BAB 4
EFEK SUARA

4.1 Pendahuluan

Pada sebuah permainan efek suara sangat dibutuhkan untuk membuat game yang dibangun lebih semakin hidup dan terasa nyata, hal ini juga yang membuat para pemain lebih tertarik, bayangkan apabila di dalam suatu permainan tidak ada suara sama sekali. Efek suara sangat diperlukan oleh game misalnya ketika sebuah objek mengalami tabrakan, ledakan bom, suara tembakan, dll.

4.2 Tujuan

Setelah mempelajari modul ini peserta diharapkan dapat:

- Memanggil dan Memainkan File Suara.
- Memanggil dan Memainkan File MIDI Sequence.
- Mengetahui perbedaan antara file berekstensi midi dan (Aiff, Au, Wav)
- Mengetahui cara menggunakan class File dan URL untuk meload file suara

4.3 Mengakses File Suara

API Sound pada Java mendukung 3 macam format suara yang digunakan di Web dan Aplikasi Desktop yaitu:

1. AIFF
2. AU
3. WAV

Untuk memanggil dan mendengarkan suara, Java memerlukan paket **javax.sound.sampled.*** untuk disertakan di baris awal program yang dapat dituliskan seperti dibawah ini:

```
import javax.sound.sampled.*;
```

Apabila menggunakan eclipse, maka akan terlihat menu pop-up yang menampilkan nama-nama package dalam javax. Jika mengetikan javax.sound diikuti dengan titik maka akan terlihat package di dalamnya yaitu sampled dan midi. Dengan menambahkan **.*** pada akhir statement maka compiler sudah memberitahukan untuk meng-import semua class dalam javax.sound.sampled.

AudioSystem dan AudioInputStream adalah class yang berada di dalam paket javax.sound.sampled dan digunakan untuk memanggil dan memainkan sebuah suara.

Kunci utama untuk pemanggilan suara adalah class bernama AudioInputStream. Class ini digunakan untuk memuat sebuah file suara dari file lokal dimana program diletakkan atau dari URL lain di internet. Berikut ini adalah pembuata sebuah obyek/instance baru dari class tersebut.

```
AudioInputStream suara;
```

Statement ini biasanya diposisikan sebagai variable global dalam class, didefinisikan pada baris-baris awal dalam class sebelum semua method ditulis. Selain itu juga dapat didefinisikan sebagai variable private, public, atau protected (defaultnya adalah public). Dalam peraturan pemrograman berbasis objek, public memposisikan variable agar dapat diakses oleh class lain diluar variable tersebut didefinisikan, private berarti variable tersebut tidak dapat diakses dari class lain, sedangkan protected hampir mirip dengan private hanya

saja class-class yang menjadi sub-classnya(melalui inheritance) bisa menggunakannya.

Kode untuk memuat suara dari URL biasanya diletakkan dalam konstruktor. Method yang digunakan untuk memuat sebuah file suara adalah AudioSystem.getAudioInputStream. Method ini menerima sebuah File , InputStream atau URL, ada dua cara untuk membuat audio stream (AudioFormat dan Encoding).

```
suara = AudioSystem.getAudioInputStream(new File("BOOM.wav"));
```

Nilai balik dari method ini adalah AudioInputStream. Karena getAudioInputStream tidak memiliki versi yang dapat di overload dan hanya menerima input berupa string dari nama file, maka harus melewatkan sebuah objek File kedalamnya. Cukup mudah menggunakan objek dari File, hanya dengan memberikan nama file pada kontruktor method File. Apabila ingin mengambil sebuah file dari sebuah URL, kode yang digunakan adalah sebagai berikut :

```
URL lokasi = new URL("http://www.vedcmalang.ac.id/suara/BOOM.wav");
suara = AudioSystem.getAudioInputStream(lokasi);
```

Apabila ingin mendistribusikan program java yang telah dibuat dan ingin mendapatkan hasil yang maksimal, sebaiknya mengganti method new File() dan new URL() untuk memuat semua sumber (seperti sebuah file gambar atau suara) dengan memanfaatkan this.getClass().getResource(), dimana Method getResource() berada di instance class yang saat ini digunakan this.getClass().. Berikut ini sebuah method yang di tulis menggunakan fungsi tersebut :

```
    public URL getURL(String namaFile){
```

```
            URL url=null;
            try{
                  url=this.getClass().getResource(namaFile);
            }catch(Exception e){}
            return url;
      }
```

AudioInputStream tidak dapat langsung digunakan untuk memainkan suara, AudioInputStream hanya dapat mengakses file audio tanpa kemampuan untuk memainkanya. Untuk itu, perlu menggunakan class lain yang bernama Clip (Import javax.sound.sampled.clip).

```
Clip klip =AudioSystem.getClip();
```

4.4 Memuat dan Mememainkan Klip Suara

Sebelumnya sudah mempunyai sebuah AudioInputStream dan sebuah Clip, jadi hanya perlu untuk membuka file suara dan memainkannya. Langkah ini dua-duanya dilakukan oleh class Clip. Pertama , bukalah file suara :

```
klip.open(suara);
```

Berikutnya, ada dua cara atau method untuk memainkan klip, dengan menggunakan class Clip yaitu : Method start() dapat digunakan untuk memainkan klip suara dengan kode seperti berikut:

```
klip.start();
```

Untuk yang kedua adalah method loop(), dimana method ini menyediakan sebuah pilihan yang memperbolehkan untuk menentukan seberapa banyak klip akan diulang. baik dengan jumlah tertentu maupun tak hingga. Dibawah ini kode yang memungkinkan untuk memainkan klip suara dengan method loop() :

```
klip.loop(0);
```

Ingat, parameter menunjukkan seberapa banyak ia akan diulang. Berikut ini adalah bagaimana agar klip suara dapat dimainkan terus menerus :

```
klip.loop(Clip.LOOP_CONTINOUSLY);
```

Pada kenyataannya programer sering lupa untuk menghentikan suara yang telah dimainkan padahal tidak semua suara dimainkan terus menerus seperti suara tabrakan dll, untuk itu jika ingin menghentikan klip suara dapat menggunakan method stop() seperti pada potongan kode dibawah ini:

```
klip.stop();
```

4.5 Memuat dan Mememainkan MIDI

MIDI adalah singkatan dari Musical Instrument Digital Interface. MIDI adalah sebuah kumpulan format musik, bukan sebuah format sample, artinya musik MIDI tidak direkam menggunakan konverter analog-to-digital (yang berada pada sound card di dalam komputer).

Perlu diketahui bahwa file berekstensi (aiff,au dan wav) berukuran cukup besar, sedangkan aplikasi tidak ingin pengguna menunggu lima menit atau lebih (khususnya dial-up) hanya untuk menunggu sampai klip selesai didownload(game jaringan). Hal ini terjadi ketika anda memanggil method open(), jadi ketika anda mencoba untuk membuka sebuah file suara berukuran besar, maka method ini akan memaksa pengguna untuk menunggu selama waktu download klip suara. Hal ini yang dapat menjawab permasalahan tersebut adalah MIDI sequence.

Memuat file MIDI memerlukan sedikit lebih banyak cara dari pada memuat file berekstensi (aiff,au dan wav). Hal ini karena file MIDI dimainkan

melalui perangkai (sequencer) bukan melalui sound mixer. Class Sequence digunakan untuk memuat file MIDI :

```
Sequence suara = MidiSystem.getSequence(new File("contoh.mid"));
```

Meskipun kode tersebut disiapkan agar file MIDI dapat dimainkan melalui sequencer, kita belum sama sekali membuat instance dari sequencer:

```
Sequencer ambilSuara= MidiSystem.getSequencer();
```

Berikutnya, Memberitahukan kepada class Sequencer bahwa kita memiliki file midi (melalui class Sequence):

```
ambilSuara.setSequence(suara);
```

Sekarang yang perlu dilakukan adalah membuka file dan memainkannya:

```
ambilSuara.open();
ambilSuara.start();
```

Untuk memainkan MIDI secara berulang-ulang sebanyak yang diinginkan gunakan method setLoopCount() dimana parameternya sebagai penentu:

```
ambilSuara.setLoopCount(1);
```

Apabila ingin dimainkan secara terus menerus tanpa henti gunakan kode berikut:

```
ambilSuara.setLoopCount(Sequencer.LOOP_CONTINUOUSLY);
```

Terakhir adalah menghentikan MIDI, prosesnya sama seperti menghentikan file (aiff,au,wav) yaitu dengan cara memanggil method stop():

```
ambilSuara.stop();
```

4.6 Penanganan Error

Java menyediakan penanganan error pada class Sound, compiler secara otomatis akan menolak atau memberikan pesan kesalahan apabila tidak menggunakan blok try... catch. Sebagai contoh, ketika memuat sebuah file, pertama kali akan diperiksa dengan IOException sebelum menggunakan handler umum(Exception). Class AudioSystem, AudioInputStream dan Clip membutuhkan handler dibawah ini :

- IOException
- LineUnavailableException
- UnsuportedAudioFileException

Cara menggunakan sebuah error handler untuk program yang akan memainkan sebuah suara. Kode berikut ini diletakkan pada konstruktor :

```
KONSTRUKTOR(){
            try {
              //kode program
      } catch (MalformedURLException e){
      } catch (IOException e){
      } catch (LineUnavailableException e){
      } catch (UnsuportedAudioFileException e){}
}
```

Disamping itu class Sequence dan Sequencer juga membutuhkan penanganan error yang tidak jauh berbeda dengan class Class AudioSystem, AudioInputStream dan Clip. Cara penanganannya dapat dituliskan sebagai berikut:

```
KONSTRUKTOR(){
            try {
                  // kode program
            } catch (InvalidMidiDataException e) {
            } catch (IOException e) {
            } catch (MidiUnavailableException e) { }
}
```

Penanganan error sangat diperlukan untuk mencegah program yang dijalankan crash atau rusak.

4.7 Class OperasiSuara

Pembuatan class OperasiSuara dimaksudkan agar dapat digunakan pada pembuatan game nanti dengan mempermudah peserta ketika mengakses dan mendengarkan suara.

```
import java.io.*;
import java.net.*;
import javax.sound.sampled.*;

public class OperasiSuara{

	private AudioInputStream suara;
	private Clip klip;
	private String namaFile = "";
	private boolean looping = false;
	private int repeat = 0;

	public OperasiSuara(){
		try{
			klip = AudioSystem.getClip();
		}catch(LineUnavailableException e)
{System.out.println("Terjadi kesalahan pada klip suara :
"+e.toString());}
	}

	public OperasiSuara(String filesuara) {
		this();//memanggil default konstruktor pertama (tanpa
parameter);
		load(filesuara);
	}

	public boolean load(String filesuara){
		try{
			setFilename(filesuara);
			suara =
AudioSystem.getAudioInputStream(getURL(namaFile));
			klip.open(suara);
			return true;
		}catch(IOException e){
			return false;
		}catch(UnsupportedAudioFileException e){
			return false;
		}catch(LineUnavailableException e){
			return false;
		}
	}
	public void setFilename(String nama_fail) {namaFile =
nama_fail;}
	public void setLooping(boolean ulang) {looping = ulang;}
```

```
	public void setRepeat(int ulangi) {repeat= ulangi;}
	public Clip getClip() { return klip;}
	public int getRepeat() {return repeat;}
	public boolean getLooping() {return looping;}
	public String getNamaFile() {return namaFile;}
	private URL getURL(String filename){
		URL url = null;
		try {
			url = this.getClass().getResource(filename);
			System.out.println(url);
		}catch(Exception e){ System.out.println("error url :
"+e.toString());}
		return url;
	}
	public boolean isLoaded(){
		return (boolean)(suara != null);
	}
	public void play(){
		if (!isLoaded()) return;

		klip.setFramePosition(0);

		if(looping)
			klip.loop(Clip.LOOP_CONTINUOUSLY);
		else
			klip.loop(repeat);
	}
	public void stop(){
		klip.stop();
	}
}
```

4.8 Class OperasiMIDI

Pembuatan class OperasiMIDI sangat dibutuhkan ketika peserta sudah masuk pada pembuatan geme dimana perserta hanya mengakses method yang disediakan di class ini dan tanpa harus menuliskannya secara berulang-ulang.

```
import java.io.*;
import java.net.*;
import javax.sound.midi.*;

public class OperasiMidi{

	private Sequence suara;
	private Sequencer ambilSuara;
	private String namaFile;
	private boolean looping = false;
	private int repeat = 0;


	public OperasiMidi(){
		try{
			ambilSuara = MidiSystem.getSequencer();
```

```
            }catch(MidiUnavailableException e){ }
      }

      public OperasiMidi(String fileMidi){
            this();//memanggil default konstruktor pertama (tanpa
parameter);
            load(fileMidi);
      }
      public boolean load(String fileMidi){
            try{
                  namaFile = fileMidi;
                  suara = MidiSystem.getSequence(getURL(namaFile));
                  ambilSuara.setSequence(suara);
                  ambilSuara.open();
                  return true;
            }catch(InvalidMidiDataException e){
                  return false;
            }catch(MidiUnavailableException e){
                  return false;
            }catch(IOException e){
                  return false;
            }
      }
      public void setLooping(boolean ulang) {looping =ulang;}
      public void setRepeat(int ulangi) {repeat= ulangi;}

      public Sequence getSuara() { return suara;}
      public String getFilename() {return namaFile;}
      public boolean getLooping() {return looping;}
      public int getRepeat() {return repeat;}
      private URL getURL(String filename){
            URL url = null;
            try {
                  url = this.getClass().getResource(filename);
            }catch(Exception e){ }
            return url;
      }

      public boolean isLoaded(){
            return (boolean)(ambilSuara.isOpen());
      }

      public void play(){
            if(!ambilSuara.isOpen()) return;
            if(looping){

      ambilSuara.setLoopCount(Sequencer.LOOP_CONTINUOUSLY);
                  ambilSuara.start();
            }else{
                  ambilSuara.setLoopCount(repeat);
                  ambilSuara.start();
            }
      }

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