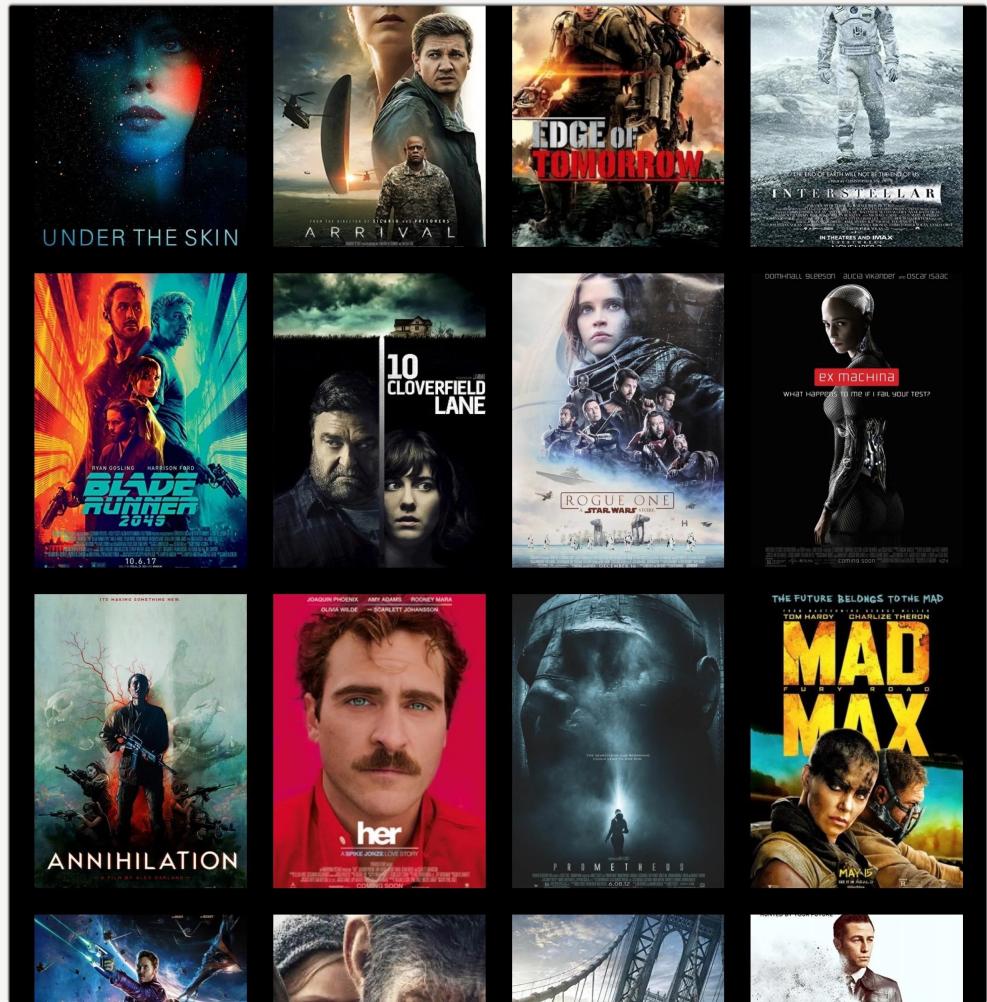


MOVIE DB

CHLOE ZHU



APPLICATION SCENARIO

- Track movies and TV shows that we have watched.
- Create a watch list of TV shows and movies
- Sort your list in the order of the director, year released, rating and Genres.

CAPABILITIES OF MOVIEDB

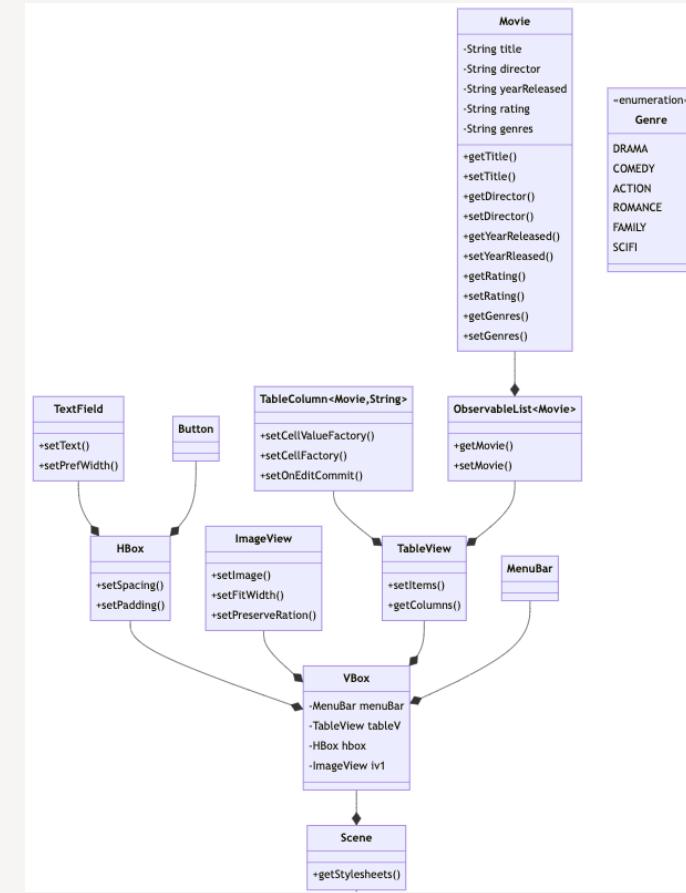
- Add a movie
 - Edit a movie
 - Rate a movie
 - Tag movie

The image shows a 3x3 grid of movie posters. The top row contains 'Under the Skin', 'Arrival', and 'Edge of Tomorrow'. The middle row contains 'Blade Runner 2049', '10 Cloverfield Lane', and 'Rogue One: A Star Wars Story'. The bottom row contains 'Interstellar', 'Ex Machina', and a poster for a movie featuring a woman's face and a bald man's head.

Demo: <https://youtu.be/xCFnyLFagDI>

UML CLASS AND IMPLEMENTATION

- Movie
- ObservableList<Movie>
- TableView
- TableColumn<Movie, String>
- Button



Movie
-String title
-String director
-String yearReleased
-String rating
-String genres
+getTitle()
+setTitle()
+getDirector()
+setDirector()
+getYearReleased()
+setYearReleased()
+getRating()
+setRating()
+getGenres()
+setGenres()

“enumeration”
Genre
DRAMA
COMEDY
ACTION
ROMANCE
FAMILY
SCIFI

```

1 public class Movie implements Serializable {
    private final SimpleStringProperty title;
    private final SimpleStringProperty director;
    private final SimpleStringProperty yearReleased;
    private final SimpleStringProperty rating;
    private final SimpleStringProperty genres;
    //constructor
    Movie(String title, String director, String yearReleased,
        String rating, String genres) {
        this.title = new SimpleStringProperty(title);
        this.director = new SimpleStringProperty(director);
        this.yearReleased = new
SimpleStringProperty(yearReleased);
        this.rating = new SimpleStringProperty(rating);
        this.genres = new SimpleStringProperty(genres);
    }
}

```

```

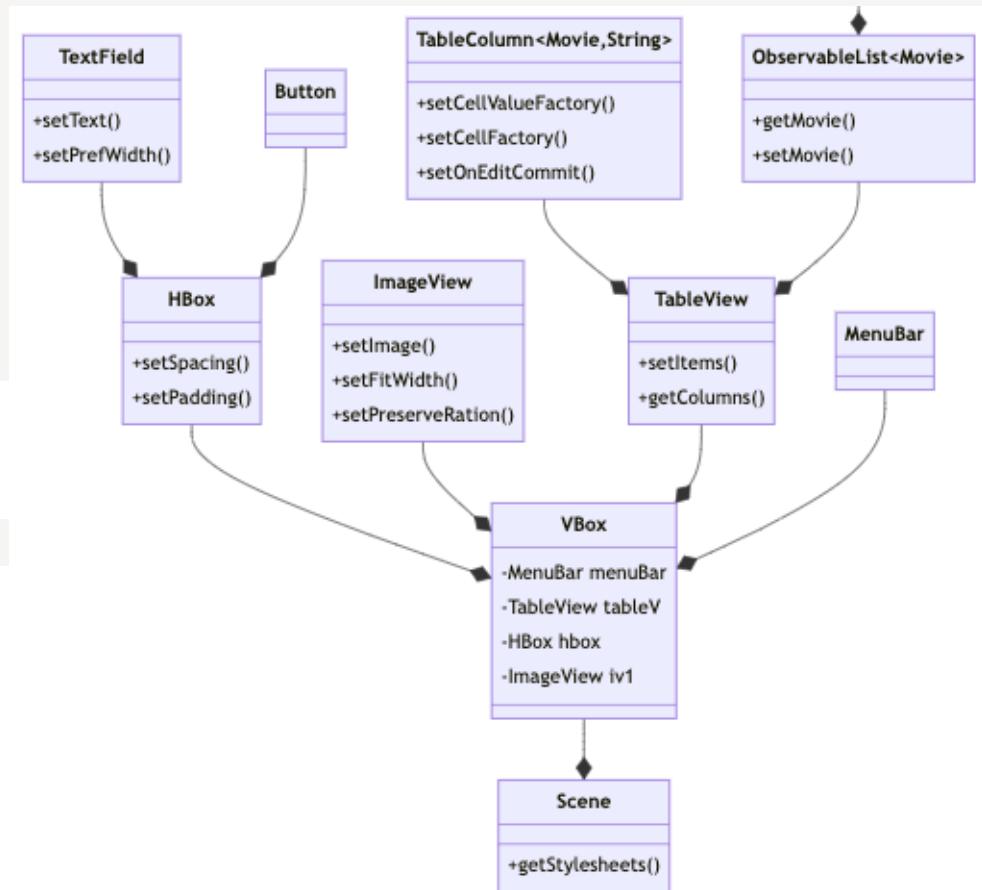
1 private final ObservableList<Movie> data =
FXCollections.observableArrayList(
    new Movie("The Matrix", "Lana Wachowski, Lilly
Wachowski", "1999", "5", "Sci-Fi"),
    new Movie("Crouching Tiger, Hidden Dragon", "Ang Lee",
"2000", "5", "Action"),
    ...
)

```

```
1 tableView.setItems(observableList<Movie>)
```

```
2 tableView.getColumns().addAll(titleCol, directorCol,  
yearReleasedCol, ratingCol, genresCol);
```

```
1 titleCol.setCellValueFactory(new PropertyValueFactory<Movie,  
String>("Title"));  
2 titleCol.setOnEditCommit(  
    new EventHandler<TableColumn.CellEditEvent<Movie,  
String>>() {  
        @Override  
        public void handle(TableColumn.CellEditEvent<Movie,  
String> t) {((Movie) t.getTableView().getItems().get(  
t.getTablePosition().getRow())).setTitle(t.getNewValue());  
        }  
    }  
);
```



LESSON LEARNED

- A more appealing appearance needed: CSS
- A deeper understanding of Java: Object is everywhere
- Debugging methods: read the error message, `System.out.println()`