Computer Programming – CS 6011
Lecture 8: Polymorphism - Project related

Fall 2023

Topics

- Controlling Access to Class Members
- Polymorphism
- Synthesizer
- JavaFX

Controlling Access to Members of a Class

- Public
- Private
- Protected

Modifier	Class	Subclass	Other Classes
public	Y	Y	Y
protected	Y	Y	N
private	Y	N	N

Inheritance Example

- As an example, IOException is a more specific version of Exception
- And FileNotFoundException is an even more specific version of IOException…
- In Java (and C++), we say IOException inherits from Exception.
  - And FileNotFoundException inherits from IOException (and thus from Exception).

Polymorphism

How to figure out which function to call?

- System.out.println (Car);
- System.out.println(myFraction);

- public void println(Object Obj) //pseudocode for println

```
{
   PrintString( Obj.toString() );
}
```

Polymorphism

- Polymorphism (many shapes) means that when we call a method, we don't actually know exactly which version of that method will be called:

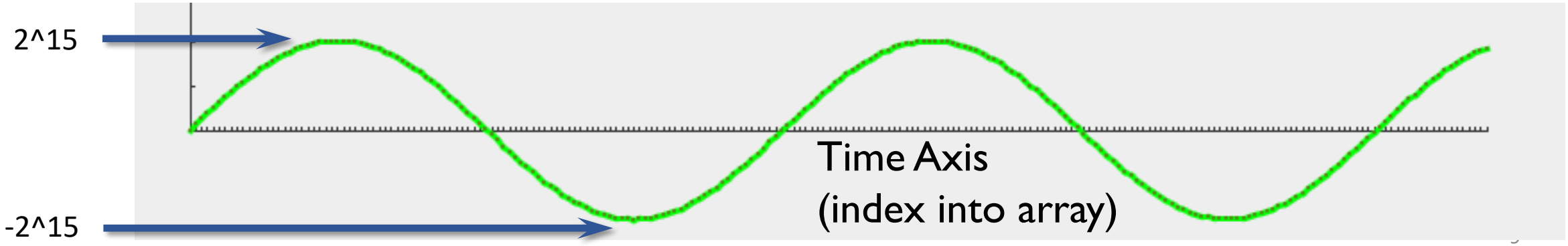
```
Employee emp1 = new Employee ();
Employee emp2 = new SalariedEmployee ();
emp1.showDetails();
emp2.showDetails();
```

- `emp1.showDetails();` // Which implementation of showDetails() is being called?
  - It might call Employee.showDetails(), or SalariedEmployee.showDetails().
  - This is a good thing! The language itself figures it out for us and calls the correct function automatically.

Project Related

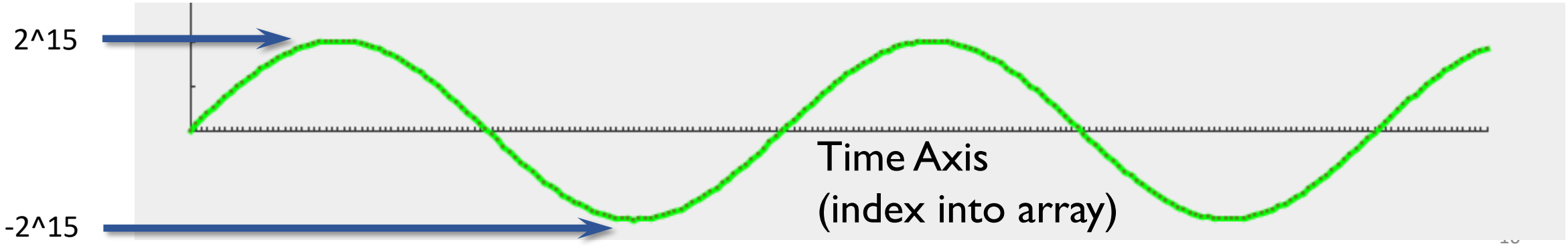
What is an `AudioClip`?

- Class that represents a “sound wave”
- Composed of a bunch of samples.
  - Each sample measures the amplitude (strength, y value) of the wave at a given point in time (ie: the x value – or think of this as the index of the sample).
  - What is the maximum amplitude that we are using for our audio waves?
    - 2^15 (32767), and the minimum value?
    - -2^15



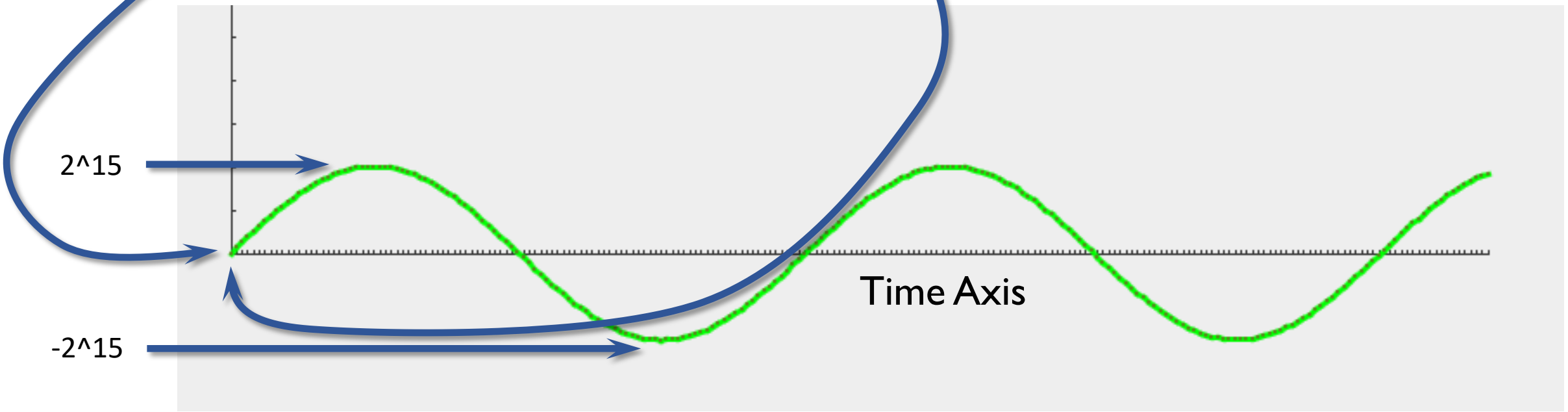
Audio Clip

- To store a (single) number with range 2^15 to -2^15, what type do we use?
  - Short (16 bits)
    - Each sample is stored in a Short



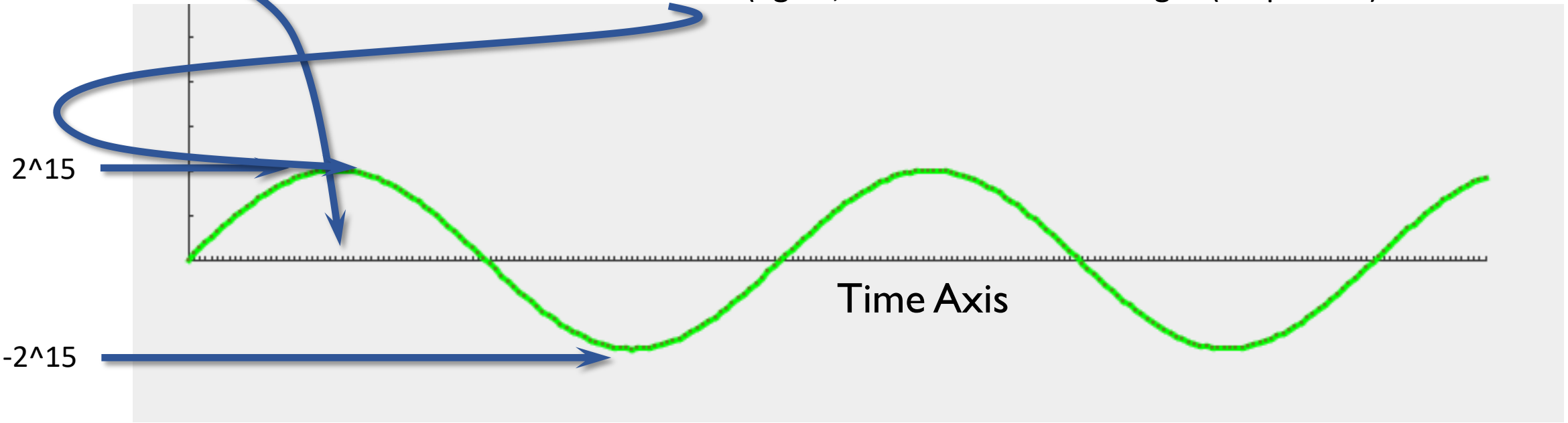
Audio Clip

- Composed of a bunch of samples.
  - Each sample measures the amplitude (strength, y value) of the wave at a given point in time (x value).
  - At time 0 (which corresponds to sample 0)
    - a sine wave has a value (strength, amplitude) of 0.



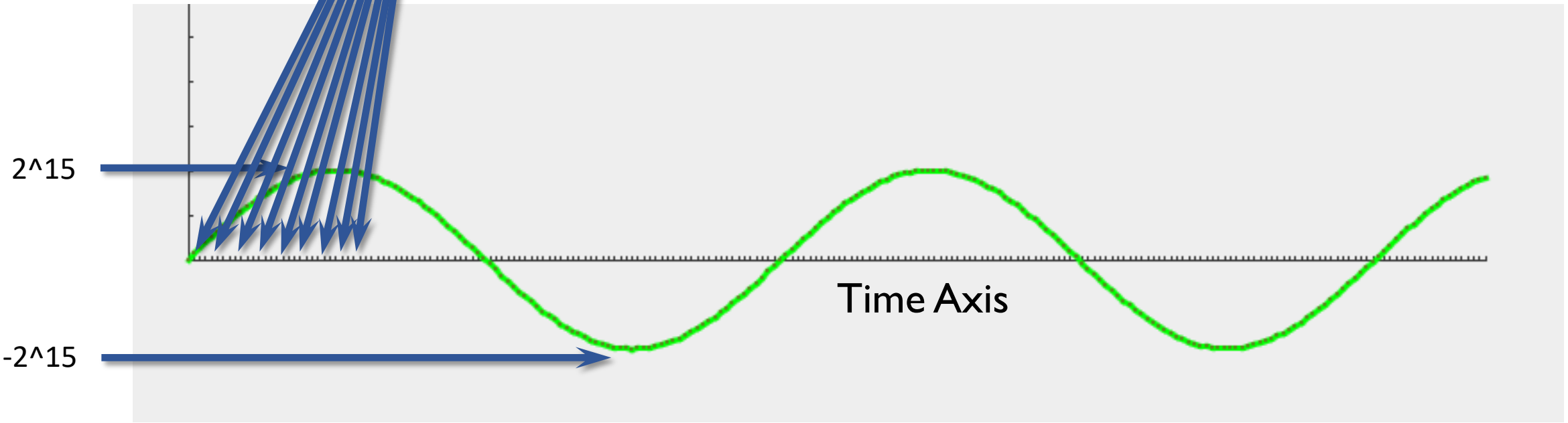
Audio Clip

- Composed of a bunch of samples.
  - Each sample measures the amplitude (strength, y value) of the wave at a given point in time (x value).
  - At time 0.1 (which corresponds to sample 300) [Note, the actual values depend on wave frequency.]
    - a sine wave has a value of 32767 (again, the value is the strength (amplitude) of the wave



Audio Clip

- Composed of a bunch of samples.
  - How many samples (x values)?
    - 44100?
    - 44100 P[illegible] SECOND!
    - So 2 s[illegible]nds would give 88200 samples!

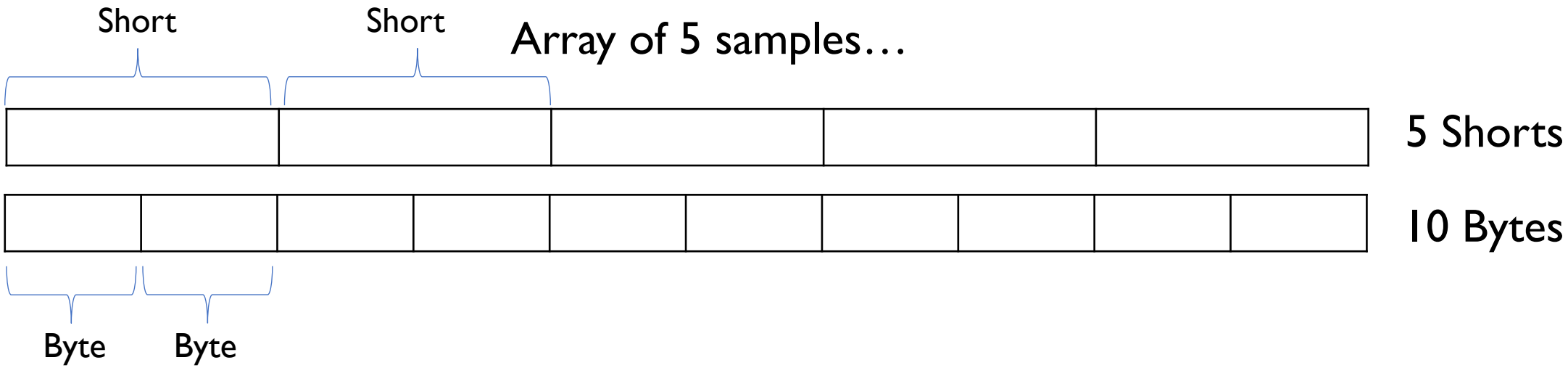


Audio Clip

- A single sample (valued between -2^15 and 2^15) is stored in a Short.
- How many bytes are in a Short?
  - 2 bytes / 16 bits
- Due to the java audio library, the data for a sound wave must be in the form of an array of bytes.
  - Thus we are storing an array of bytes to hold our samples.
  - `byte [] data = new byte[ ??? ];` // But how many bytes?

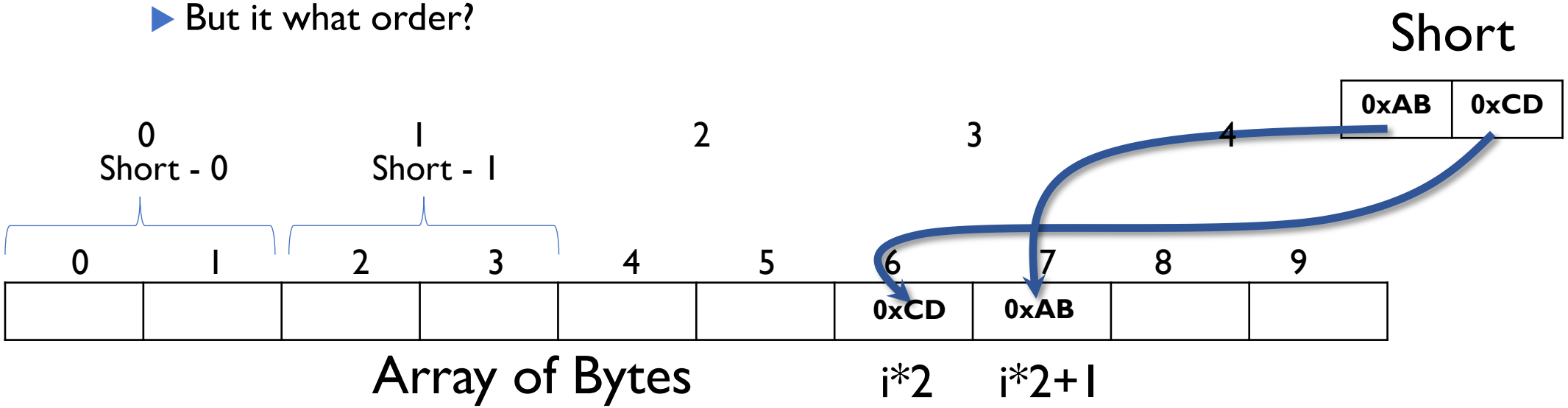
Audio Clip

- How many samples are we storing again?
  - Duration * RATE [Say 2 seconds * 44100 => 88200]
  - Each individual sample is stored as a?
    - Short [Because each individual sample can have a value between -2^15 and 2^15]
  - How many bytes are necessary to store 88200 samples?
  - In other words, how many bytes are necessary to store 88200 Shorts?
- Well, each Short is comprised of 2 bytes, thus we need 2 * 88200 bytes.

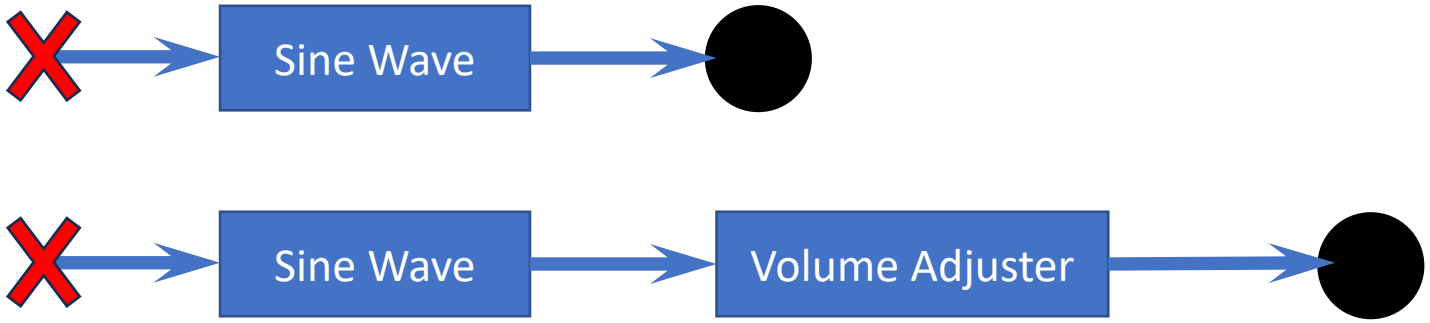


Short to Byte in Array

- Set the i^{th} Short's position in an array of bytes to the value in that short.
  - Which bytes in array represent the i^{th} short?
    - `i*2, i*2+1`
    - So if we are looking at the `4th` Short (Index == 3), which bytes are we going to use?
      - 6 and 7
    - But it what order?



Audio Components



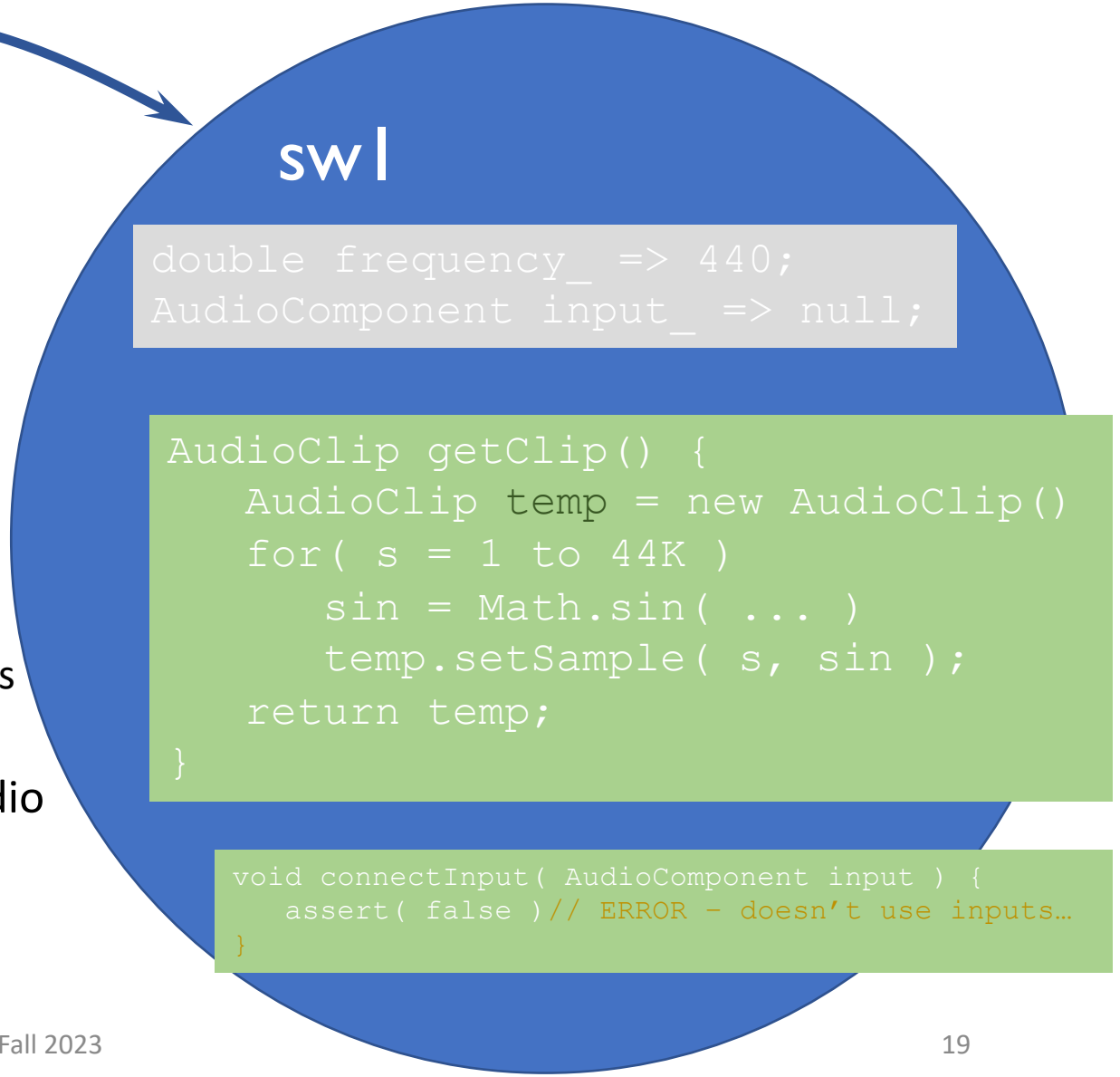
Audio Components

- The purpose of an Audio Component is to generate (or at least provide) an Audio Clip.
- An Audio Component can be hooked directly to the speaker and a sound will be played.
  - Specifically, the speaker will ask the Audio Component for its clip, and then play that.
    - And the Audio Component will ask its input (another Audio Component) for its clip, etc…
- Audio Component examples:
  - A Sine Wave Component generates a sine wave so is an Audio Component.
    - Remember, the actual Audio Clip which is the sound that will be played, is “stored” in the Audio Component.
  - A Mixer Component generates a wave that can be played (and thus is an Audio Component).
- The difference between the Sine Wave Component and the Mixer Component is that the mixer must have inputs that it will mix together.
  - The Sine Wave does not need any (sound) inputs as it generates the sound itself.

Audio Components

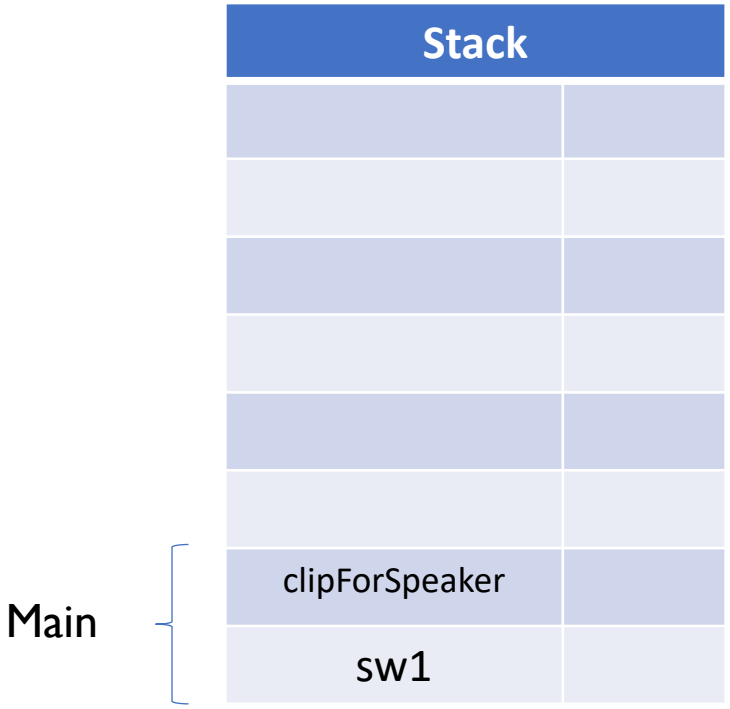
```
SineWaveAC sw1 = new SineWaveAC( 440 );
```

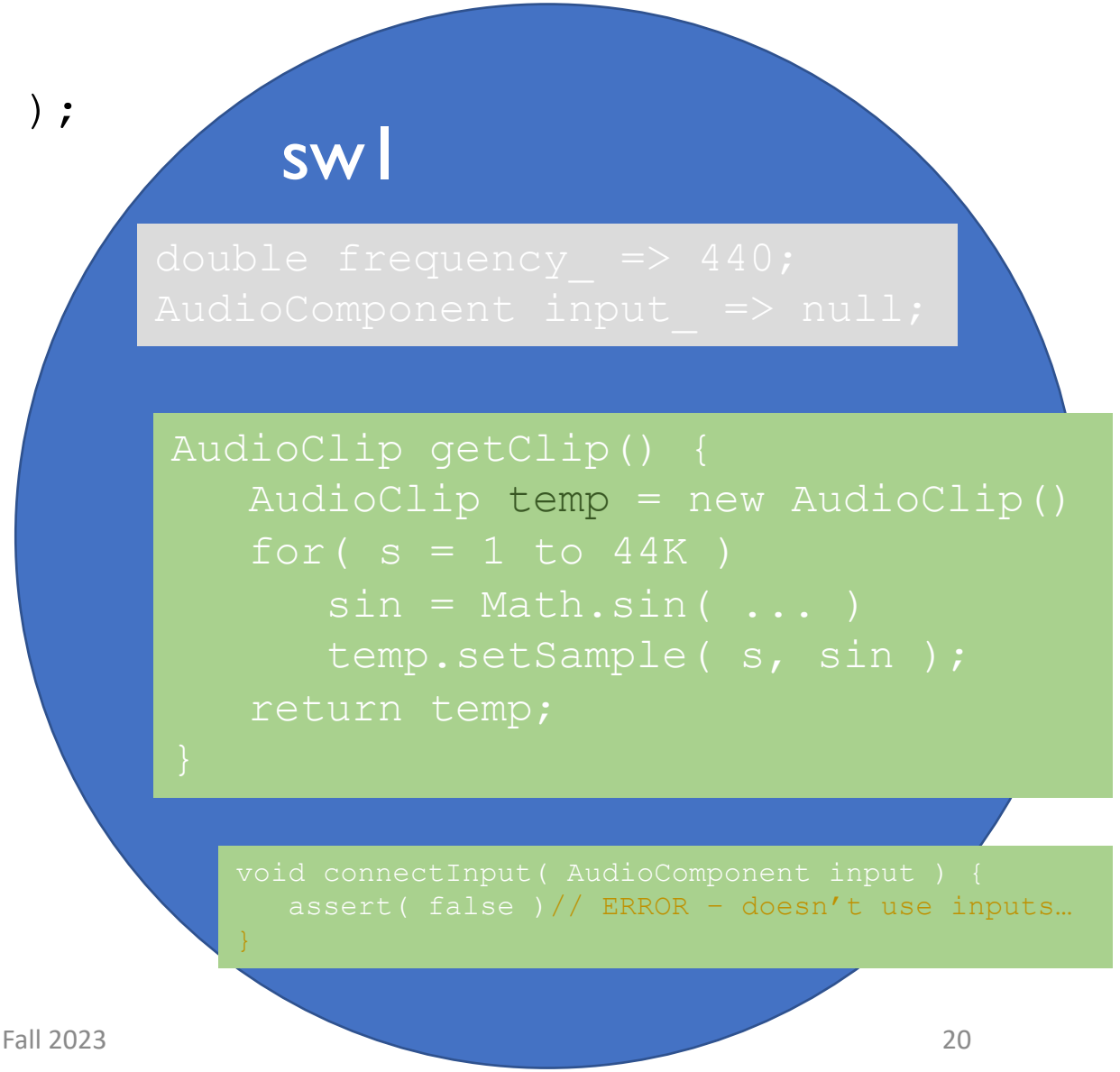
- Data stored in `sw1`?
  - frequency and input. (Values?)
- SineWave does not store an AudioClip because it will *generate* a new one every time someone asks for it - using getClip().
- `temp` clip starts with all its data (`byte [] data`) initialized to 0s.
  - Calculate each of 88200 (2 seconds at 44100 Hz) samples using math.
- Since the Sine Wave Audio Component generates a audio wave (AudioClip) only when it is requested, it does not have (or allow) any other audio input.



Audio Components

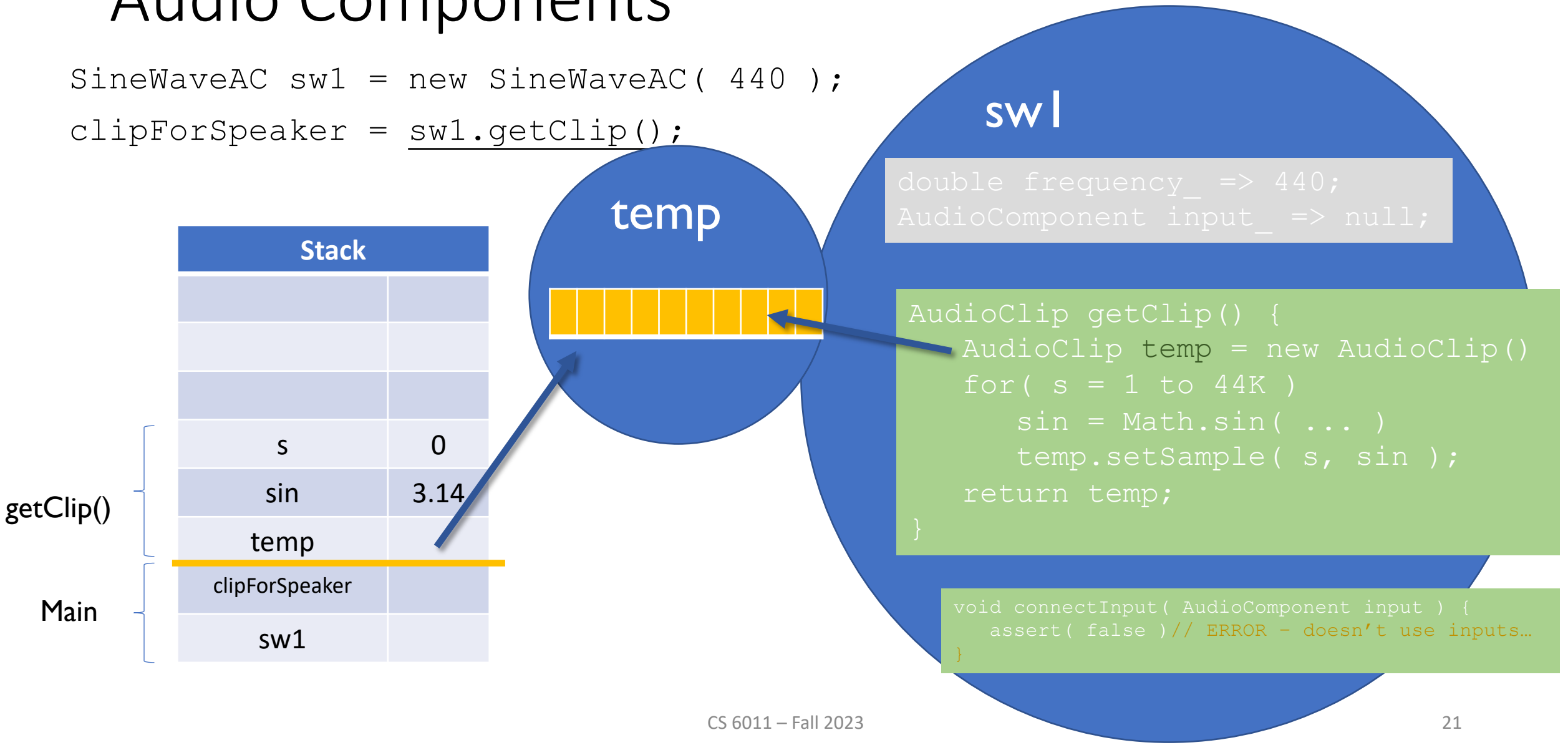
```
SineWaveAC sw1 = new SineWaveAC( 440 );
clipForSpeaker = sw1.getClip();
```





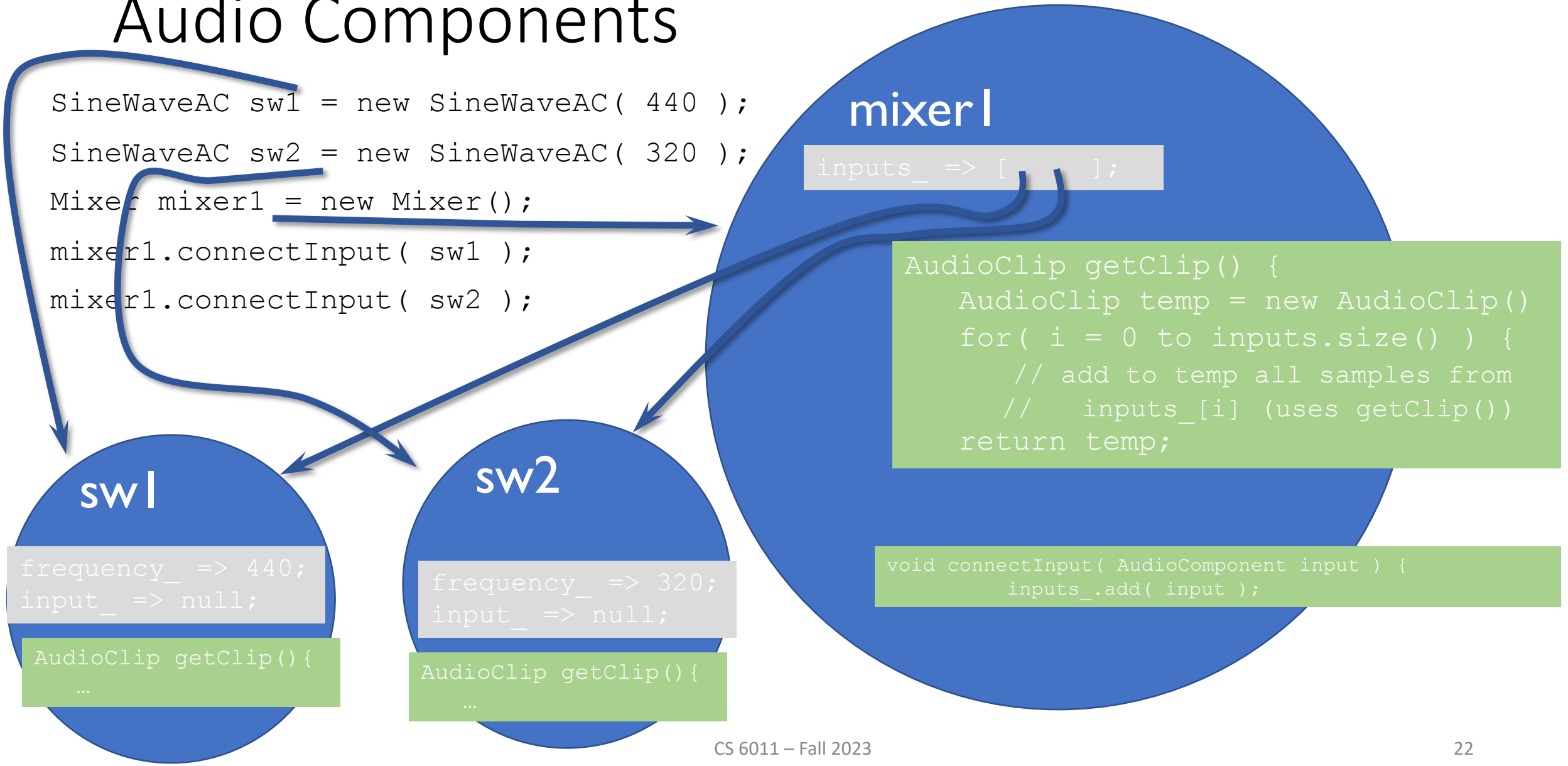
Audio Components

```
SineWaveAC sw1 = new SineWaveAC( 440 );
clipForSpeaker = sw1.getClip();
```

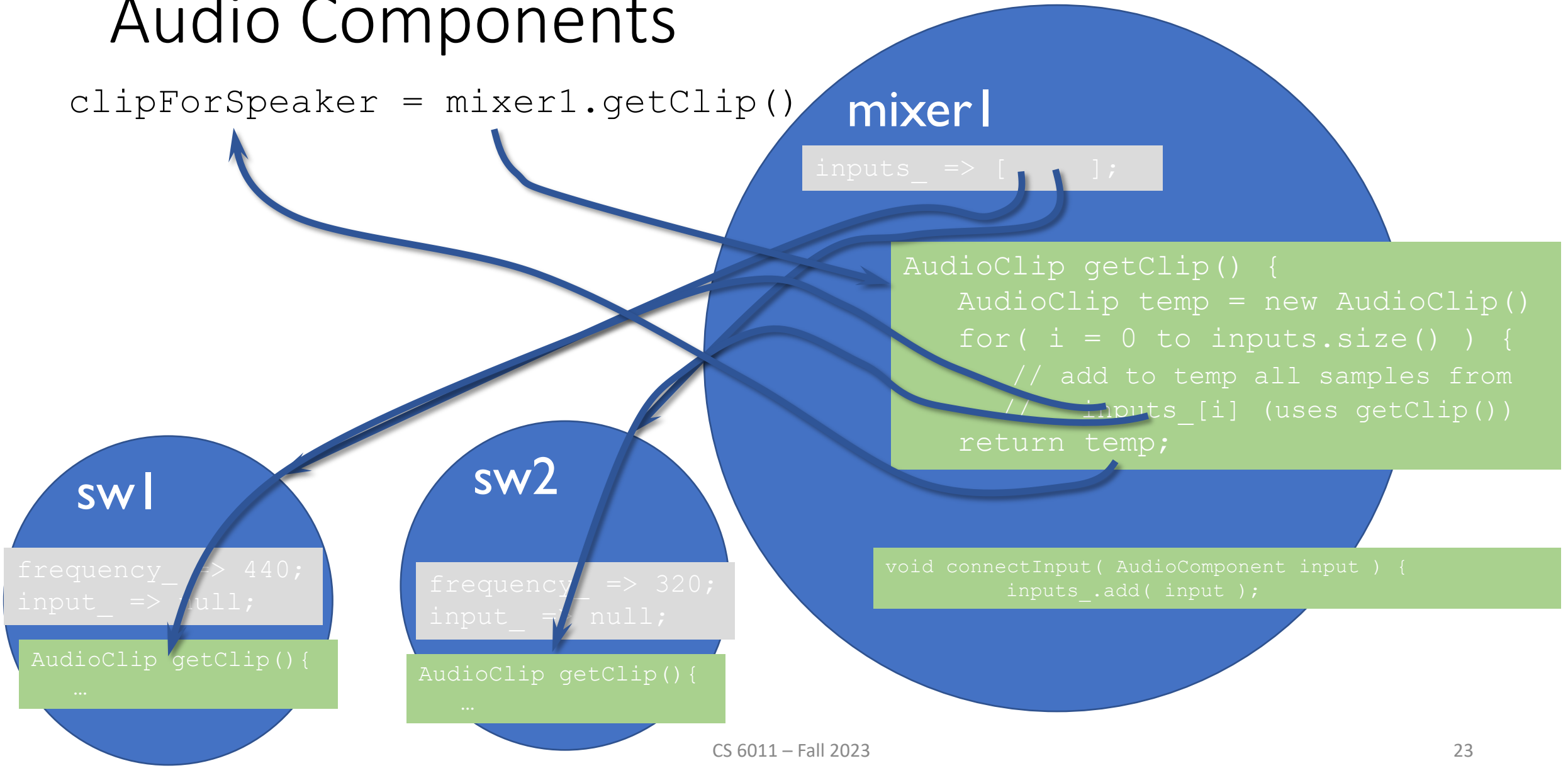


Audio Components

```
SineWaveAC sw1 = new SineWaveAC( 440 );
SineWaveAC sw2 = new SineWaveAC( 320 );
Mixer mixer1 = new Mixer();
mixer1.connectInput( sw1 );
mixer1.connectInput( sw2 );
```



Audio Components



JavaFX

- How to we create (or get) a Stage?
  - The `start` method (which is called by Java for us), gives it to us:

```
@Override
public void start( Stage stage ) throws IOException {
    AnchorPane root = new AnchorPane();
    Scene scene = new Scene( root, 320, 240 );
    stage.setTitle( "Synthesizer" );
    ...
```

JavaFX

- We need a top-level layout (parent) to be the base for holding everything. Let's use a BorderPane which will have a top, bottom, right, left, and center position for children layouts.

```
BorderPane bp = new BorderPane();
```

- To hold the top-level layout (the BorderPane), we need a scene. Note, we will add the BorderPande to the Scene, and then add the scene to the Stage, but at that point we don't care about the Stage or the Scene (for our purposes).

```
Scene scene = new Scene( bp, 1000, 500 );
// We also provide the size of the window
```

JavaFX

- Inside each piece of the border pane we will place another layout. For example, on the top, we will use a horizontal box (HBox) to hold a bunch of “menu” buttons.

```
HBox menu = new HBox();
menu.setStyle( "-fx-background-color: lightgreen"); // Using “css” to set style attributes.


  // We need some Buttons inside our HBox layout (parent), so:
Button playBtn = new Button( "Play Sound" );
Button sineWaveBtn = new Button( "Create Sine Wave AC ");

menu.getChildren().add( playBtn ); // Add the children
menu.getChildren().add( sineWaveBtn );
menu.setAlignment( Pos.CENTER ); // Center them and give some padding/spacing
menu.setSpacing( 50 );
menu.setPadding( new Insets(10,10,10,10) );
```

JavaFX

- Now, add the menu HBox to the BorderPane (ie, set the BP as the parent of the hbox. Note, unlike the hbox where we used "getChildren.add()", the BP has specific methods to add children to the various positions:

```
bp.setTop( menu );
bp.setBottom( bottom );   // Haven't created bottom yet, see next slides
bp.setCenter( ap );
```

Slider

```
Slider slider = new Slider(min, max, default);
bottom.getChildren().add( slider );
slider.setOnMouseDragOver( e-> handleSlider( e, msg,
slider  ) );
```

- But we need the handler method:
  - Note, this method is just and example and it is changing the value of a Text that was passed into the handler (above) when it was assigned to the slider.

```
private void handleSlider( Text msg, Slider slider ) {
    msg.setText( Double.toString( slider.getValue() ) );
}
```

Wednesday Assignments

- Code Reviews:
  - Synthesizer, Part 1 (4a)
- Assignment – Synthesizer GUI, Stages 1 and 2 (4b & 4c)