```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 beads;

import main.Template;
import net.beadsproject.beads.core.AudioContext;
import net.beadsproject.beads.core.UGen;
import net.beadsproject.beads.core.io.JavaSoundAudioIO;
import net.beadsproject.beads.data.Buffer;
import net.beadsproject.beads.ugens.WavePlayer;
import net.beadsproject.beads.ugens.Compressor;
import net.beadsproject.beads.ugens.LPRezFilter;
import net.beadsproject.beads.ugens.Reverb;

/**
 *
 * @author Reznov
 */
public class BeadsTemplate extends Template<WavePlayer, LPRezFilter, Compressor, Reverb> {

    private AudioContext audioCtx;

    public BeadsTemplate() {
        super("Beads");
    }

    @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 WavePlayer(audioCtx, 440, Buffer.SAW));
        }
        for (i = 0; i < voicesToEQAndComp; i++) {
            this.equalizers.add(new LPRezFilter(audioCtx));
            this.compressors.add(new Compressor(audioCtx));

            this.equalizers.get(i).addInput(this.voices.get(i));
            this.compressors.get(i).addInput(this.equalizers.get(i));
        }
        for (i = 0; i < voicesToEffects; i++) {
            for (j = 0; j < effects; j++) {
                this.effects.add(new Reverb(audioCtx));
                UGen previousModule;

                if (j == 0) {
                    if (usesCompressors()) {
                        previousModule = this.compressors.get(i);
                    } else {
                        previousModule = this.voices.get(i);
                    }
                    this.effects.get(i).addInput(previousModule);
                } else {
                    this.effects.get(i * effects + j).addInput(this.effects.get(i * effects + j - 1));
                }
            }
            audioCtx.out.addInput(this.effects.get(i * effects + j - 1));
        }
        for (i = i; i < voicesToEQAndComp; i++) {
            audioCtx.out.addInput(this.compressors.get(i));
        }
        for (i = i; i < voices; i++) {
            audioCtx.out.addInput(this.voices.get(i));
        }
    }

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

    @Override
```

```
    public void stop() {
        audioCtx.stop();
    }

    @Override
    public void tearDown() {
        voices.forEach(voice -> audioCtx.out.removeAllConnections(voice));
        equalizers.forEach(equalizer -> audioCtx.out.removeAllConnections(equalizer));
        compressors.forEach(compressor -> audioCtx.out.removeAllConnections(compressor));
        effects.forEach(effect -> audioCtx.out.removeAllConnections(effect));

        reset();

        System.gc();
    }

    @Override
    protected void initLibrary() {
        JavaSoundAudioIO aio = new JavaSoundAudioIO();
        aio.selectMixer(2);
        audioCtx = new AudioContext(aio);
    }

}
```