Media Manager

9/16/15

Group 1

Jacob Ayers, Christopher Diglio, Benjamin Kilmer, Zachary Purdy

Computer Science and Programming II

CSC-231-02

Professor Joed Graf

Table of Contents

Proposal

Description

The media manager (tentative title) will primarily be a media organizer and playlist editor. The user will be able to input what movies, television shows, and music they own into the manager and organize them into various playlists that can be saved and loaded up again for future viewing or editing. The different media will be media objects that can utilize tags to help organize the content. For example, any music albums uploaded on a certain date could be tagged with that particular date. Then the user could simply use that date tag to find certain albums. Tags would also be used for genres, artists, producers, decade, etc. The program will be fairly minimalistic and easy to use. Drop down menus, radio buttons, and text boxes will be common use for some "advanced" Java programming. Click and drag and drop with a mouse would also be a nice feature to explore.

The program will also utilize a M.V.C. (Model View Control) design to organize and execute the code. In a nutshell, the user would import a media object through the viewer, the viewer would send the controller this request, the controller would tell the model to make a new media object, the model would then tell the controller to send a display command to the viewer, and finally the viewer would have the media pop up on the screen.

Classes

Model

Model

Music

Songs

Albums

Artists

Videos

Movies

TVShows

MusicVideos

Pictures

Playlists

Viewer

Viewer

The Viewer's classes and methods will be based of anything clickable in the GUI. At the moment, we have not learned enough graphics based programming to determine the proper class

and method naming for what we need. We will use temporary naming for now to plan the programming and update later on with proper Java syntax.

Controller

MediaController

Behaviors

Model

listByTag();

listSongs();

listAlbums();

listArtists();

listPlaylists();

Viewer

The Viewer's classes and methods will be based of anything clickable in the GUI. At the moment, we have not learned enough graphics based programming to determine the proper class and method naming for what we need. We will use temporary naming for now to plan the programming and update later on with proper Java syntax.

Controller

MediaController

createViewer(this)

createModel(this)

receiveModelCommand(Command)

receiveViewerCommand(Command)

createNewCommand(this)

execute(Command)

Command

importCommand()

exportCommand()

updateViewerCommand()

Questions

1. Will we be using actual .mp3, .mlv, etc. files and the information they contain, or creating our own objects?
2. Will this be a playlist editor or a media player?

Team Members

Jacob – For my part of the project, I will be working on the Model module of the M.V.C (Model View Control) design plan. My job will be to create objects and containers that will organize the different forms of media such underneath the classes of videos and music. My part of the code will have to be later integrated with the control module of the M.V.C design as the control module will both send commands and receive info from the Model to create or delete objects for music and video. For the objects, there will be the main objects or categories like movies and music and then sub-objects that can be accessed once the user has decided to go down into them by selecting movies for example, it would bring them to the sub-objects of movies which could be chosen to organize the movies by genre, date, director, date watched or by user-defined tags. For the music object, it will go down to genre, artist, alphabetical order or by user-defined tags. As for the user-defined tags, they could be anything from a “Home” playlist and “Workout” playlist to a tag like the date they listened to the songs. By adding the ability to sort by a user defined tag object, users will be able to personalize their playlists further than we could with just the typical categories to organize media.

Chris – My job for this project will be to create the graphic interface for the program. Also considered the viewer, the graphic interface will be what the user sees and where they can

control different aspects of the program. The viewer will “talk” to the controller by sending and receiving information when the user does a certain task or when something needs to be changed or updated. The viewer I create will be easy enough for the user to quickly understand but also contain all the features we have planned.

Ben – My job will be to create the control portion of the media manager. The Control Object will be responsible for passing information back and forth between the Viewer, the user interface, and the Model, the storage for all the information. The controller will handle all of the verbs, importing, exporting, create playlist etc. It will tell the other sections what information is required from them, and what they need to do.

Zack – My responsibility is project leader. I will write and organize most text related documents, such as project proposal updating, while the rest of the team focuses on programming. I will also be scheduling team meetings, making sure everyone’s work is getting done in a timely manner, and communicating project progress with the professor.

Technology

We will be using Java under the Eclipse environment.

Timeline

Our timeline will follow the course syllabus for the time being and will be updated on a week by week basis.

9/18 – Project proposal due

10/9 – Project deliverable 2

11/6 – Project deliverable 3

12/4 – Final project due

References

This will be filled in as time goes on.

Benefits/Risks

The major risk we would be concerned about are the scheduling of team meetings. Three of us are commuters. Cars could break down, a family or work issue could prevent us from getting to the school, and as the semester gets closer to winter the weather could be an issue. The project shouldn't suffer from any of this as we all have ways to communicate online and transfer files to each other. Two of us also have work related schedules that change on a weekly basis. Team meetings may not always be on a set schedule and will be made on a week by week basis.

Summary

Original

In short, this media manager will be a simple and easy to use program for a user that wants to simply view what they own. The playlist creating aspect is a fun but also smart way of organizing the user's content rather than having to mess with a larger piece of software such as Itunes. Example: A user wants to create a Halloween playlist. They input a season of American Horror Story, two Black Sabbath Albums, Nightmare on Elm Street the movie, and some photos from last year's Halloween party. They will be able to tag all these forms of media with phrases such as spooky, evil, Halloween, scary, etc. Then they could search for one of these tags, see all they media they tagged with it, and organize them into a playlist however they liked. Randomization will even be a fun option down the road. They can then save the playlist and two weeks later on Halloween, load it back up, make any changes, and they'd have a template to use when they go and setup Windows Media Center or ITunes for their party that night.

Final

Throughout the semester, the media manager project has become a playlist editor. Originally, we had mis-interpreted the project as something that would actually play media like ITunes. We had met for a team meeting or two and developed a plan to create this media player. We then re-read the initial project requirements and realized it was only a playlist editor. The project was heavily scaled back to reflect this, but because of the MVC framework, we did not have to change much in terms of pseudo code. The framework allows objects of any kind to be

passed back and forth between the Model, Viewer, and Controller. Instead of .mp3s and .mov files, they are just test or data objects.

Working on this project as a team proved to have some difficulties as time went on. Overall, our group worked well together and there wasn't too much difficulty in getting our deliverables done on time. However, as the project neared completion, it was becoming clear that the code was more complicated than what we were learning in class. I suppose you could say we were over ambitious. We had to downgrade our proposal yet again from total song organization with album art, album/song info, etc. to strictly importing songs and organizing them in a list. This simpler model became much more manageable and understandable for the whole team. In the last week of working on the project, we had troubles getting everyone's code working as we intended it to. We managed to finish the project with a GUI that can have songs imported and organized into a list. This could be expanded into a deeper playlist editor with other media, but it is a working song playlist editor that fills the projects requirements. Since it does have a user interface besides console, I'd say it more than fills them.

We definitely all had a good learning experience on working on a team. Looking back, I would've liked to schedule more face to face meetings. Also, sometimes it was tough getting the team members to respond in a timely matter when exchanging work mid-week. Next time, it would be valuable to get cellphone numbers or personal emails for additional contact.

Initial Code Outline

At the time of creating this second deliverable, I have been unable to get the JavaDoc tool working. I have included the basic class files that I would have put through the JavaDoc generator. As soon as I get the JavaDoc utility working properly, I will update everyone with a newer version of the project proposal.

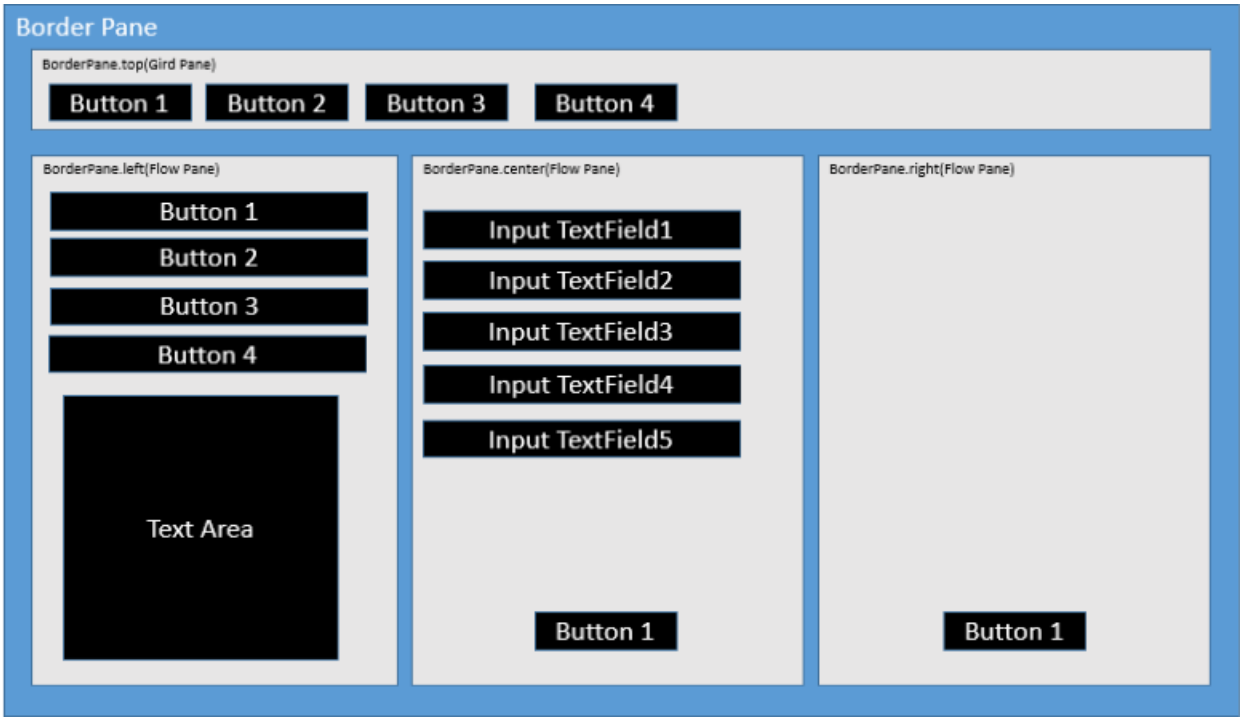
A large portion of our team meetings for this deliverable have been getting down the idea of an M.V.C. system. Ben is the brains behind using this system and has a decent amount more programming experience than us other three. It seems to be a little bit of a steep learning curve, especially because we just haven't been introduced to all the concepts we will need to use in the book yet. Much of our "documentation" has been drawn out graphs/charts. I feel as though we don't have much in terms of coding figured out yet, our planning pre code is solid and we will be ready to go in a short amount of time. I expect us to actually start coding in less than two weeks.

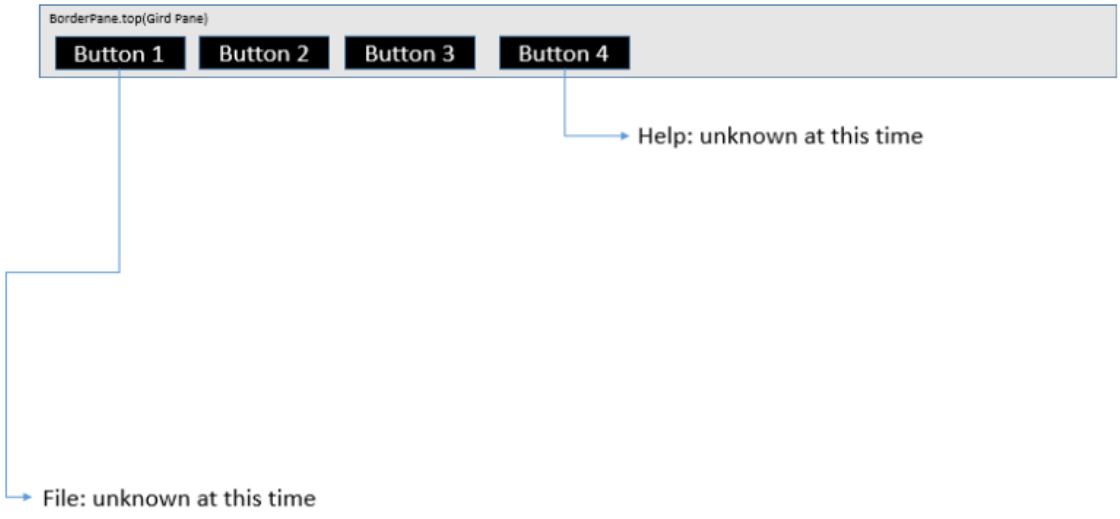
```
/**
 * The Model class
 *
 */
public class Model
{
	/**
	 * Sorts playlist by tag, video, song, etc.
	 */
	public void list()
	{

	}

}
```

```
/**
 * Class for the Viewer.
 * At the moment, we do not have all the details down for GUI programming.
 * These are just some basic placeholder names until the GUI is full planned
out
 * and we know what the actual Java functions are for buttons, lists, etc.
 *
 */
public class Viewer
{
	public void buttonClick()
	{

	}

	public void dropList()
	{

	}

	public void nextItem()
	{

	}

}
```

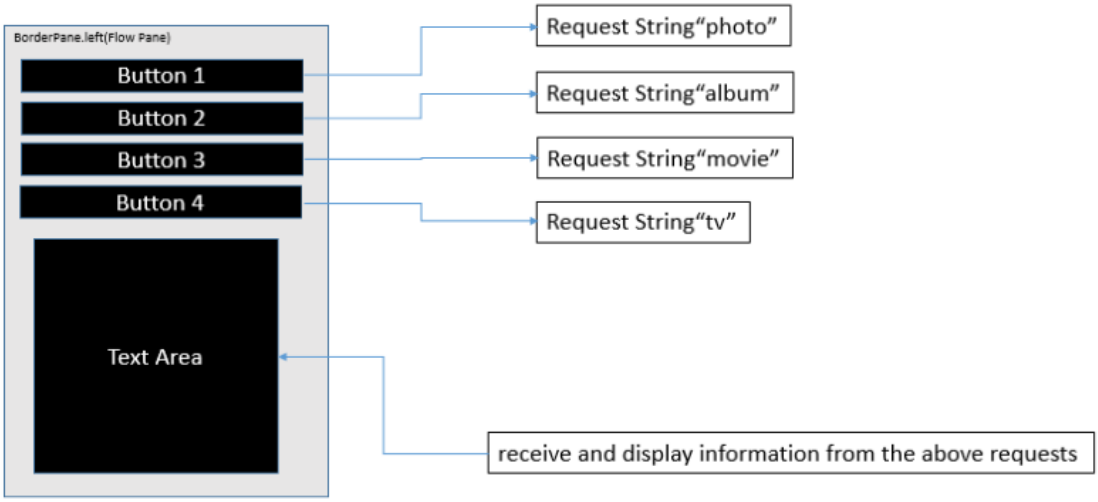
```
/**
 * The Media Controller class
 *
 */
public class MediaController
{

	/**
	 * Create a new Viewer object
	 */
	public void createViewer()
	{

	}

	/**
	 * Create a new Model object
	 */
	public void createModel()
	{

	}

	/**
	 * Read a command from the Model
	 */
	public void receiveModelCommand()
	{

	}

	/**
	 * Read a command from the Viewer
	 */
	public void receiveViewerCommand()
	{

	}

	/**
	 * Create a new command to send to the Model or the Viewer
	 */
	public void createNewCommand()
	{

	}

	/**
	 * Execute a command received from the Model or the Viewer
	 */
	public void execute()
```

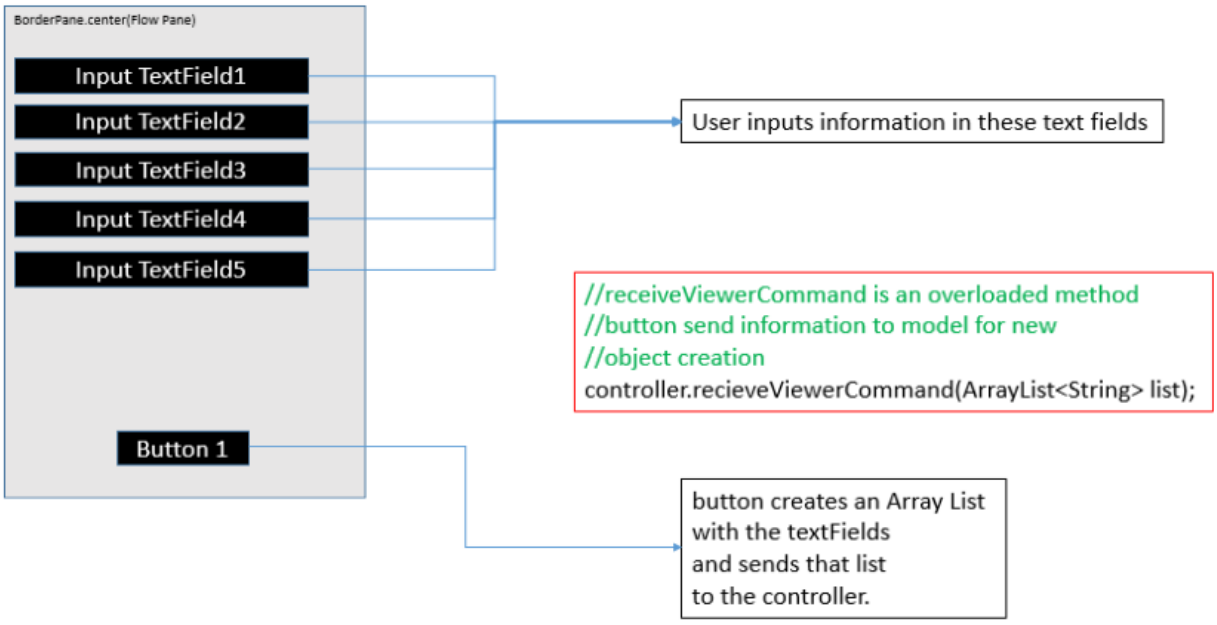
```
	{

	}
}



/**
 * The command class. A command moves between the Model, Viewer, and
Controller and
 * informs each object what to execute.
 */
public class Command
{
	/**
	 * Take in a new command
	 */
	public void importCommand()
	{

	}

	/**
	 * Send out a command
	 */
	public void exportCommand()
	{

	}

	/**
	 * Update the Viewer GUI for the user
	 */
	public void updateViewerCommand()
	{

	}
}
```


Border Pane
BorderPane.top(Gird Pane)
Button 1
Button 2
Button 3
Button 4
BorderPane.left(Flow Pane)
Button 1
Button 2
Button 3
Button 4
Text Area
BorderPane.center(Flow Pane)
Input TextField1
Input TextField2
Input TextField3
Input TextField4
Input TextField5
Button 1
BorderPane.right(Flow Pane)
Button 1


BorderPane.top(Gird Pane)
Button 1
Button 2
Button 3
Button 4
Help: unknown at this time
File: unknown at this time

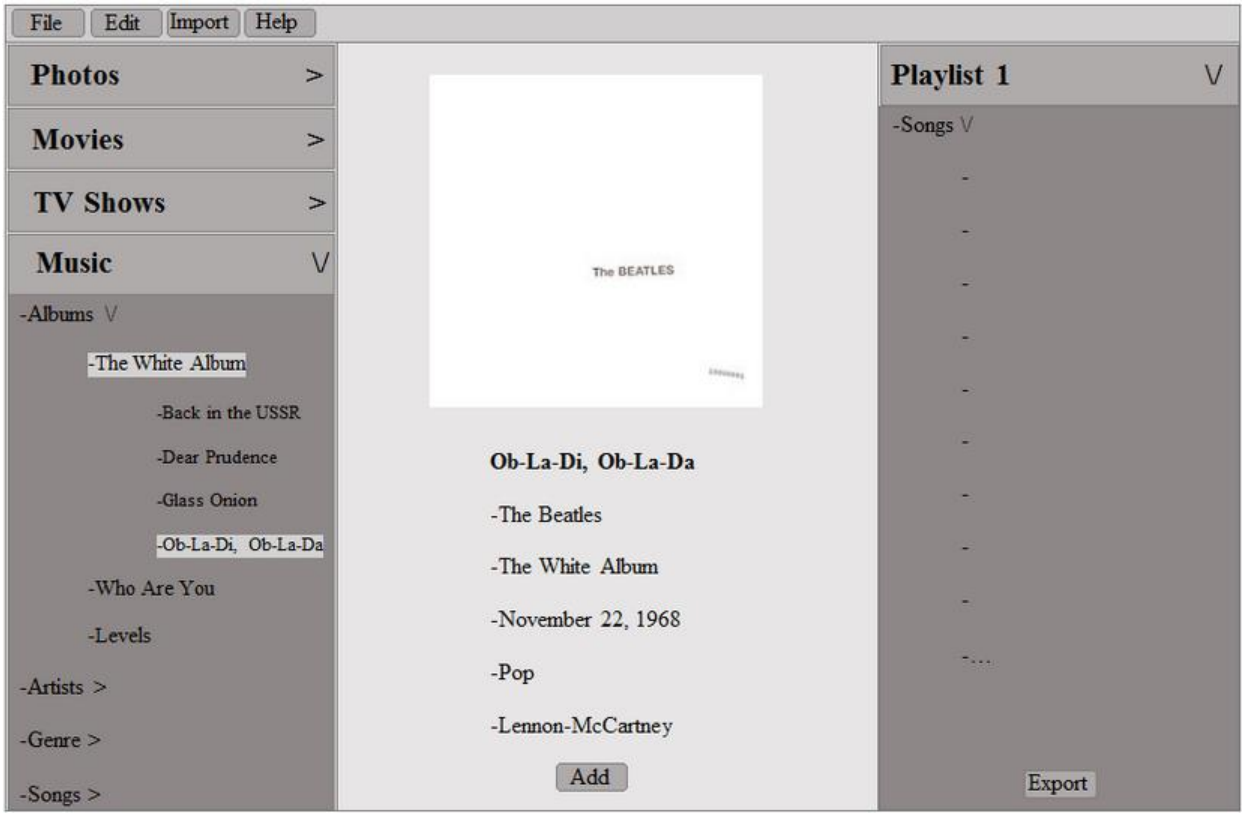
```
//receiveViewerCommand is an overloaded method
//each button request info to display
controller.recieveViewerCommand(String string);
```






```
//receiveViewerCommand is an overloaded method
//button send information to model for new
//object creation
controller.recieveViewerCommand(ArrayList<String> list);
```

```
//receive command allows controller to send you information
receiveCommand(Command command) {
        command.execute();
}
//constructor method takes controller in as a parameter
private Controller controller;
Class(Controller controller) {
        this.controller = controller;
}
```



Code

```
package FirstRun;


import javafx.application.Application;
import javafx.stage.Stage;


public class Run extends Application{
	public static void main(String[] args) {
		Application.launch(args);
	}


	@Override
	public void start(Stage primaryStage)  {
		MediaController mc1 = new MediaController();
```

```
	}
}

package FirstRun;

import Commands.Command;
import Commands.NewSongCommand;
import Commands.ShowSongCommand;
import Commands.ViewerCommand;
import Media.Song;



public class MediaController {

	private MediaViewer mv1;
	private MediaModel mm1;

	public MediaController() {
		this.mv1 = new MediaViewer(this);
		this.mm1 = new MediaModel(this);
	}

	public void receiveViewerCommand(String s1, String s2, String s3, String s4, String s5) {
//		System.out.println("receiveViewerCommand");
		Command mc1 = new NewSongCommand(s1, s2, s3, s4, s5);
		mc1.setMediaModel(mm1);
		mm1.commandFromController(mc1);
	}
```

```
	public void receiveViewerCommand(String i) {
		System.out.println("here");
		Command mc1 = new ShowSongCommand(this.mm1, this.mv1, i);
		mc1.execute();
	}

	public void updateSongList() {
//		System.out.println("updateSongList");
		Command vc1 = new ViewerCommand(mm1.getSongTitles());
		vc1.setMediaViewer(mv1);
		mv1.receiveControllerCommand(vc1);
	}
}

package FirstRun;

import java.util.ArrayList;

import Commands.Command;
import Media.Song;

public class MediaModel {

	private MediaController mc1;
	private ArrayList<Song> songs;
	private ArrayList<Song[]> playLists;


			public MediaModel(MediaController mc1) {
```

```
        this.mc1 = mc1;
        songs = new ArrayList<>();
}

public void commandFromController(Command command1){
        command1.execute();
}

public void addNewSong(Song song1) {
        this.songs.add(song1);
        mc1.updateSongList();
}

public String[] getSongTitles() {
        String[] songTitles = new String[songs.size()];

        for (int i = 0; i < this.songs.size(); i++) {
                Song x = songs.get(i);
                songTitles[i] = x.getTitle();
                System.out.println(x.getTitle());
        }

        return songTitles;
}

public Song getSong(String title) {
        for(Song e: songs) {
                if (e.getTitle().equalsIgnoreCase(title)) {
                        System.out.println("here");
```

```
                                        return e;
                                    }
                                }
                                return null;
                            }
}

package FirstRun;

        import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.FlowPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

import Commands.Command;
import Media.Song;
import Panes.LeftPane;
import Panes.RightPane;

        public class MediaViewer extends MediaPiece{
```

```
	private MediaController mc1;
	private LeftPane lp1;
	private RightPane rp1;


	public MediaViewer(MediaController mc1){

		this.mc1 = mc1;
		this.lp1 = new LeftPane(this);
		this.rp1 = new RightPane(this);

		Stage primaryStage = new Stage();
		BorderPane pane = new BorderPane();

		pane.setTop(new TopPane());
		pane.setLeft(this.lp1);
		pane.setCenter(new CenterPane(this));
		pane.setRight(this.rp1);

		Scene scene = new Scene(pane);
		primaryStage.setTitle("Media Viewer");
		primaryStage.setScene(scene);
		primaryStage.show();

	}

	public void createNewSong(String s1, String s2, String s3, String s4, String s5) {
//		System.out.println("createNewSong");
		mc1.receiveViewerCommand(s1, s2, s3, s4, s5);
```

```
	}

	public void receiveControllerCommand(Command command1) {
		System.out.println("receiveControllerCommand");
		command1.execute();
	}

	public void receiveSongDisplay(Song song) {
		rp1.listSongValues(song);
	}

	public LeftPane getLeftPane() {
		return this.lp1;
	}



	public void showSong(String i) {
		this.mc1.receiveViewerCommand(i);
	}



class CustomPane extends StackPane {
	public CustomPane(String title){
		getChildren().add(new Label(title));
		setStyle("-fx-border-color: Black");
		setPadding(new Insets(11.5, 12.5, 13.5, 14.5));

	}
}
```

```
class TopPane extends GridPane {
	public TopPane(){



		add(new Button("File"),0, 0);
		add(new Button("Edit"), 1, 0);
		add(new Button("Import"), 2, 0);
		add(new Button("Help"), 3, 0);
		//GridPane.setHallignment()


		setStyle("-fx-border-color: Black");
		setPadding(new Insets(11.5, 12.5, 13.5, 14.5));


	}
}



class CenterPane extends GridPane {
	private Button newSong;
	private MediaViewer mv1;
	private TextField tf1 = new TextField();
	private TextField tf2 = new TextField();
	private TextField tf3 = new TextField();
	private TextField tf4 = new TextField();
	private TextField tf5 = new TextField();
	public CenterPane(MediaViewer mv1){
		this.mv1 = mv1;
		setAlignment(Pos.CENTER);
```

```
                add(tf1,0, 0);
                add(tf2, 0, 1);
                add(tf3, 0, 2);
                add(tf4, 0, 3);
                add(tf5, 0, 4);

                this.newSong = new Button("Add");
                this.newSong.setOnAction(e -> this.mv1.createNewSong(tf1.getText(),
tf2.getText(),
                                tf3.getText(), tf4.getText(), tf5.getText()));

                add(newSong, 0, 8);

                setStyle("-fx-border-color: Black");
                setPadding(new Insets(11.5, 12.5, 13.5, 14.5));

            }
        }
}

package Commands;

import FirstRun.MediaModel;
import FirstRun.MediaViewer;


public abstract class Command {
```

```
	protected MediaViewer mv1;

	protected MediaModel mm1;


	public abstract void execute();


	public void setMediaModel(MediaModel mm1) {

		this.mm1 = mm1;

		System.out.println("setMediaModel");

	}

	public void setMediaViewer(MediaViewer mv1) {

		this.mv1 = mv1;

	}

}
```

```
package Commands;

import Media.Song;


public class NewSongCommand extends Command {

	private String s1;
	private String s2;
	private String s3;
	private String s4;
	private String s5;

	public NewSongCommand(String s1, String s2, String s3, String s4, String s5) {
		this.s1 = s1;
		this.s2 = s2;
		this.s3 = s3;
		this.s4 = s4;
		this.s5 = s5;
	}

	@Override
	public void execute() {
		Song song1 = new Song(this.s1, this.s2, this.s3, this.s4, this.s5);
//		System.out.println(mm1.toString());
		this.mm1.addNewSong(song1);
	}
```

```
}
package Commands;

import FirstRun.MediaModel;
import FirstRun.MediaViewer;
import Media.Song;

public class ShowSongCommand extends Command {

	private MediaModel mm1;
	private MediaViewer mv1;
	private String title;

	public ShowSongCommand(MediaModel mm1, MediaViewer mv1, String title) {
		this.mm1 = mm1;
		this.mv1 = mv1;
		this.title = title;
		System.out.println("in song command");
	}


	@Override
	public void execute() {

		mv1.receiveSongDisplay(this.mm1.getSong(this.title));
	}

}
```

```
package Commands;

public class ViewerCommand extends Command {

	private String[] songTitles;

	public ViewerCommand(String[] songTitles) {
		this.songTitles = songTitles;
	}

	@Override
	public void execute() {
		this.mv1.getLeftPane().setSongTitles(this.songTitles);
		this.mv1.getLeftPane().listSongTitles();
	}

}
```

```
package Media;

public class Media {

}
```

```
package Media;

public class Song extends Media {

	private String title;
	private String artist;
	private String genre;
	private String album;
	private String releaseDate;

	public Song(String s1, String s2, String s3, String s4, String s5) {
		System.out.println("song");
		this.title = s1;
		this.artist = s2;
		this.album = s3;
		this.genre = s4;
		this.releaseDate = s5;
	}

	public String getTitle() {
		return title;
	}

	public String getArtist() {
		return artist;
	}

	public String getGenre() {
		return genre;
	}
```

```
	public String getAlbum() {
		return album;
	}

	public String getReleaseDate() {
		return releaseDate;
	}


}
```

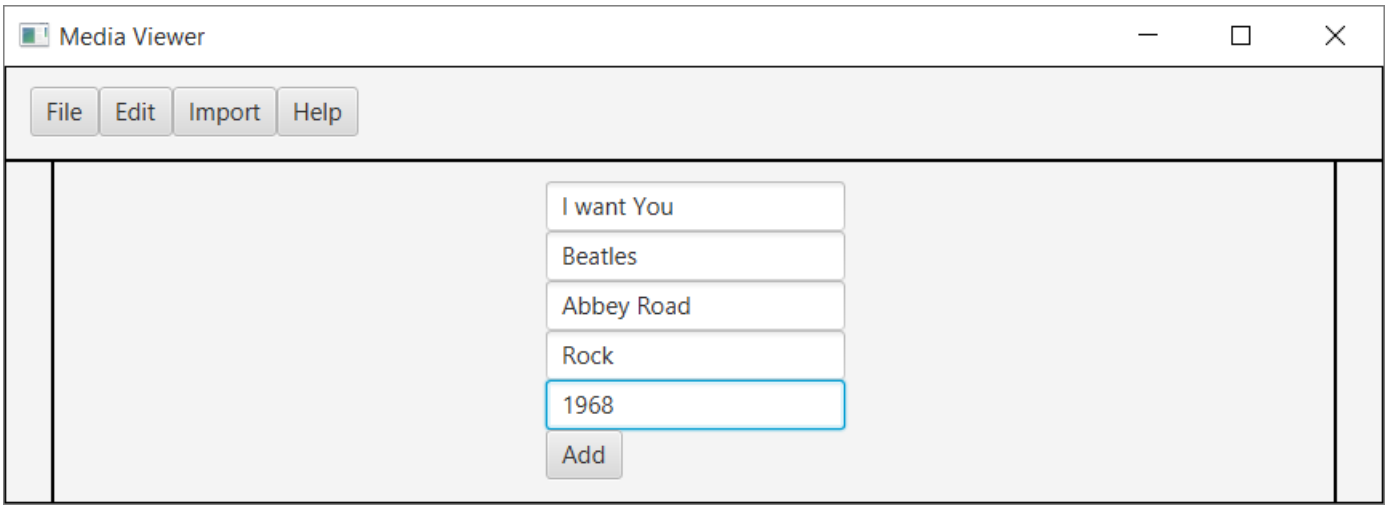
```
package Panes;


import Commands.Command;

import Commands.ShowSongCommand;

import FirstRun.MediaViewer;

import javafx.collections.ObservableList;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Node;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextArea;

import javafx.scene.layout.FlowPane;

import javafx.scene.layout.GridPane;

import javafx.scene.layout.VBox;


public class LeftPane extends GridPane {


	private String[] songTitles = new String[0];

	private MediaViewer mv1;
```

```
public LeftPane(MediaViewer mv1){
        this.mv1 = mv1;

        listSongTitles();
        setAlignment(Pos.CENTER);
        setStyle("-fx-border-color: Black");
        setPadding(new Insets(11.5, 12.5, 13.5, 14.5));

}

public void setSongTitles(String[] songTitles) {
        this.songTitles = songTitles;
}

public void listSongTitles() {
        getChildren().clear();
        for (int i = 0; i < songTitles.length; i ++) {
                int x = i;
                Button button = new Button("list");
                button.setOnAction(e -> listSong(songTitles[x]));
                add(new Label(songTitles[i]), 0, i);
                add(button, 1, i);
        }
}

public void listSong(String i) {
        System.out.println(i);
        mv1.showSong(i);
```

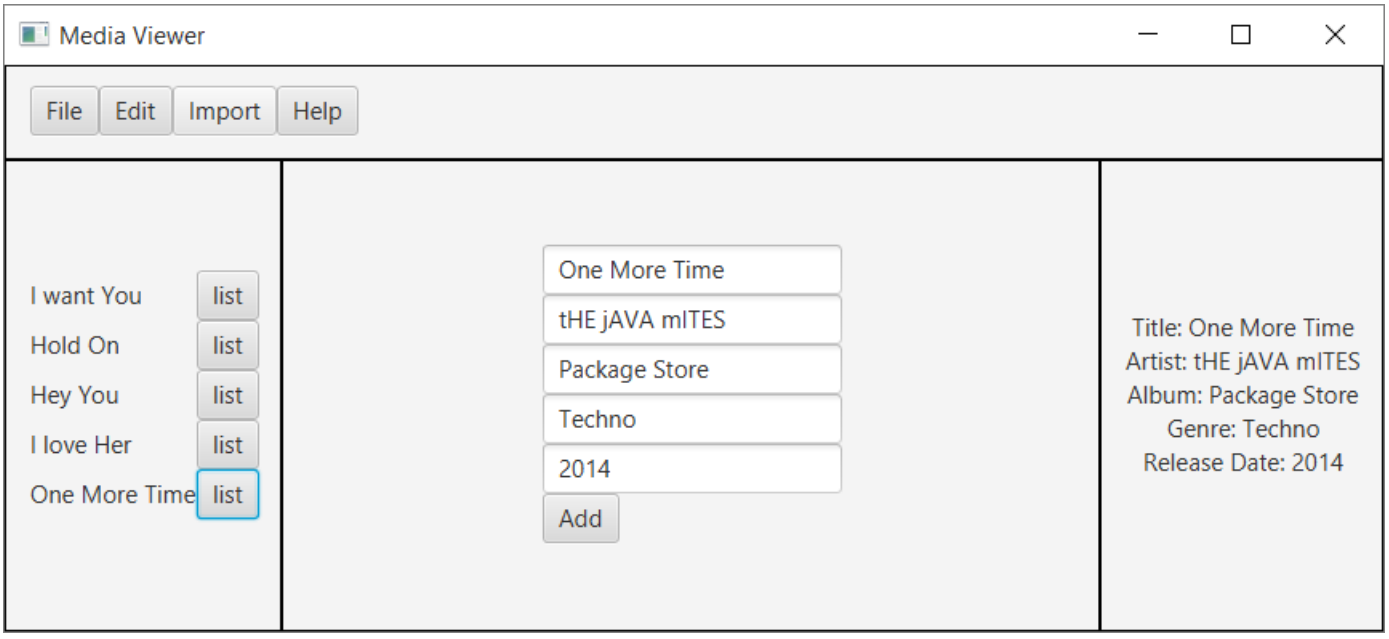
```
        }
}

package Panes;

import FirstRun.MediaViewer;
import Media.Song;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.layout.FlowPane;
import javafx.scene.layout.VBox;

public class RightPane extends VBox{



        private MediaViewer mv1;

        public RightPane(MediaViewer mv1){

                this.mv1 = mv1;
                setAlignment(Pos.CENTER);
                setStyle("-fx-border-color: Black");
                setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
        }
```

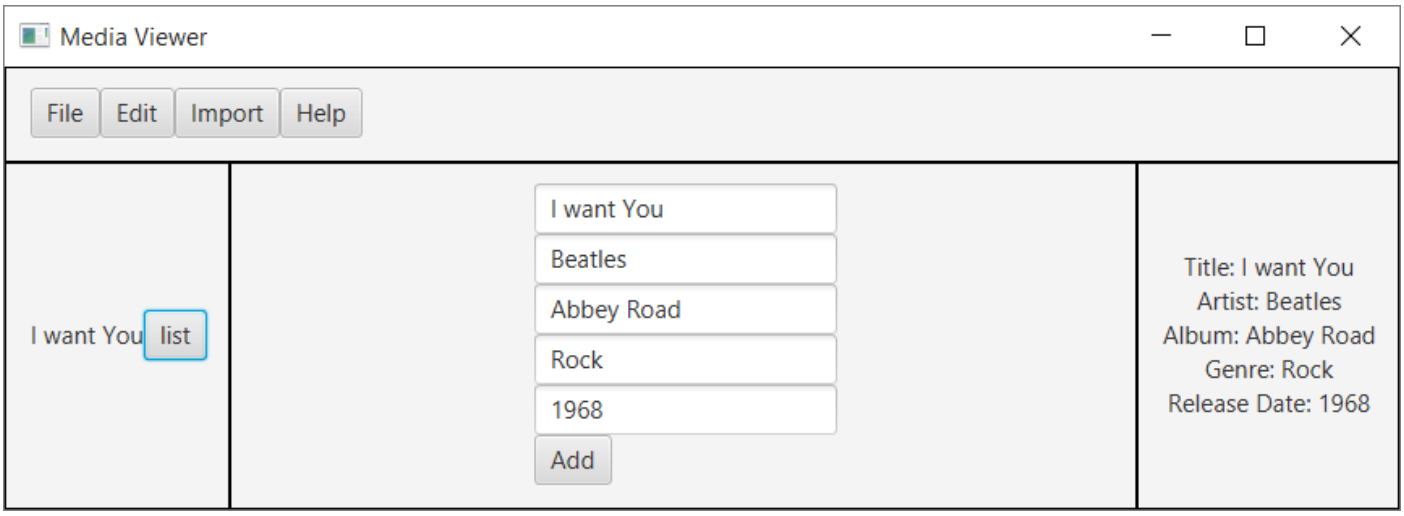
```
	public void listSongValues(Song song) {

		getChildren().clear();

		getChildren().add(new Label("Title: " + song.getTitle()));
		System.out.println(song.getTitle());
		getChildren().add(new Label("Artist: " + song.getArtist()));
		System.out.println(song.getArtist());
		getChildren().add(new Label("Album: " + song.getAlbum()));
		System.out.println(song.getAlbum());
		getChildren().add(new Label("Genre: " + song.getGenre()));
		System.out.println(song.getGenre());
		getChildren().add(new Label("Release Date: " + song.getReleaseDate()));
		System.out.println(song.getReleaseDate());

	}
}
```

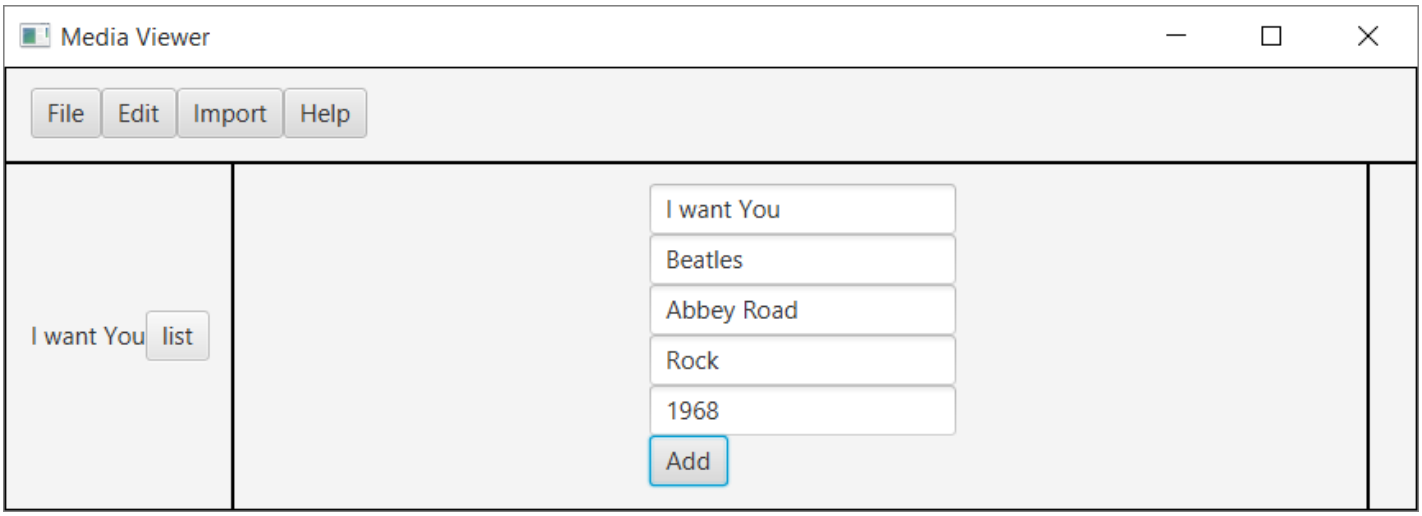
Final Program Screenshots


Media Viewer
File
Edit
Import
Help
I want You
Beatles
Abbey Road
Rock
1968
Add


Media Viewer
File
Edit
Import
Help
Add


Media Viewer
File
Edit
Import
Help
I want You
list
Hold On
list
Hey You
list
I love Her
list
One More Time
list
One More Time
tHE jAVA mITES
Package Store
Techno
2014
Add
Title: One More Time
Artist: tHE jAVA mITES
Album: Package Store
Genre: Techno
Release Date: 2014


Media Viewer
File
Edit
Import
Help
I want You
list
I want You
Beatles
Abbey Road
Rock
1968
Add
Title: I want You
Artist: Beatles
Album: Abbey Road
Genre: Rock
Release Date: 1968


Media Viewer
File
Edit
Import
Help
I want You
list
I want You
Beatles
Abbey Road
Rock
1968
Add