```java
package jass;

import jass.engine.SinkIsFullException;
import jass.engine.Source;
import jass.engine.ThreadMixer;
import jass.generators.Delay;
import jass.generators.Mixer;
import jass.generators.OnePoleLowPass;
import jass.generators.Sine;
import jass.render.SourcePlayer;
import main.Template;


public class JassTemplate extends Template<Sine, OnePoleLowPass, Mixer, Delay> {

        private SourcePlayer sourcePlayer;
        private PlayThread playThread;

        private static final int BUFFER_SIZE = 64;
        private static final int SAMPLE_RATE = 44100;

        private class PlayThread extends Thread {

                private boolean started, running;

                public void run() {
                        if (!running) {
                                running = true;
                                started = false;
                                while (running) {
                                        if (!started) {
                                                sourcePlayer.run();
                                                started = true;
                                        }
                                }
                        }
                }

                public void halt() {
                        running = false;
                }
        }

        public JassTemplate() {
                super("JASS");
        }

        @Override
        public void setup(int voices, int voicesToEQAndComp, int effects, int voicesToEffects) {
                initLibrary();
                try {
                        int i, j;
                        for (i = 0; i < voices; i++) {
                                this.voices.add(new Sine(BUFFER_SIZE, SAMPLE_RATE));
                        }
                        for (i = 0; i < voicesToEQAndComp; i++) {
                                this.equalizers.add(new OnePoleLowPass(BUFFER_SIZE));
                                this.compressors.add(new Mixer(BUFFER_SIZE, 1));

                                this.equalizers.get(i).addSource(this.voices.get(i));
                                this.compressors.get(i).addSource(this.equalizers.get(i));

                                this.compressors.get(i).setGain(0, 1.0f);
                        }
                        for (i = 0; i < voicesToEffects; i++) {
                                for (j = 0; j < effects; j++) {
                                        this.effects.add(new Delay(BUFFER_SIZE));
                                        this.effects.get(i * effects + j).setRawDelay(0.5f);

                                        if (j == 0) {
                                                Source previousModule;
                                                if (this.usesCompressors()) {
                                                        previousModule = this.compressors.get(i);
                                                } else {
                                                        previousModule = this.voices.get(i);
                                                }
                                                this.effects.get(i * effects + j).addSource(previousModule);
                                        } else {
                                                this.effects.get(i * effects + j).addSource(this.effects.get(i * effects + j - 1));
                                        }
                                }
                                sourcePlayer.addSource(this.effects.get(i * effects + j - 1));
                        }
                        for (i = i; i < voicesToEQAndComp; i++) {
                                sourcePlayer.addSource(this.compressors.get(i));
                        }
                        for (i = i; i < voices; i++) {
                                sourcePlayer.addSource(this.voices.get(i));
                        }
                } catch (SinkIsFullException ex) {
                        ex.printStackTrace();
                }
        }

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

        @Override
        public void stop() {
                sourcePlayer.stopPlaying();
                playThread.halt();
        }

        @Override
        public void tearDown() {

                reset();

                System.gc();
        }

        @Override
        protected void initLibrary() {
                sourcePlayer = new SourcePlayer(BUFFER_SIZE, BUFFER_SIZE, SAMPLE_RATE, "default [default]");
                sourcePlayer.setOutputChannelNum(2);
                playThread = new PlayThread();
        }
}
```