```java
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package jsyn;

import com.jsyn.JSyn;
import com.jsyn.Synthesizer;
import com.jsyn.unitgen.*;
import main.Template;

/**
 *
 * @author Reznov
 */
public class JSynTemplate extends Template<SawtoothOscillator, FilterLowPass, EnvelopeAttackDecay, InterpolatingDelay> {

    private Synthesizer synth;
    private LineOut lineOut;

    public JSynTemplate() {
        super("JSyn");
    }

    @Override
    public void setup(int voices, int voicesToEQAndComp, int effects, int voicesToEffects) {
        initLibrary();
        int i, j;
        for (i = 0; i < voices; i++) {
            this.voices.add(new SawtoothOscillator());
            synth.add(this.voices.get(i));
        }
        for (i = 0; i < voicesToEQAndComp; i++) {
            this.equalizers.add(new FilterLowPass());
            this.compressors.add(new EnvelopeAttackDecay());

            synth.add(this.equalizers.get(i));
            synth.add(this.compressors.get(i));

            this.voices.get(i).output.connect(this.equalizers.get(i).input);
            this.equalizers.get(i).output.connect(this.compressors.get(i).input);
        }
        for (i = 0; i < voicesToEffects; i++) {
            for (j = 0; j < effects; j++) {
                this.effects.add(new InterpolatingDelay());
                this.effects.get(i * effects + j).allocate(2);
                synth.add(this.effects.get(i * effects + j));

                if (j == 0) {
                    if (usesCompressors()) {
                        this.compressors.get(i).output.connect(this.effects.get(i * effects + j).input);
                    } else {
                        this.voices.get(i).output.connect(this.effects.get(i * effects + j).input);
                    }
                } else {
                    this.effects.get(i * effects + j - 1).output.connect(this.effects.get(i * effects + j).input);
                }
            }
            this.effects.get(i * effects + j - 1).output.connect(0, lineOut.input, 0);
            this.effects.get(i * effects + j - 1).output.connect(0, lineOut.input, 1);
        }
        for (i = i; i < voicesToEQAndComp; i++) {
            this.compressors.get(i).output.connect(0, lineOut.input, 0);
            this.compressors.get(i).output.connect(0, lineOut.input, 1);
        }
        for (i = i; i < voices; i++) {
            this.voices.get(i).output.connect(0, lineOut.input, 0);
            this.voices.get(i).output.connect(0, lineOut.input, 1);
        }
    }

    @Override
    public void run() {
        synth.start();
        lineOut.start();
    }

    @Override
    public void stop() {
        lineOut.stop();
        synth.stop();
    }
```

```
    @Override
    public void tearDown() {
        lineOut.input.disconnectAll();
        this.voices.forEach(voice -> {
            voice.output.disconnectAll();
            synth.remove(voice);
        });
        this.equalizers.forEach(equalizer -> {
            equalizer.input.disconnectAll();
            equalizer.output.disconnectAll();
            synth.remove(equalizer);
        });
        this.compressors.forEach(compressor -> {
            compressor.input.disconnectAll();
            compressor.output.disconnectAll();
            synth.remove(compressor);
        });
        this.effects.forEach(effect -> {
            effect.input.disconnectAll();
            effect.output.disconnectAll();
            synth.remove(effect);
        });
        synth.remove(lineOut);

        reset();

        System.gc();
    }

    @Override
    protected void initLibrary() {
        this.synth = JSyn.createSynthesizer();
        this.synth.add(lineOut = new LineOut());
    }

}
```