Elektrotehnički fakultet
Univerzitet u Banjoj Luci

KATEDRA ZA RAČUNARSKU TEHNIKU
(smijer: Informacioni sistemi)

SEMINARSKI RAD IZ PREDMETA MULTIMEDIJALNI SISTEMI

- Java Media Framework -

Implementacija mpeg4 kodeka u JAVI
Implementacija Java Media Player-a

Članovi grupe:

Slavko Bojagić
Branimir Cvijić
Predrag Bajić

JAVA MEDIA FRAMEWORK (JMF)

Java Media Framework API (*JMF*) omogućuje dodavanje audia i videa aplikacijama i apletima koji su zasnovani na Java tehnologiji. Ovaj paket ili biblioteka nije sastavni dio standardne distribucije Jave. Ovaj opcioni paket, koji može snimati, reprodukovati, raditi sa strimovima i obrađivati različite multimedijalne formate, je proširenje za Java 6 Platform Standard Edition, i daje programerima veoma moćan alat za pisanje skalabilnih, kros-platformskih aplikacija.

Koliko je taj dodatni paket moćan i koliko formata podržava na različitim platformama, pokazuje i tabela ispod:

D – format može biti dekodiran i prezentovan
E – medija stream može biti enkodovan u odgovarajućem formatu
Read – format može biti ulazni
Output – format može biti generisan kao izlazni

Media Type	**JMF 2.1.1** **Cross Platform Version**	**JMF 2.1.1** **Solaris/Linux Performance Pack**	**JMF 2.1.1** **Windows Performance Pack**
AIFF (.aiff)	**read/write**	**read/write**	**read/write**
8-bit mono/stereo linear	D,E	D,E	D,E
16-bit mono/stereo linear	D,E	D,E	D,E
G.711 (U-law)	D,E	D,E	D,E
A-law	D	D	D
IMA4 ADPCM	D,E	D,E	D,E
AVI (.avi)	**read/write**	**read/write**	**read/write**
Audio: 8-bit mono/stereo linear	D,E	D,E	D,E
Audio: 16-bit mono/stereo linear	D,E	D,E	D,E
Audio: DVI ADPCM compressed	D,E	D,E	D,E
Audio: G.711 (U-law)	D,E	D,E	D,E
Audio: A-law	D	D	D
Audio: GSM mono	D,E	D,E	D,E
Audio: ACM**	-	-	D,E
Video: Cinepak	D	D,E	D
Video: MJPEG (422)	D	D,E	D,E
Video: RGB	D,E	D,E	D,E
Video: YUV	D,E	D,E	D,E
Video: VCM**	-	-	D,E
GSM (.gsm)	**read/write**	**read/write**	**read/write**
GSM mono audio	D,E	D,E	D,E

HotMedia (.mvr)	**read only**	**read only**	**read only**
IBM HotMedia	D	D	D
MIDI (.mid)	**read only**	**read only**	**read only**
Type 1 & 2 MIDI	-	D	D
MPEG-1 Video (.mpg)	**-**	**read only**	**read only**
Multiplexed System stream	-	D	D
Video-only stream	-	D	D
MPEG Layer II Audio (.mp2)	**read only**	**read/write**	**read/write**
MPEG layer 1, 2 audio	D	D,E	D,E
QuickTime (.mov)	**read/write**	**read/write**	**read/write**
Audio: 8 bits mono/stereo linear	D,E	D,E	D,E
Audio: 16 bits mono/stereo linear	D,E	D,E	D,E
Audio: G.711 (U-law)	D,E	D,E	D,E
Audio: A-law	D	D	D
Audio: GSM mono	D,E	D,E	D,E
Audio: IMA4 ADPCM	D,E	D,E	D,E
Video: Cinepak	D	D,E	D
Video: H.261	-	D	D
Video: H.263	D	D,E	D,E
Video: JPEG (420, 422, 444)	D	D,E	D,E
Video: RGB	D,E	D,E	D,E
Sun Audio (.au)	**read/write**	**read/write**	**read/write**
8 bits mono/stereo linear	D,E	D,E	D,E
16 bits mono/stereo linear	D,E	D,E	D,E
G.711 (U-law)	D,E	D,E	D,E
A-law	D	D	D
Wave (.wav)	**read/write**	**read/write**	**read/write**
8-bit mono/stereo linear	D,E	D,E	D,E
16-bit mono/stereo linear	D,E	D,E	D,E
G.711 (U-law)	D,E	D,E	D,E
A-law	D	D	D
GSM mono	D,E	D,E	D,E
DVI ADPCM	D,E	D,E	D,E
MS ADPCM	D	D	D
ACM**	-	-	D,E

Što se tiče hardverskih zahtijeva, da bi se ova biblioteka mogla koristiti, potrebno je imati minimalno procesor čija je radni takt 200MHz za Pentium procesore. Kada su u pitanju softverski zahtijevi, neophodno je na računaru imati instaliranu Javu, JDK 1.1.5 ili neku od novijih verzija ali sa uključenom Swing bibliotekom, Java Media Framework klase. S obzirom da je i Java, kao i njeno proširenje u vidu JMF-a portabilno kroz platforme, nema potrebe navoditi softverske zahtijeve kada je u pitanju operativni sistem.

Da bi se JMF uspješno mogao koristiti, neophodno je izvršiti njegovu pravilnu „instalaciju“, a mi ćemo se u ovom seminarskom radu ograničiti samo na Windows operativni sistem i potrebna podešavanje vezana samo za njega.

Za instalaciju je neophodno pokrenuti EXE verziju JMF-a, za Windows, a nakon toga izvršiti odgovarajuće podešavanje CLASSPATH i PATH varijable:

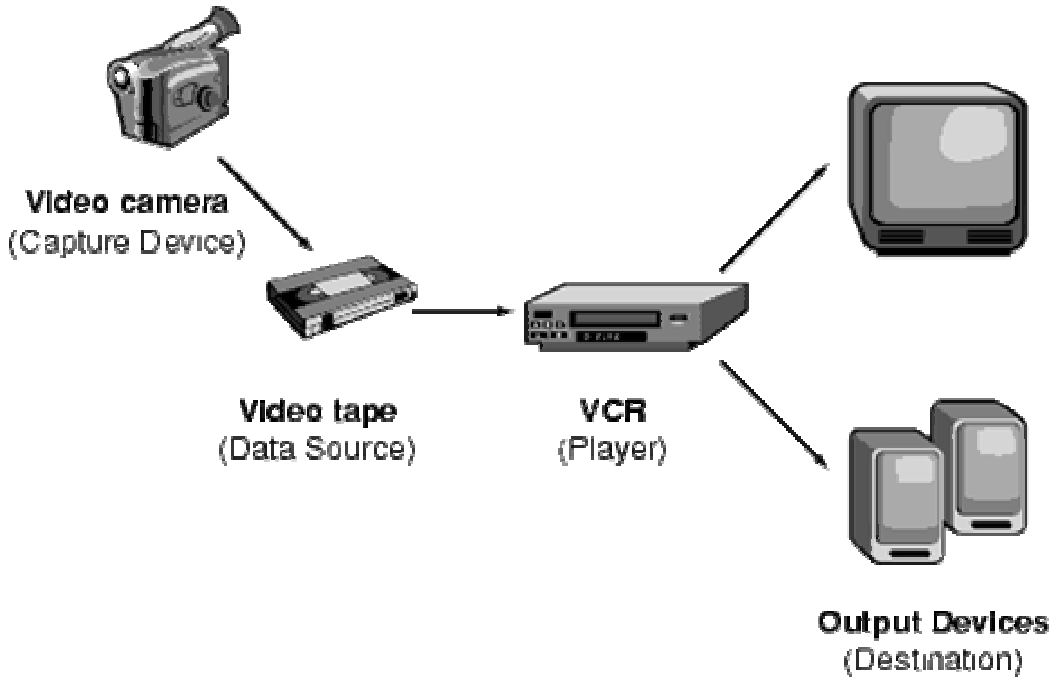
```
set CLASSPATH=%JMFDIR%\lib\jmf.jar;%JMFDIR%\lib\sound.jar;.;%CLASSPATH%

set PATH=%JMFDIR%\lib;%PATH%
```

JMF se sastoji od kolekcije medija hendlera, plaginova i capture uređaja. Oni su konfigurabilni i proširivi. Da se konfiguriše JMF, neophodno je koristiti JMFRegistry aplikaciju koja dolazi kao sastavni dio instalacije JMF-a.

JMF: High-Level Arhitektura

Na slici je su prikazani uređaji koji čine arhitekturu visokog nivoa. Tu imamo uređaj koji prikuplja audio-video podatke, uređaj koji čuva te podatke, koji reprodukuje sadržaj, i uređaje koji prikazuju reprodukciju.



Data Source: Predstavlja lokaciju na kojoj se nalaze audio/video podaci. Komunikacija sa drugim uređajima i softverom, definisana je odgovarajućim protokolom.

Capture Device: Uređaj koji prihvata podatke iz okruženja. Ujedno može biti i DataSource

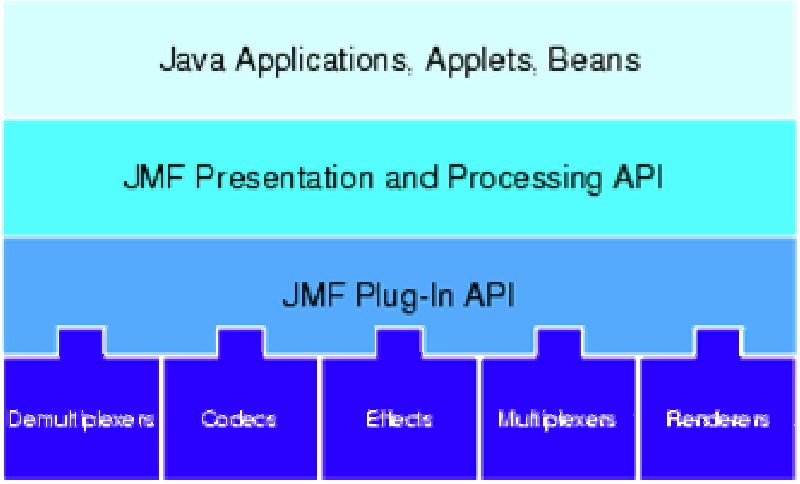
Player: Vrši demultipleksiranje, dekodiranje i renderovanje.

Destination: Izlazni uređaji (video, zvuk...)

Koristeći JMF mogu se implementirati neke funkcionalnosti ovih uredjaja i pokazati kako oni zapravo rade. Cilj našeg seminarskog rada je bio jasan, napraviti u Javi audio/video player koji pušta zapise u mpeg4 formatu (uključujući i sve ostale podržane formate od strane JMF-a kao i dodatnog kodeka FFOBS, koji je naknadno objašnjen), tako da smo se bazirali samo na reprodukciji već snimljenih zapisa, a ne snimanjem ili konverzjom istih.

JMF: API slojevi

• API (Aplication Programing Interface) - Open, extensible, baziran na interfejsima (Interfaces)



Arhitektura Java Media Frameworka sastoji se od četiri sloja. Ovim slojevima pristupamo preko interfejsa za programiranje aplikacija (API).

Najviši sloj, predstavlja reprodukovanje media preko aplikacije, apleta ili bean-a.

Dva srednja sloja, definišu na koji način se prezentuju podaci, njihovu obradu i formiranje Plug-In lanca.

Najniži sloj je u direktnom "kontaktu" sa hardverom. Odgovoran je za multipleksiranje, demultipleksiranje kanala, kodovanje/dekodovanje, primjenu efekata i rendering.

Niži slojevi koriste servise višeg sloja.

JMF: Programski Model

Osnovna klasa je klasa **Manager**. Sadrži metode koje provjeravaju sistemske resurse. Kreira **DataSource**, **Player**, **Processor** i **DataSink** objekte, koji implementiraju odgovarajuće ponašanje.

• **DataSource**
<protocol package-prefix>.media.protocol.<protocol>.Data Source

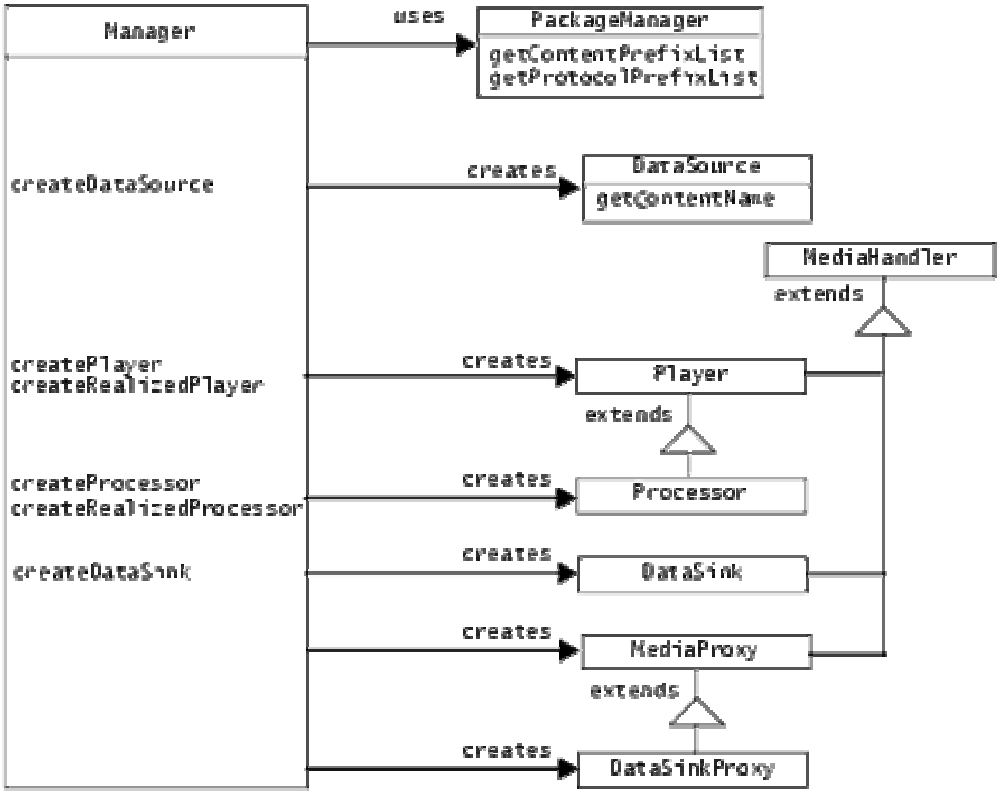
primjer: com.sun.media.protocol.avi.DataSource

• **Player**
<content package-prefix>.media.content.<content-type>.Handler

primjer: com.sun.media.content.application.x_jmx.Handler

• **Processor**
<content package-prefix>.media.processor.<content-type>.Handler

• **DataSink** (simplified)
<content package-prefix>.media.datasink.protocol.Handler



IBM MPEG-4 Video for JMF

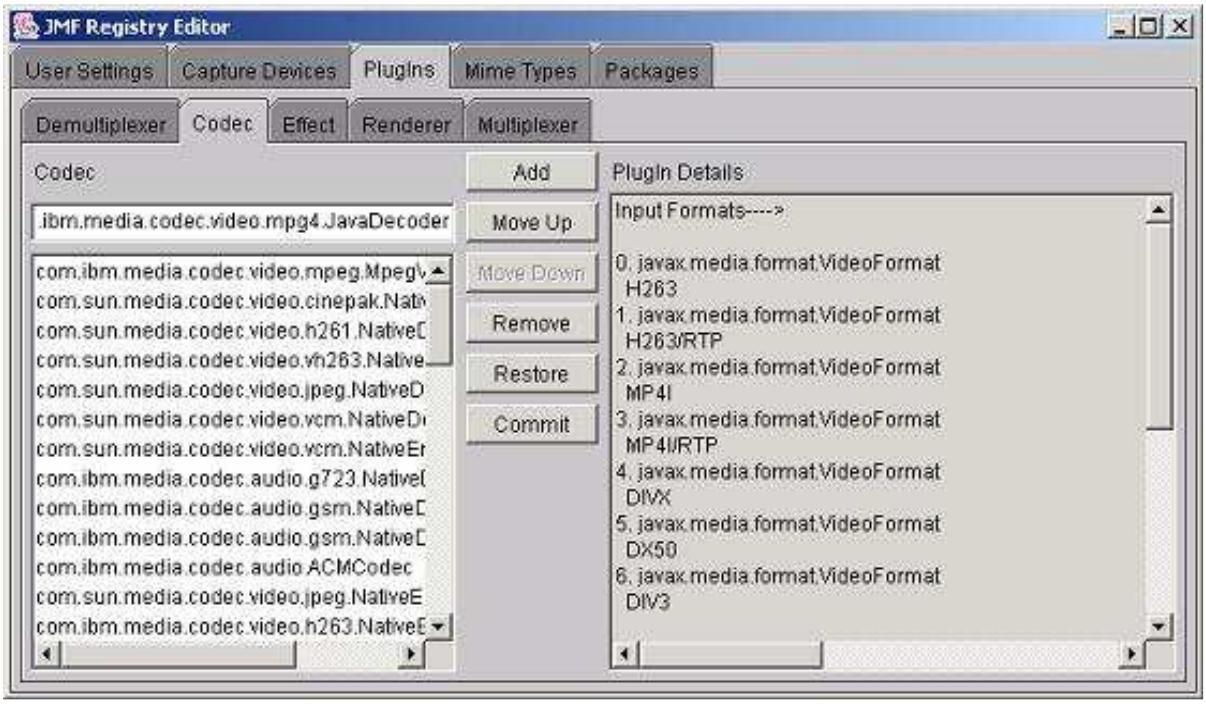
Ovaj MPEG-4 Video plagin donosi novu tehnologiju MPEG-4 video kompresije. U potpunosti napisan u programskom jeziku JAVA i omogućava dekodovanje MPEG-4 videa. Moze se koristiti na bilo kojoj JMF platformi. Dolazi u vidu plagina i razvijen je od strane IBM-a. Kodek je u potpunosti prošao potpunu optimizaciju kako bi se osigurale što bolje performanse. MPEG-4 Video za JMF implementira CODEC interfejs i da bi se moga koristiti neophodno je, naravno, imati instaliranu Javu i JMF, verzija 2.1.1, kao i ispoštovati ostale hardverske i softverske zahtjeve navedene gore u tekstu.

Da bi se uredno instalirao i mogao koristiti, nephodno je izvesti sljedeće korake:

3 **MPEG4VideoDecoderEval.zip** uključiti u postojeći CLASSPATH,
4 Registrovati ga sa **JMFRegistry** aplikacijom.

Koraci neophodni za registraciju:

5 Startovati JMFRegistry,
6 Kliknuti na PlugIns tab, pa na Codec tab.
7 Upisati *com.ibm.media.codec.video.mpg4.JavaDecoder* u Codec text field, i kliknuti na Add button, a zatim na Commit.



Izgled JMFRegistry Editora

FFOBS (**F**fmpeg **OB**ject**S**) OmniVidea

Pored već pomenutog IBM kodeka, u Java Media Player smo implementirali još jedan kodek na koji smo naišli na internetu. Riječ je o objektno orijentisanoj C++ omotačkoj bliblioteci za ffmpeg biblioteku. Ovaj kodek nam omogućava pisanje multimedijalnih aplikacija u multiplatformskim okruženjima.

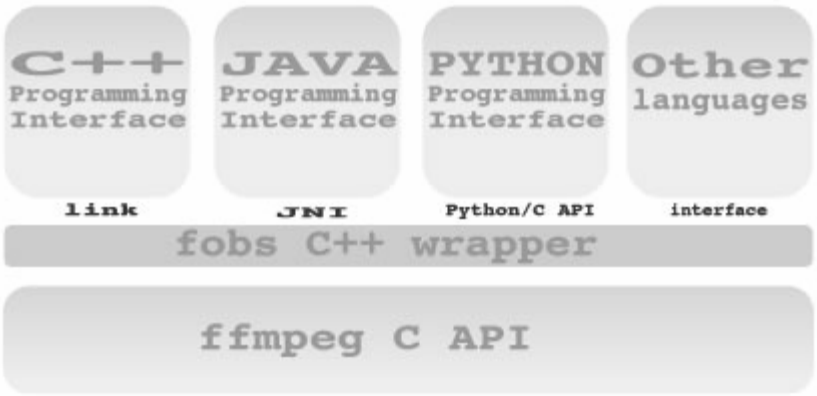
FFmpeg je kolekcija besplatnog softvera, sa mogućnostima snimanja, konvertovanja i striminga digitalnog audio i video sadržaja. Uključuje **libavcodec** (vodeću audio/video biblioteku kodeka) i **libavformat** (takođe jednu od vodećih audio/video biblioteka za mux i demux). FFmpeg je razcijen pod Linux-om, ali može biti kompajliran pod većinom operativnih sistema, uključujući i Microsoft Windows.
FFmpeg je pod GNU GPL licencom, dakle besplatan je i može se slobodno koristiti i dalje distribuirati.

Ono što još možemo reći o FFmpeg biblioteci jeste da je to veoma brz audio i video konverter. Takođe može raditi i „grebovanje“ audio/video sadržaja sa nekog live izvora. Moguće je konvertovanje iz bilo kog sample rate-a u bilo koji drugi, promjena veličine videa u „letu“, sa visokokvalitetnim filtrom.

Nakon kratkog predstavljanje ffmpeg biblioteke, vratićemo se na opis FFOBS kodeka, koji je u stvari i koristi. Dakle, FFOBS je skup objektnoorjentisanog API-a, za rad sa multimedijalnim datotekama. Programerima daje veoma jednostavan programerski interfejs.

FFOBS je bio dostupan samo u C++. Java API je skoro razvijen i trenutna verzija već uključuje JMF plagin. Uspješno je testiran na različitim platformama (Linux, MacOS X, Win32).

Arhitektura FFOBS-a je data na slici ispod:



Ono što FFOBS karakteriše je:

- jednostavnost upotrebe,
- platformska nezavisnost,
- besplatan je, kao i sam ffmpeg,
- snaga koja se ogleda u velikom broju podržanih audio i video formata, strimova (ulaz može biti kako fajl, tako i udaljeni resurs: url, rtsp, ...)

FFOBS se unutar Java integriše zahvaljujući Java JNI. Dakle koristi C++ API, ali putem Java JNI je dustupan za upotrebu i u Java programskom jeziku. Ovaj plagin se savršeno integriše u Java Media Framework, dozvoljavajući postojećim i novim aplikacijama da otvore i edituju video i audio fajlove u svim formatima podržanim od strane ffmpeg biblioteke, upotrebom standardnog JMF API-ja.

Instalacija i podešavanje FFOBS-a je sistemski zavisna, pa ćemo ovdje samo prikazati podešavanje koja se odnose na Microsoft Windows operativni sistem.

Dakle, da bi se FFOBS uspješno instalirao i mogao koristiti potrebno je uraditi sljedeće:

- instalirati JMF (verzija 2.1.1.)
- uključiti $FOBS_HOME/lib/fobs4jmf.jar na postojeći CLASSPATH
- kopirati fajl fobs4jmf.dll u windows/system32 direktorij
- registrovati plagin sa JMFRegistry aplikacijom.

Što se registracije FFOBS-a tiče, neophodno je uključiti sljedeće pakete:

<u>U Demultiplexer tab:</u> *com.omnividea.media.parser.video.Parser*

<u>U Codex tab:</u> *com.omnividea.media.codec.video.NativeDecoder*
com.omnividea.media.codec.audio.NativeDecoder
com.omnividea.media.codec.video.JavaDecoder

<u>U Package tab</u>: *com.omnividea (samo pod opcijom Protocol Prefix)*

Za svaki od ovih unosa potrebno je kliknuti na dugme Add, a zatim na Commit, kao što je gore objašnjeno za registraciju MPEG-4 plagina.

Nakon ovih koraka, ovaj plagin je uspješno instaliran na sistemu i biće korišten automatski kada bude potreban da bi se odgovarajući audio/video sadržaj reprodukovao ili obrađivao od strane Java aplikacije.

JAVA MEDIA PLAYER

Naš primjer media player-a ima jednostavan grafički korisnički interfejs (GUI) i mogućnost odabira fajla koji će se puštati, putem grafičke komponente koja je slična explorer-ovom Open File dijalogu. Odlučili smo se da implementiramo GUI kako bi naš primjer A/V player bio vizuelno dopadljiv i jednostavniji za neku buduću upotrebu.

Java Media Player je implementiran upotrebom sljedećih tehnolgija:

1. *Java SE Development Kit 6.0*
2. *Java Media Framework 2.1.1e*
3. *NetBeans 5.5*
4. *IBM MPEG-4 decoder*
5. *FOBS ffmpeg decoder*

JMF ulaz (input)

U JMF, ulaz je generalno reprezentovan objektom tipa **MediaLocator**. MediaLocator liči i ponaša se kao i URL (Universal Resouce Locator), u tom smislu da na jedinstven način identifikuje neki resurs. U principu, moguće je napraviti MediaLocator koristeći URL; to je i urađeno u prethodnom primjeru.

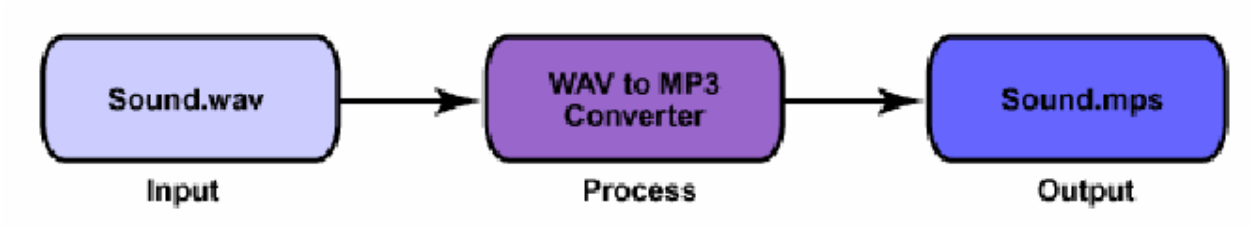
JMF procesor (obrađivač)

U slučaju da smo vršili konverziju audio ili video formata u neki drugi format, koristili bi smo instancu Processor interfejsa. Ovaj interfejs omogućiće dva dodatna stanja: *Configuring* i *Configured*. Ova dodatna stanja (i njihovi odgovarajući event-i) koriste se kada Processor prikuplja informacije iz ulaznog stream-a.
Mi nismo u cilju imali koverziju tipova, već implementaciju mpeg4 dekodera tako da interfejs Processor nismo ni implementirali.

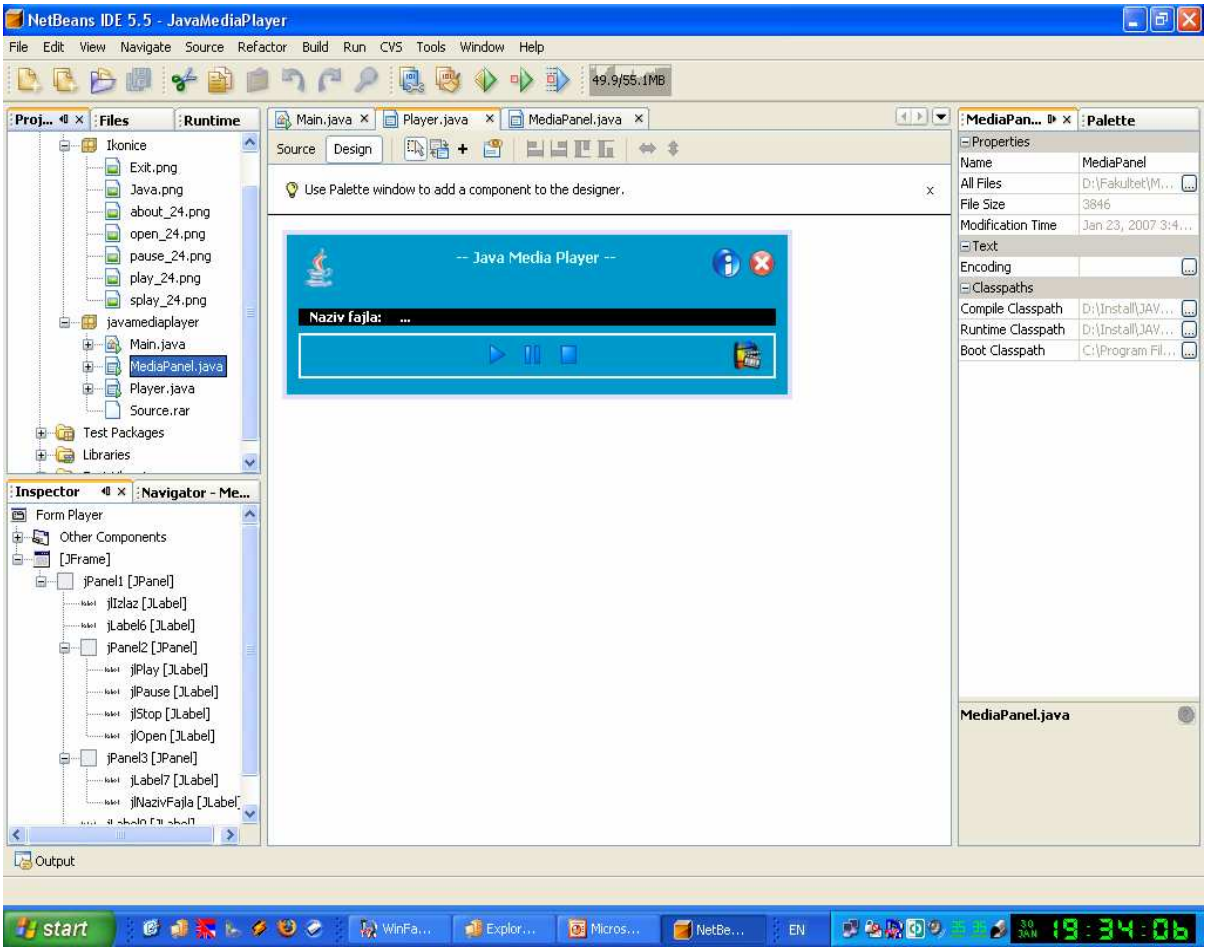
JMF output

Nakon procesne faze (faza obrade) dolazi se do izlazne faze JMF procesnog modela.
Sam izlazni posupak (output) implementiramo pomoću javax.media.DataSink interfejsa.

DataSink čita sadržaj medija i vrši rendering. DataSink se kreira kroz Manager klasu preko objekta klase DataSource (definisan izlaz iz DataSink) i preko objekta klase MediaLocator (definisan ulaz u DataSink).
Objekat klase DataSource predstavlja ulazni podatak, koji kasnije koristi Processor i DataSink. Output iz Processor-a je opet objekat tipa DataSource.



Na slici ispod se može vidjeti sam izgled plejera u IDE alatu, NetBeans, kao i izgled glavne klase Player, koja je sama po sebi GUI komponenta, koja proširuje JFrame.



Naš plejer se sastoji od 3 klase:

10 Main,

11 Player

12 MediaPanel

Klasa Main.java je osnovna klasa, koja u sebi sadrži main() metodu koja inicijalizuje glavni prozor i prikazuje plejer na ekranu.

Klasa Player je najbitnija klasa. U njoj je implementiran kompletan GUI za naš plejer, napravljene su potrebne instance media objekata, implementiran File Open... dijalog, kao i ostale operacije nad samim fajlom.

Uvećan prikaz glavnog prozora je dat ispod:

Dakle, na slici vidimo da su implementirane osnovne operacije nad fajlom: puštanje fajla, pauziranje, stopiranje, otvaranje novog fajla, info o programu i zatvaranje programa.

Za potrebe ovog plejera smo registrovali i MPEG-4 i FFOBS biblioteku, tako da je broj formata koje ovaj plejer čita zaista veliki.

Klasa MediaPanel u okviru Jpanel-a dodaje 2 komponente, jednu za video, drugu za kontrole.

Izgled plejera kada se pusti neki audio/video fajl unutar njega, je dat na sljedećoj slici:

Sama reprodukcija upotrebom FFOBS kodeka teče kao na slici dole:

U nastavku je dat kod naše aplikacije:

Main.java

```
/*
 * Main.java
 *
 * Created on January 22, 2007, 1:38 PM
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package javamediaplayer;

/**
 *
 * @author Salex
 */
public class Main {

    /** Creates a new instance of Main */
    public Main() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        new Player().setVisible(true);
    }

}
```

Player.java

```
/*
 * Player.java
 *
 * Created on January 22, 2007, 1:39 PM
 */

package javamediaplayer;

import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import javax.media.CannotRealizeException;
import javax.media.Manager;
import javax.media.MediaLocator;
import javax.media.NoPlayerException;
import javax.swing.JFileChooser;

/**
 *
 * @author  Salex
 */
public class Player extends javax.swing.JFrame {

    /** Creates new form Player */
    public Player() {
        initComponents();
        java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-427)/2, (screenSize.height-130-50), 427, 130);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">//GEN-BEGIN:initComponents
    private void initComponents() {
        jPanel1 = new javax.swing.JPanel();
        jlIzlaz = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jlPlay = new javax.swing.JLabel();
        jlPause = new javax.swing.JLabel();
        jlStop = new javax.swing.JLabel();
        jlOpen = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();
        jLabel7 = new javax.swing.JLabel();
```

```
    jlNazivFajla = new javax.swing.JLabel();
    jLabel9 = new javax.swing.JLabel();
    jlInfo = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setUndecorated(true);
    jPanel1.setBackground(new java.awt.Color(0, 153, 204));
    jlIzlaz.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Ikonice/Exit.png")));
    jlIzlaz.setToolTipText("Izlaz");
    jlIzlaz.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jlIzlazMouseClicked(evt);
        }
    });

    jLabel6.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Ikonice/Java.png")));

    jPanel2.setBackground(new java.awt.Color(0, 153, 204));
    jPanel2.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(255, 255, 255),
2));
    jlPlay.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Ikonice/play_24.png")));
    jlPlay.setToolTipText("Play");
    jlPlay.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jlPlayMouseClicked(evt);
        }
    });

    jlPause.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Ikonice/pause_24.png")));
    jlPause.setToolTipText("Pause");
    jlPause.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jlPauseMouseClicked(evt);
        }
    });

    jlStop.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Ikonice/splay_24.png")));
    jlStop.setToolTipText("Stop");
    jlStop.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jlStopMouseClicked(evt);
        }
    });

    jlOpen.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Ikonice/open_24.png")));
    jlOpen.setToolTipText("Otvori...");
    jlOpen.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jlOpenMouseClicked(evt);
        }
    });
```

```
        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(156, 156, 156)
                .addComponent(jlPlay)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jlPause)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jlStop)
                .addGap(129, 129, 129)
                .addComponent(jlOpen)
                .addContainerGap())
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(4, 4, 4)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jlOpen)
                    .addComponent(jlPlay)
                    .addComponent(jlStop)
                    .addComponent(jlPause))
                .addContainerGap())
        );

        jPanel3.setBackground(new java.awt.Color(0, 0, 0));
        jLabel7.setFont(new java.awt.Font("Tahoma", 1, 11));
        jLabel7.setForeground(new java.awt.Color(255, 255, 255));
        jLabel7.setText("Naziv fajla:");

        jlNazivFajla.setFont(new java.awt.Font("Tahoma", 1, 11));
        jlNazivFajla.setForeground(new java.awt.Color(255, 255, 255));
        jlNazivFajla.setText("...");

        javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel7)
                .addGap(16, 16, 16)
                .addComponent(jlNazivFajla)
                .addContainerGap(311, Short.MAX_VALUE))
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel7)
            .addComponent(jlNazivFajla))
    );

    jLabel9.setFont(new java.awt.Font("Tahoma", 1, 12));
    jLabel9.setForeground(new java.awt.Color(204, 255, 255));
    jLabel9.setText("-- Java Media Player --");

    jlInfo.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Ikonice/about_24.png")));
    jlInfo.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jlInfoMouseClicked(evt);
        }
    });

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(jLabel6)
                    .addGap(103, 103, 103)
                    .addComponent(jLabel9)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 82,
Short.MAX_VALUE)
                    .addComponent(jlInfo)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jlIzlaz))
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addContainerGap())
    );
    jPanel1Layout.setVerticalGroup(
        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jlIzlaz)
                .addComponent(jLabel6)
                .addComponent(jLabel9)
                .addComponent(jlInfo))
            .addGap(17, 17, 17)
            .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, 37,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(66, 66, 66))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, 130, Short.MAX_VALUE)
        );
        setBounds(0, 0, 427, 130);
    }// </editor-fold>//GEN-END:initComponents

    private void jlPlayMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jlPlayMouseClicked
// TODO add your handling code here:
        mediaPlayer.start();
    }//GEN-LAST:event_jlPlayMouseClicked

    private void jlStopMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jlStopMouseClicked
// TODO add your handling code here:
        mediaPlayer.close();
        mediaPanel.setVisible(false);
    }//GEN-LAST:event_jlStopMouseClicked

    private void jlPauseMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jlPauseMouseClicked
// TODO add your handling code here:
        mediaPlayer.stop();
    }//GEN-LAST:event_jlPauseMouseClicked

    private void jlOpenMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jlOpenMouseClicked
// TODO add your handling code here:
        JFileChooser f = new JFileChooser();
        int result = f.showOpenDialog(null);
        if (result == JFileChooser.APPROVE_OPTION) {
            mediaURL = null;
            try {
                mediaURL = f.getSelectedFile().toURI().toURL();
                jlNazivFajla.setText(f.getSelectedFile().getName());
            } catch (MalformedURLException ex) {
```

```
            ex.printStackTrace();
        }
        if (mediaURL != null) {
            try {
                // Create a JMF player to play the media specified in the URL:
                mediaPlayer = Manager.createRealizedPlayer(new MediaLocator(mediaURL));
                mediaPanel = new MediaPanel(mediaPlayer);
                mediaPanel.setVisible(true);
            } catch (CannotRealizeException ex) {
                ex.printStackTrace();
            } catch (NoPlayerException ex) {
                ex.printStackTrace();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }
    }//GEN-LAST:event_jlOpenMouseClicked

    private void jlInfoMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jlInfoMouseClicked
// TODO add your handling code here:
    }//GEN-LAST:event_jlInfoMouseClicked

    private void jlIzlazMouseClicked(java.awt.event.MouseEvent evt) {//GEN-
FIRST:event_jlIzlazMouseClicked
// TODO add your handling code here:
        setVisible(false);
        System.exit(1);
    }//GEN-LAST:event_jlIzlazMouseClicked

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /*java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Player().setVisible(true);
            }
        });*/
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JLabel jlInfo;
    private javax.swing.JLabel jlIzlaz;
```

```
    private javax.swing.JLabel jlNazivFajla;
    private javax.swing.JLabel jlOpen;
    private javax.swing.JLabel jlPause;
    private javax.swing.JLabel jlPlay;
    private javax.swing.JLabel jlStop;
    // End of variables declaration//GEN-END:variables
    private URL mediaURL;
    private javax.media.Player mediaPlayer;
    private MediaPanel mediaPanel;
}
```

MediaPanel.java

```
/*
 * MediaPanel.java
 *
 * Created on January 22, 2007, 9:07 PM
 */

package javamediaplayer;

import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Dimension;
import java.io.IOException;
import java.net.URL;
import javax.media.CannotRealizeException;
import javax.media.Manager;
import javax.media.MediaLocator;
import javax.media.NoPlayerException;
import javax.media.Player;

/**
 *
 * @author  Salex
 */
public class MediaPanel extends javax.swing.JFrame {

    /** Creates new form MediaPanel */
    public MediaPanel(Player mediaPlayer) {
        this.mediaPlayer = mediaPlayer;
        initComponents();

        // Get the components for the video and the playback controls:
        Component video = mediaPlayer.getVisualComponent();
        Component control = mediaPlayer.getControlPanelComponent();
```

```
        if (video != null )
            jpMediaPanel.add(video, BorderLayout.CENTER); // add video component
        videoDimension = jpMediaPanel.getPreferredSize();
        Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        if (videoDimension.width == 0 || videoDimension.height == 0)
            setBounds((screenSize.width-300)/2, (screenSize.height-55)/2, 300, 55);
        else
            setBounds((screenSize.width-videoDimension.width)/2, (screenSize.height-
videoDimension.height)/2, videoDimension.width, videoDimension.height);

        if (control != null)
            jpMediaPanel.add(control, BorderLayout.SOUTH);

        // Start the JMF player:
        mediaPlayer.start(); // start playing the media clip
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc=" Generated Code ">//GEN-BEGIN:initComponents
    private void initComponents() {
        jpMediaPanel = new javax.swing.JPanel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                formWindowClosing(evt);
            }
        });

        jpMediaPanel.setLayout(new java.awt.BorderLayout());

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jpMediaPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 291,
Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jpMediaPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 267,
Short.MAX_VALUE)
        );
    }// </editor-fold>//GEN-END:initComponents

    private void formWindowClosing(java.awt.event.WindowEvent evt) {//GEN-
```

```
FIRST:event_formWindowClosing
// TODO add your handling code here:
        mediaPlayer.close();
    }//GEN-LAST:event_formWindowClosing

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /*java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MediaPanel().setVisible(true);
            }
        });*/
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JPanel jpMediaPanel;
    // End of variables declaration//GEN-END:variables
    private Dimension videoDimension;
    private Player mediaPlayer;
}
```

REFERENCE

- http://ffmpeg.mplayerhq.hu/index.html
- http://fobs.sourceforge.net/index.html
- http://www.alphaworks.ibm.com/tech/mpeg-4
- http://www.wikipedia.org/
- http://java.sun.com/products/java-media/jmf/index.jsp
- http://www.netbeans.org/