Homework 06 – Jukebox

Due: Day 24– April 4th, 2022 (beginning of class)

Objective:

The goal of this assignment is to gain experience with the new Views that we discussed this week (e.g., ImageView, Radio Group and Buttons, ProgressBar, SeekBar, and RatingBar) as well as learn about how to play sound/music using a new Android class: the MediaPlayer class.

Instructions:

For this assignment, assume that you have been hired by a company to work on its single-activity app, **Jukebox**, to be used at kiosks during public events to introduce new music to visitors. On Moodle, I have provided a starting Android project that I want you to use for implementing the required features described below. **Important**: The Views in the layout file have already been given an appropriate ID name for this homework. Therefore, do not add/modify/remove any of the Views in the layout file that I provided you. You will need to setup a View Binding in your Java controller file.

Your app needs to work as follows (i.e., the features/behavior listed below will be explicitly graded for this assignment).

- **Top Portion (Color Picker)**
  - The company would like for visitors who use the app to be able to customize the background color of the activity/screen to enhance the visitor's mood during their listening experience. At the top of the screen, you will find three seekbars whose minimum value is **0** and whose maximum value is **255**.
  - Many Views' classes have more specialized types of Listeners (i.e., interfaces) than the Listener types we have looked at so far. The SeekBar class has its own Listener type named **SeekBar.OnSeekBarChangeListener** that you will need to use for SeekBars. Note: The only callback method that you will add code to for with this Listener is the **onProgressChanged( )** method.
  - When the visitor slides a SeekBar, the TextView on the right should be updated do display the current setting (i.e., a value between 0 and 255)
  - To change the background color of the activity/screen, you will need to find the appropriate attribute of the **ConstraintLayout** element and assign it the color produced from the seekbars' settings. To create the Color integer, you will need to use a different method in the Color class than the .parseColor( ) method that we demonstrated earlier this semester. Instead, you will need to find the appropriate method that can construct a Color integer using only the red, green, and blue (**rgb**) values that range from 0 to 255.
  - Tip: To test that this portion is working correctly, if all of the seekbars are to the left (0), then the background color of the activity will be black. If all of the seekbars are to the right (255), then the background color of the activity will be white. Adjust the seek bars for different levels for red, green, and blue and confirm that the color matches the seek bar's settings.
- **Middle Portion (Music Player)**
  - When the activity first opens, it should begin playing whichever song is selected in the RadioGroup and display their album cover in the ImageView. The album covers for each song are in the project's resource >> drawable folder.
    - **Important**: When I test your program, I will randomly be selecting different starting songs so your code should not be written to assume that the current starting song will always be the starting song. Look at the methods in the RadioGroup class to make your code flexible to work with any starting song
  - To play music/songs In Android, you will need to construct a **Media Player** object. You should create a member variable (field) in your Activity class to store a reference to the Media Player object. Once you

have constructed a Media Player object, you will then need to call its **create**( ) method. As you will notice, one of the parameters is the resource ID of the audio file you wish to play. I have placed the three audio files in the project's resource >> raw folder and therefore, their resource ID will be of the form: **R.raw._____** . Next, to start playing the music/song, you call its **start**( ) method. Be sure your computer's volume and the AVD's volume are not muted and you should hear the song's audio playing.

- The company also wishes to provide visitors with the ability to "seek" (i.e., drag the seek bar) to any point in the current song to begin playing from. The seek bar's minimum value is **0** (i.e., 0% of song) and its maximum value is **100** (i.e., 100% of song). Explore the methods in the Media Player class to determine how to "seek" to a specified point in the song and continue playing.
  - **Note**: The Seek Bar should not automatically update/move as the song plays, it is only for a visitor to drag the slider to a particular point in the song to begin playing from.
- When the user selects a different song in the RadioGroup, the Media Player should automatically switch to playing the newly-selected song and display the album cover in the ImageView.. Recall, many Views' classes have more specialized types of Listeners (i.e., interfaces) than the Listener types we have looked at so far. The RadioGroup class has its own Listener type named **RadioGroup.OnCheckedChangeListener** that you must use for this assignment. The **onCheckedChanged** callback method provides the resource ID of the radio button that was most recently selected that you can use to compare with the IDs of the Views (e.g., **R.id._____**)

- **Bottom Portion (Vote Casting)**
  - Finally, the company would like for a visitor to select a 1-5 star rating for the currently selected/playing song and cast their vote
  - Each song has a progress bar whose minimum value is **0** and whose maximum value is **5**. The progress bar shows the current "average number of stars" based upon the votes that have been casted for that song so far. The "average number of stars" should be updated each time a new rating is casted by using the running average technique
  - Each time the Cast Vote button is pressed:
    - The progress bar for the currently selected/playing song should be updated by using the running average technique previously discussed
      - Important: Do not create a data structure / array to store each song's rating values – you will lose a considerable amount of the points
    - The number of votes for the currently selected/playing song should be incremented by 1
    - The star rating should be cleared for the next visitor

Submission:

When you are finished, you must **zip** your Android Studio project. On Moodle, you should upload your zip file to the Homework 06 assignment box