Bluegrass Community and Technical College
CIT 149: Java I
Chapter 6 - Radio

This week you will create your first Java GUI. It will simulate an old car radio (and a few other features).

Attached to this assignment in Blackboard is a zipped file, named **FiveSongs.zip**, which contains 5 audio clips. These will be the "songs" that play on the 5 stations of the simulated radio (think of the image to the right of an old car radio).

To complete this GUI application, you may start with the **JukeBox** and **JukeBoxControl** classes found in the **JukeBox.zip** file in Blackboard. Your goal is to create an application (**using only 1 panel: DO NOT CREATE MULTIPLE PANELS**) that simulates an old car console. Follow the guidelines below.

(1) Radio:

Use a set of radio buttons to allow the user to select a radio station. The **JukeBox** and **JukeBoxControls** classes in the chapter example used a combo box to select clips to play. You will instead use radio buttons to select a station (song). Instead of having a knob on the console to turn on/off the radio, we will use an addition radio button to turn the radio off. For example the radio buttons will be:

- Turn off the radio
- Classical Station
- Hitchcock Station
- New Age Station
- Country Western Station
- 80s Station

When the first radio button is clicked, you should simulate the radio being turned off. Make sure you stop any sound clips which are playing. When another radio button is selected, make sure you any other station's song is stopped before playing the new song. It is possible to play 2 sound clips at once in Java – we do not want this. **The program should start with the radio off** (first button pre-selected and not audio clip running).

(2) Windshield Wiper control:

Add a checkbox to the GUI. It will simulate turning the windshield wipers on and off. Start with the checkbox not selected (no check mark). Add a JLabel component in the panel. When the GUI starts, the JLabel text will say "Windshield Wipers OFF".

If the user clicks on the checkbox, the user should now see a checkmark in the checkbox. The message in the JLabel component should now read: "Windshield Wipers ON".

If the checkbox is clicked again, the checkmark goes away and the message will indicate the wipers are OFF.

(3) Radio Volume Control:

Create another JLabel object which read "Radio Volume". Use a slider to **simulate** controlling the volume of the radio. The slider should have 4 settings (1-4 representing lowest to highest volume.) You will not actually manipulate the volume of the songs playing. You will instead simulate the volume by increasing/decreasing the font size of the "Radio Volume" label. For example, if the volume is set at 1, make the font size 8. If the volume is 2, set the font size to 10. If the volume is set to 3, use a font size of 12 and the volume set to 4 will use a font size of 14. The slider should only have 4 settings (ticks).

Since the radio starts with the radio off, this JLabel component should not be visible (because there is no volume when the radio is off). It will only be shown when the radio is ON. HINT: You can control the visibility of this label when the radio button is changed.

If you have a JLabel component named volumeLabel, then volumeLabel.setVisible(true); will cause it to appear in the GUI and volumeLabel.setVisible(false); will cause it to "disappear".

IT WILL BE EASIER TO COMPLETE THIS IN STEPS. Complete the work for (1) above. When it works, add the components to the panel for (2) above. Test that and when it works, add the components for (3) above. This can all be done in one panel. Since we are just learning how to use GUIs, do not complicate the assignment by creating multiple panels. You may use a Layout Manager if you want but it is not required for this assignment.

DO NOT forget to document your programs === Including the ones I provided === Update the documentation (you name, date, etc.) at the top of the Java files.

This is an exercise in using addition controls and learning how to use listeners with those objects. You will notice, this assignment does not use JButton objects.

USING IDEs such as Eclipse, NetBeans, etc:

Some IDEs require external resources, such as sound file and images, to be placed within a specific folder within your Java project folder. For example, Eclipse likes to use:

ProjectnameFolder>>src>>resources

Turning in your work:

Zip all of the JAVA files for grading, including the sound files. Do not zip a collection of folders such as are created with Eclipse and Netbeans --- only zip the .java files and sound files. That will make grading easier and I won't have to copy these files into your folder before grading. In other words, every file that I need to grade your homework should be in the zip file. Points will be deducted for missing or extra files.