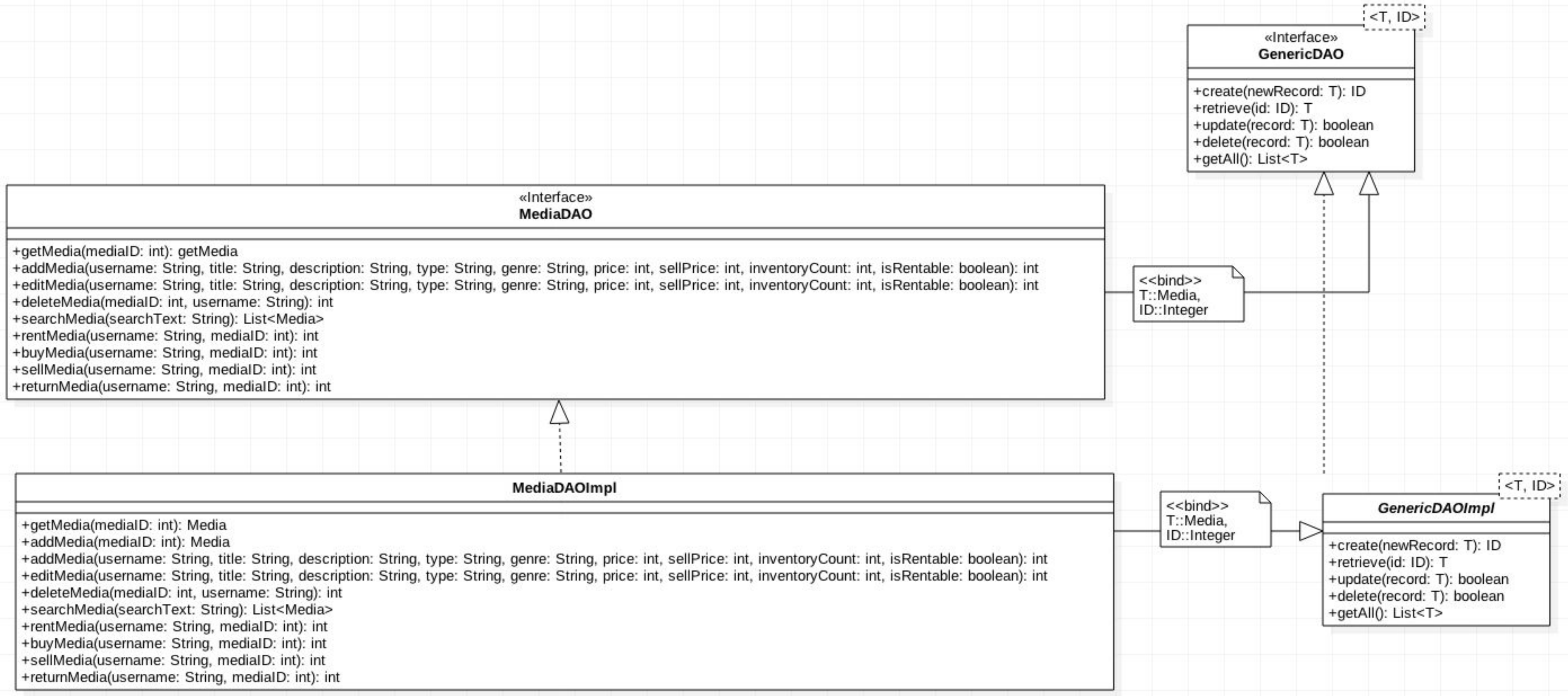
Data Access Object (DAO) Participants

- GenericDAO: Java Interface containing methods for CRUD operations
- GenericDAOImpl: Abstract class implementing GenericDAO methods
- Specific Entity DAO: Java Interface containing business logic methods for a specific Entity table
    - For example, UserDAO for a User table
    - Extends GenericDAO to initialize the generic types
- Specific Entity DAO Impl: Class implementing the business logic methods for its Specific Entity DAO
    - Extends GenericDAOImpl to use CRUD operations

GenericDAO Interface



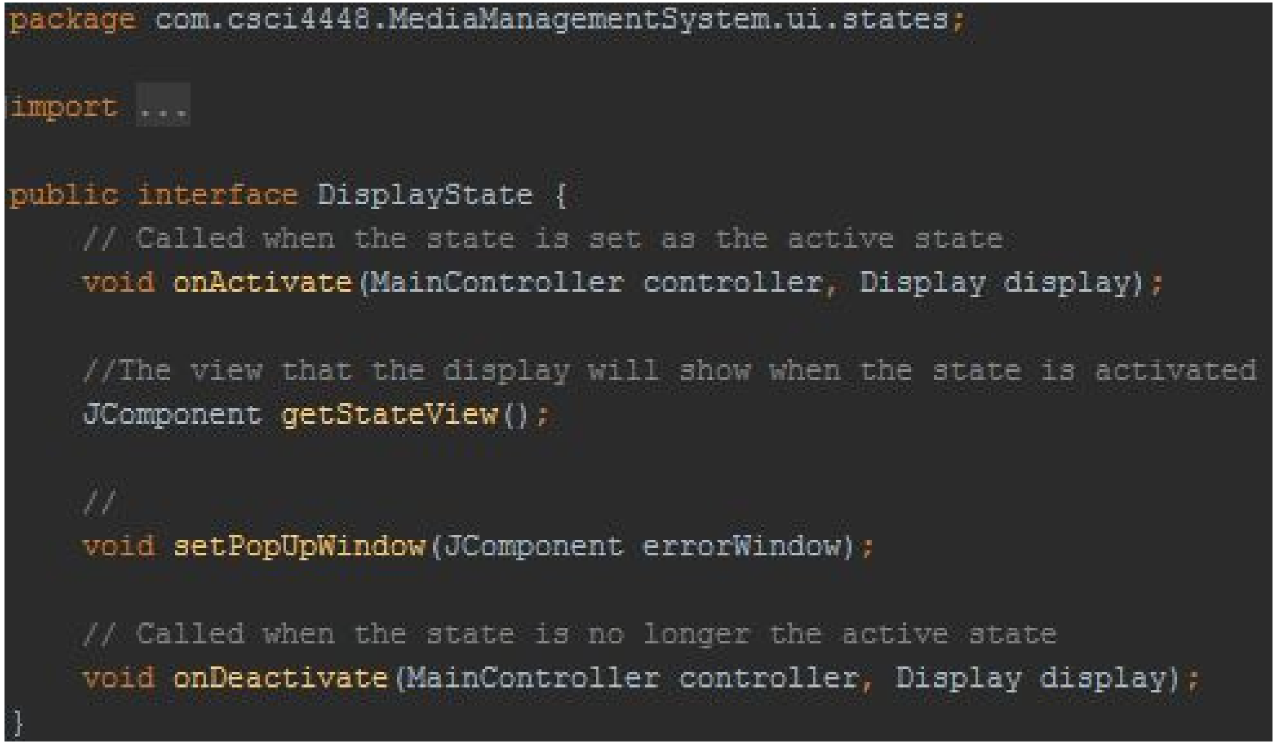
```
package com.csci4448.MediaManagementSystem.model;

import java.util.List;

public interface GenericDAO <T, ID> {

    ID create(T newRecord);
    T retrieve(ID id);
    boolean update(T record);
    boolean delete(T record);
    List<T> getAll();

}
```


<T, ID>
«Interface»
GenericDAO
+create(newRecord: T): ID
+retrieve(id: ID): T
+update(record: T): boolean
+delete(record: T): boolean
+getAll(): List<T>
«Interface»
MediaDAO
+getMedia(mediaID: int): getMedia
+addMedia(username: String, title: String, description: String, type: String, genre: String, price: int, sellPrice: int, inventoryCount: int, isRentable: boolean): int
+editMedia(username: String, title: String, description: String, type: String, genre: String, price: int, sellPrice: int, inventoryCount: int, isRentable: boolean): int
+deleteMedia(mediaID: int, username: String): int
+searchMedia(searchText: String): List<Media>
+rentMedia(username: String, mediaID: int): int
+buyMedia(username: String, mediaID: int): int
+sellMedia(username: String, mediaID: int): int
+returnMedia(username: String, mediaID: int): int
<<bind>>
T::Media,
ID::Integer
MediaDAOImpl
+getMedia(mediaID: int): Media
+addMedia(mediaID: int): Media
+addMedia(username: String, title: String, description: String, type: String, genre: String, price: int, sellPrice: int, inventoryCount: int, isRentable: boolean): int
+editMedia(username: String, title: String, description: String, type: String, genre: String, price: int, sellPrice: int, inventoryCount: int, isRentable: boolean): int
+deleteMedia(mediaID: int, username: String): int
+searchMedia(searchText: String): List<Media>
+rentMedia(username: String, mediaID: int): int
+buyMedia(username: String, mediaID: int): int
+sellMedia(username: String, mediaID: int): int
+returnMedia(username: String, mediaID: int): int
<<bind>>
T::Media,
ID::Integer
<T, ID>
GenericDAOImpl
+create(newRecord: T): ID
+retrieve(id: ID): T
+update(record: T): boolean
+delete(record: T): boolean
+getAll(): List<T>

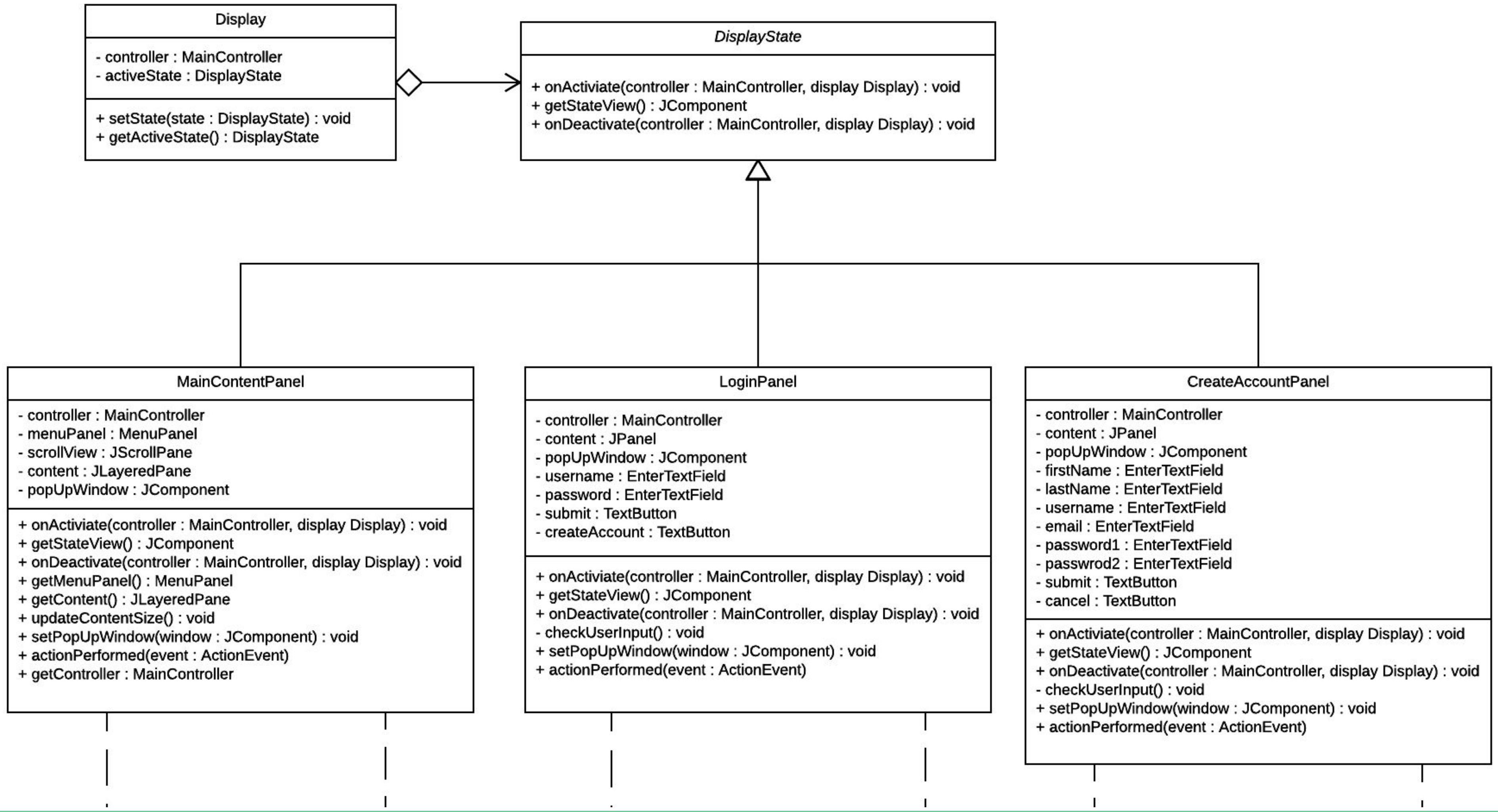
Other Design Patterns

- **Singleton** for the System Inventory of media
- **Factory** for creating text components (buttons, fields, etc.)
- **State** for switching the display between login, create account, and main views

State



```
package com.csci4448.MediaManagementSystem.ui.states;

import ...

public interface DisplayState {
    // Called when the state is set as the active state
    void onActivate(MainController controller, Display display);

    //The view that the display will show when the state is activated
    JComponent getStateView();

    //
    void setPopUpWindow(JComponent errorWindow);

    // Called when the state is no longer the active state
    void onDeactivate(MainController controller, Display display);
}
```


Display
- controller : MainController
- activeState : DisplayState
+ setState(state : DisplayState) : void
+ getActiveState() : DisplayState
DisplayState
+ onActiviate(controller : MainController, display Display) : void
+ getStateView() : JComponent
+ onDeactivate(controller : MainController, display Display) : void
MainContentPanel
- controller : MainController
- menuPanel : MenuPanel
- scrollView : JScrollPane
- content : JLayeredPane
- popUpWindow : JComponent
+ onActiviate(controller : MainController, display Display) : void
+ getStateView() : JComponent
+ onDeactivate(controller : MainController, display Display) : void
+ getMenuPanel() : MenuPanel
+ getContent() : JLayeredPane
+ updateContentSize() : void
+ setPopUpWindow(window : JComponent) : void
+ actionPerformed(event : ActionEvent)
+ getController : MainController
LoginPanel
- controller : MainController
- content : JPanel
- popUpWindow : JComponent
- username : EnterTextField
- password : EnterTextField
- submit : TextButton
- createAccount : TextButton
+ onActiviate(controller : MainController, display Display) : void
+ getStateView() : JComponent
+ onDeactivate(controller : MainController, display Display) : void
- checkUserInput() : void
+ setPopUpWindow(window : JComponent) : void
+ actionPerformed(event : ActionEvent)
CreateAccountPanel
- controller : MainController
- content : JPanel
- popUpWindow : JComponent
- firstName : EnterTextField
- lastName : EnterTextField
- username : EnterTextField
- email : EnterTextField
- password1 : EnterTextField
- passwrod2 : EnterTextField
- submit : TextButton
- cancel : TextButton
+ onActiviate(controller : MainController, display Display) : void
+ getStateView() : JComponent
+ onDeactivate(controller : MainController, display Display) : void
- checkUserInput() : void
+ setPopUpWindow(window : JComponent) : void
+ actionPerformed(event : ActionEvent)