Audio Visualizer

Description: Create an audio visualizer for the AudioViz media player application in Java using JavaFX based on the Visualizer interface.

Purpose: This challenge provides experience in creating a class based on an interface, working with updates based on events, and programming using the JavaFX platform.

Requirements:

Test application: AudioViz (provided with challenge as AudioViz.zip)
Language: Java 8 SE
Platform: JavaFX
Interface class is to implement: Visualizer

The AudioViz application provided in AudioViz(1).zip is the test application for the audio visualizer you are to build. AudioViz also contains the Visualizer interface that is to be implemented by the visualizer class as well as three visualizers examples: EllipseVisualizer1, EllipseVisualizer2, and EllipseVisualizer3.

Develop a class based on the Visualizer interface that displays a visual experience in the vizPane AnchorPane based on the data provided to the update() method.

The visualizer class is to use the following naming convention. The first part of the class name is to be your pawprint. The remaining part of the class name is your choice. The name must use camel-case and, as with all classes, the first character of the class name must be upper-case. For example, if the pawprint is abcxyz9 and the other part of the class name is SuperVisual, the following is the name for the class: Abcxyz9SuperVisual.

If you include additional files with your project such as images, the names of the files must also begin with your pawprint following the same rules as above. So, for example, if you include an image that would normally be named image1.png and your pawprint is abcxyz9, then the image is to be named Abcxyz9Image1.png.

One of the methods that must be implemented in the visualizer class is getName() that returns a human-readable name as a String for the visualizer that is used to list it in the Visualizers menu and is placed in a text object below the visualizer by the AudioViz application. The name that is returned is to include your pawprint as the first part of the name. Example: Abcxyz Super Visual

To test your visualizer class it needs to be added to the visualizers ArrayList in the initialize() method of the PlayerController class provided in the AudioViz application.

The following is a segment of the existing code that is in initialize():

```
      visualizers = new ArrayList<>();
      visualizers.add(new EllipseVisualizer1());
      visualizers.add(new EllipseVisualizer2());
      visualizers.add(new EllipseVisualizer3());
```

The visualizer you create is to be added by creating an instance of your visualizer class and adding it to visualizers. Using the class name example from above of Abcxyz9SuperVisual the line of code to be added is:

```
      visualizers.add(new Abcxyz9SuperVisual());
```

The following is the Visualizer interface that your visualizer class must implement.

```
public interface Visualizer {
      public void start(Integer numBands, AnchorPane vizPane);
      public void end();
      public String getName();
      public void update(double timestamp, double duration,
float[] magnitudes, float[] phases);
}
```

The start() method is called to give your visualizer an opportunity to prepare the objects and visual interface for your visualizer. It is in this method where you want to create the visual elements that you want to use in your visualizer. Note that start() may get called multiple times without an associated end() being called. If start() is called and you have previously done setup you need to do whatever is necessary to not duplicate() objects or have memory leaks if start() is called again. start() is called each time something has changed that may require a different setup such as a different number of bands being specified. The numBands value provided as a parameter to start() indicates the number of bands that are used in the analysis that will provide the magnitudes and phases in the update() method. The vizPane parameter is a reference to the AnchorPane in which your visualizer is to render its visualization. The dimensions of vizPane are to be used in determining how the visualization is to be laid out. The visualization is to happen inside of vizPane.

The end() method is called to give your visualizer an opportunity to cleanup the objects and interface elements that were created in start() once it is no longer needing those objects and interface elements (because, for example, AudioViz has switched from your visualizer to another visualizer). The end() method gives your visualizer the ability to clean up after it is done. Any elements added to the vizPane are to be removed and any other changes to vizPane are to be undone. When another visualizer gets control of the vizPane it should find it ready to use. end() is also the place to remove any objects that were created in start() that are no longer needed.

The getName() method, as previously mentioned, is to return a human readable name for your visualizer as a String. Follow the rules specified previously for the name.

The update() method is called repeatedly to give your visualizer information about the audio that is playing in the form of an array of magnitudes and an array of phases. There is a magnitude and a phase for each band. This information is to be used to make the visual changes occur in the visualizer. Note that when the number of bands is changed (by selecting the number of bands in the UI) there may be a period of time between when update() delivers data using the previous number of bands and the new number of bands. You need to use the actual lengths of magnitudes and phases arrays and the lengths of any other arrays or collections your visualizer uses to make sure you don't attempt to access off the end of an array or collection.

Testing

Once you have your visualizer written you need to test it to make sure it handles various situations properly. Make sure that it continues to work under the following circumstances.

- Verify that changing the number of bands using the Bands menu works.
- Verify that switching to your visualizers from another visualizer works.
- Verify that switching from your visualizer to another visualizer works.
- Verify that switching to a new song from another song while your visualizers is selected works.

Submission

You are to submit a zip file of the AudioViz application that contains your visualizer. Create a zip file of the AudioViz NetBeans project folder. The zip file of the folder will be called AudioViz.zip if you zip the NetBeans project folder. Once you have the AudioViz.zip file rename it to include your pawprint as the first part of the zip file name. For example if the pawprint is abcxyz9 rename the zip file to Abcxyz9AudioViz.zip. The zip file whose filename contains your pawprint is to be submitted as the challenge submission.

Important!

You are to create your own visualizer. Do not simply copy the example visualizers and make minor changes. You are to build your own visualizer by coding it from the beginning based on the Visualizer interface and the notes provided in this challenge. You will not learn how to be a programmer by taking the example visualizer code and simply modifying it to meet the challenge. The examples do serve as a reference to help you through the challenge, but should not be used in a way where you are just duplicating the code in them and making changes to that code.

Be creative! Do something that is uniquely yours. You can use all types of visual elements. I chose to create my visualizers based on ellipses and to line them up side by side. Your visualizer can use completely different objects and they can be arranged in any way you choose. How you manipulate the objects in your visualizer based on the amplitudes and phases can be anything you choose. Do something interesting and fun!