Java Multimedia Framework

Reproducción y captura

Introducción
JMF

- API de Java que permite la captura reproducción, procesamiento y transmisión de medios continuos (video y sonido)
- Alto nivel de abstracción en el diseño
  - Trata de igual forma a sonido y vídeo
- Gestiona varios formatos y códecs
- Incorpora controles visuales

Introducción

Principales clases

Fuentes de datos

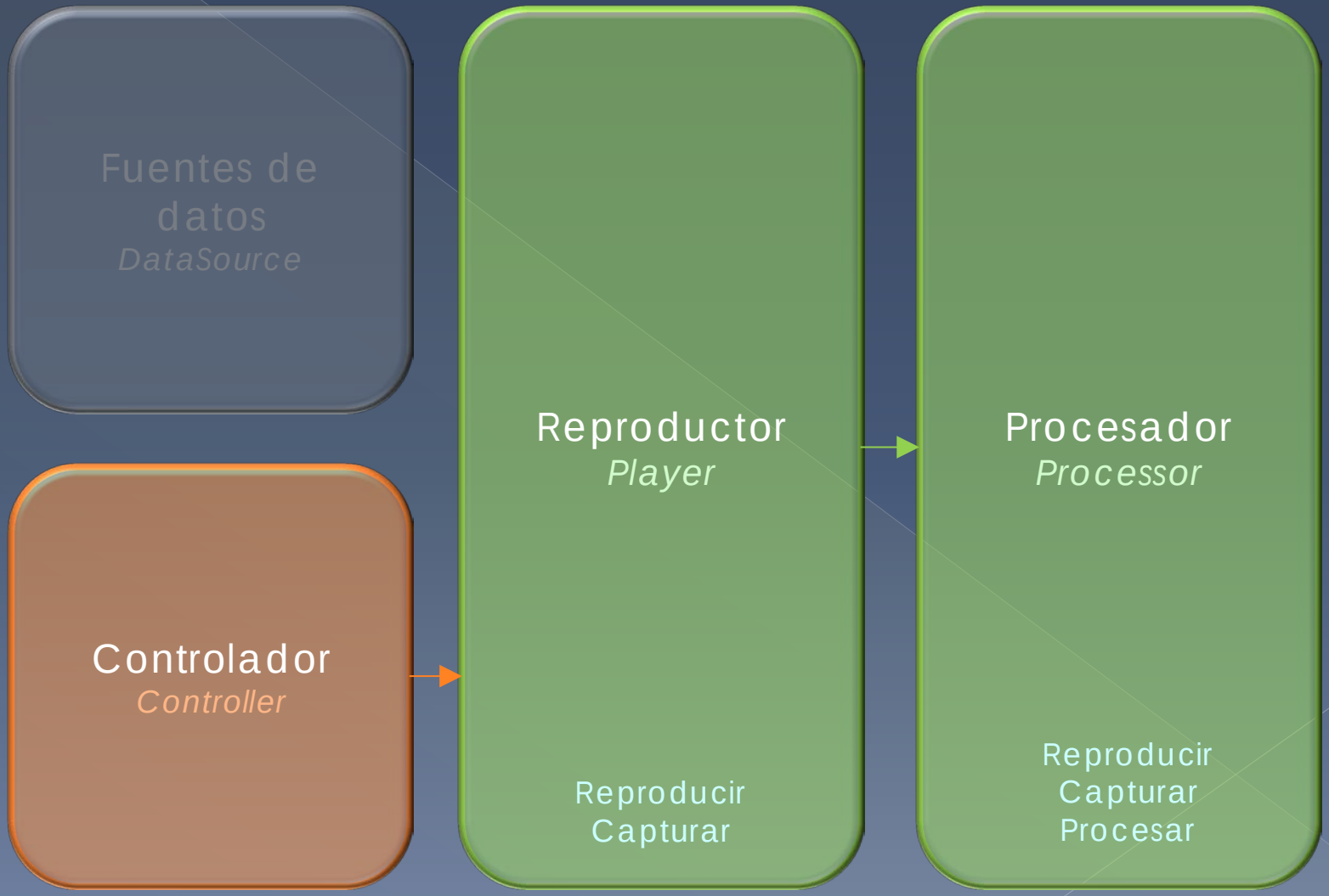
DataSource

Controlador

Controller

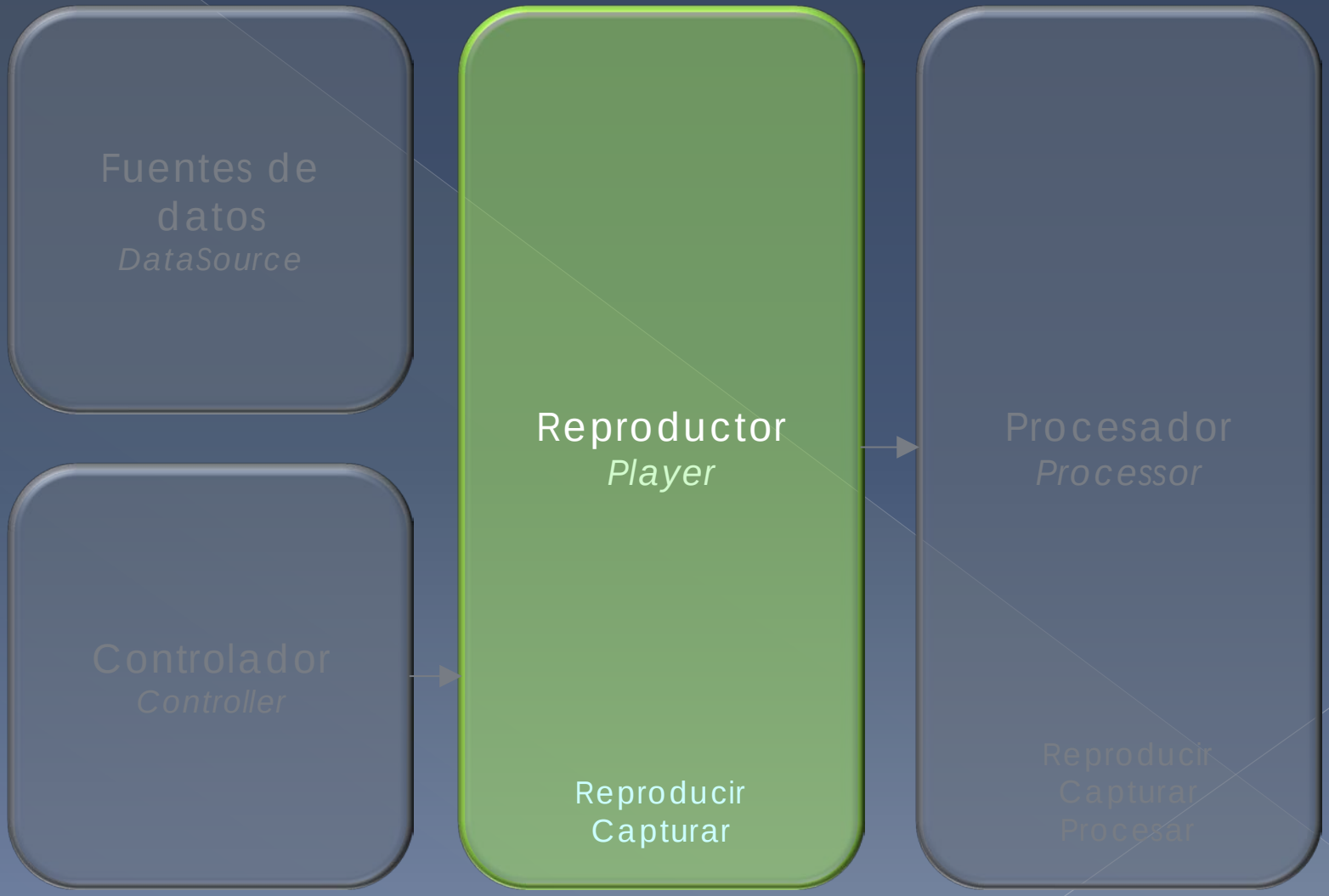
Introducción

Principales clases

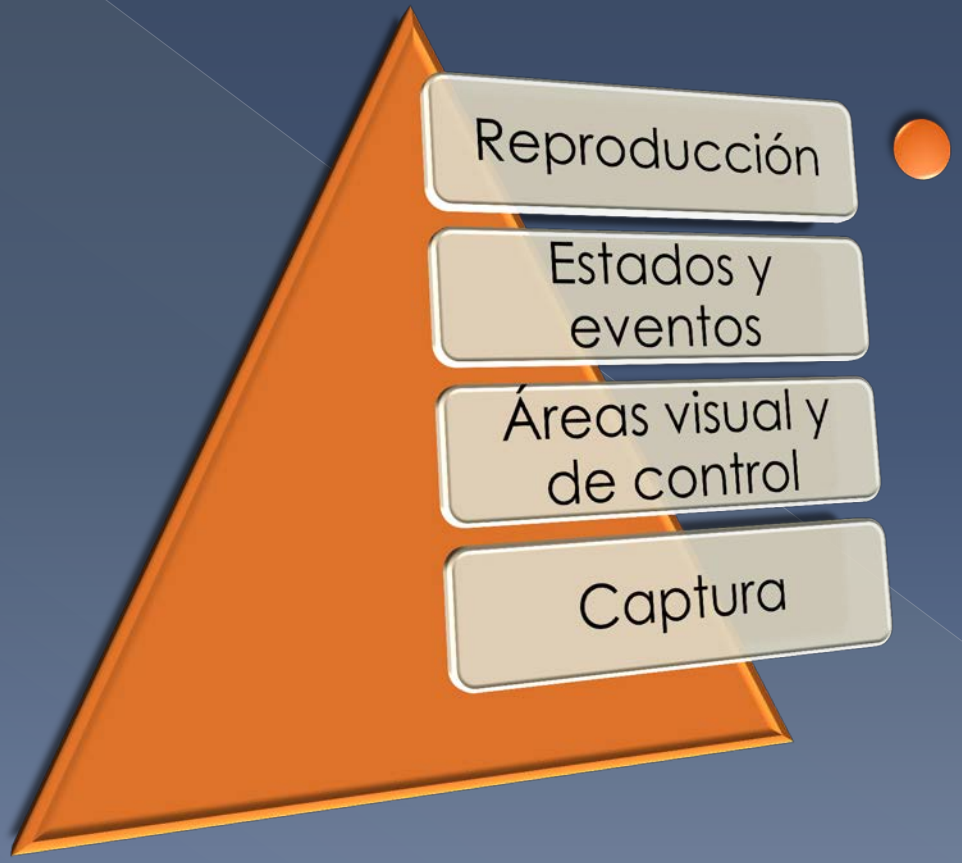


Introducción

Principales clases



Índice


Reproducción
Estados y eventos
Áreas visual y de control
Captura

Reproducción

Player

```
MediaLocator ml = new MediaLocator(``c:/sonido.wav");
```

1º

Localización del medio

- Sonido
  - WAVE
  - AIFF
  - AU
  - RMF
  - MP3
  - MIDI
- Vídeo
  - MPEG
  - MOV
  - AVI

Reproducción

Player

```
public void play(){
  try {
        MediaLocator ml = new MediaLocator(``c:/sonido.wav");
        Player p = Manager.createPlayer(ml);

        p.start();
  } catch(Exception e) {}
}
```

1º Localización del medio

2º Creación del reproductor

3º Comienzo de la reproducción

Reproducción

Player

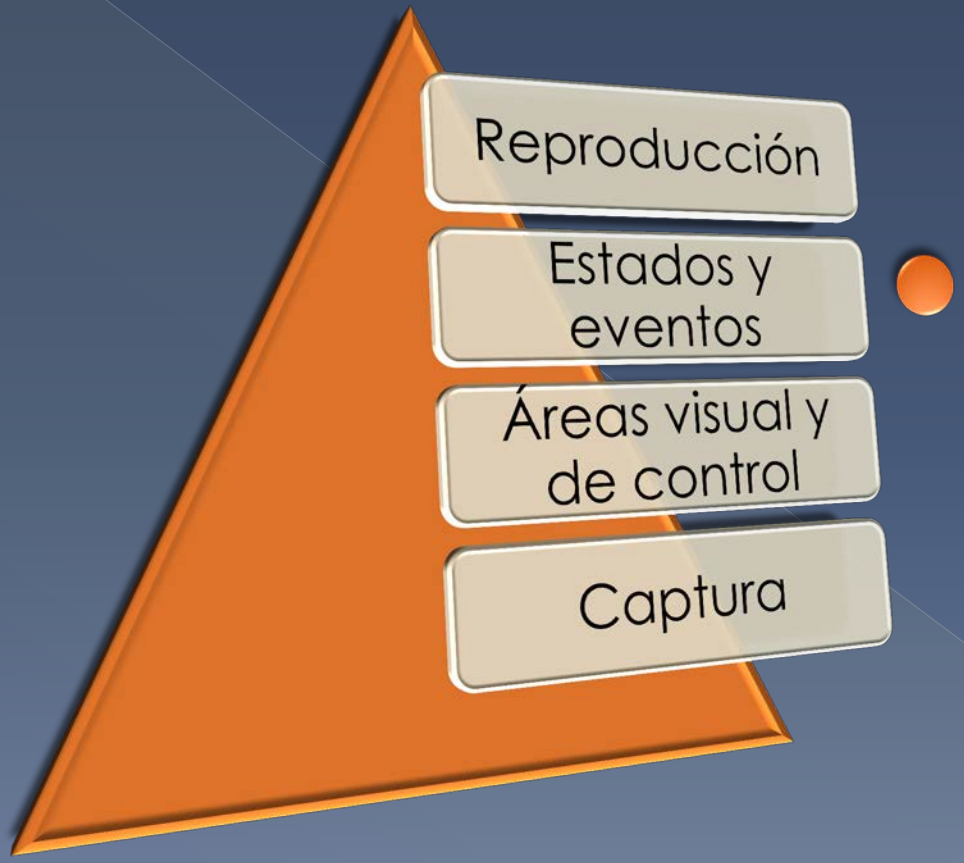
```
public void play(File f){
  try {
        MediaLocator ml = new MediaLocator(f.getAbsolutePath());
        Player p = Manager.createPlayer(ml);

        p.start();
  } catch(Exception e) {}
}
```

1º Localización del medio

2º Creación del reproductor

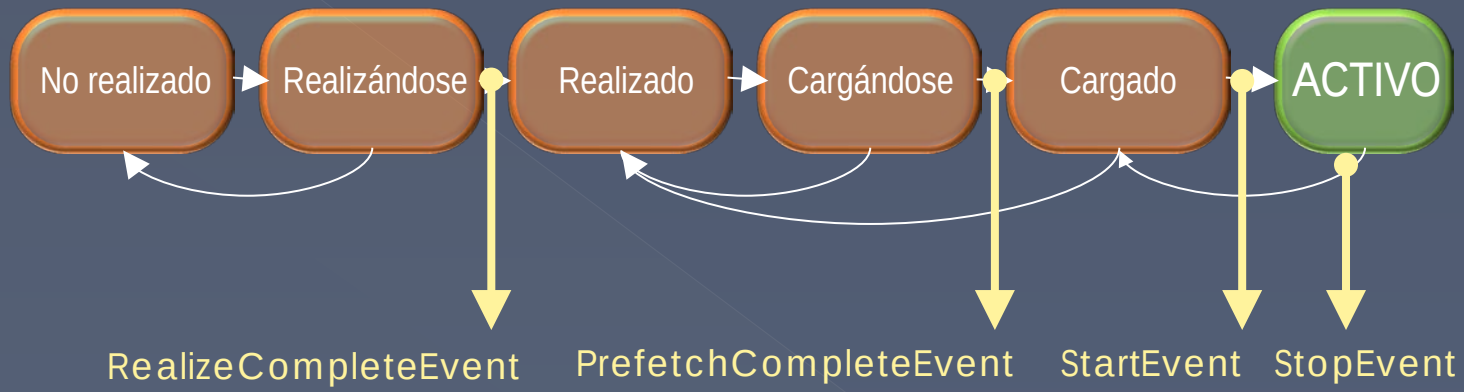
3º Comienzo de la reproducción

Índice


Reproducción
Estados y eventos
Áreas visual y de control
Captura

Estados y eventos

Estados

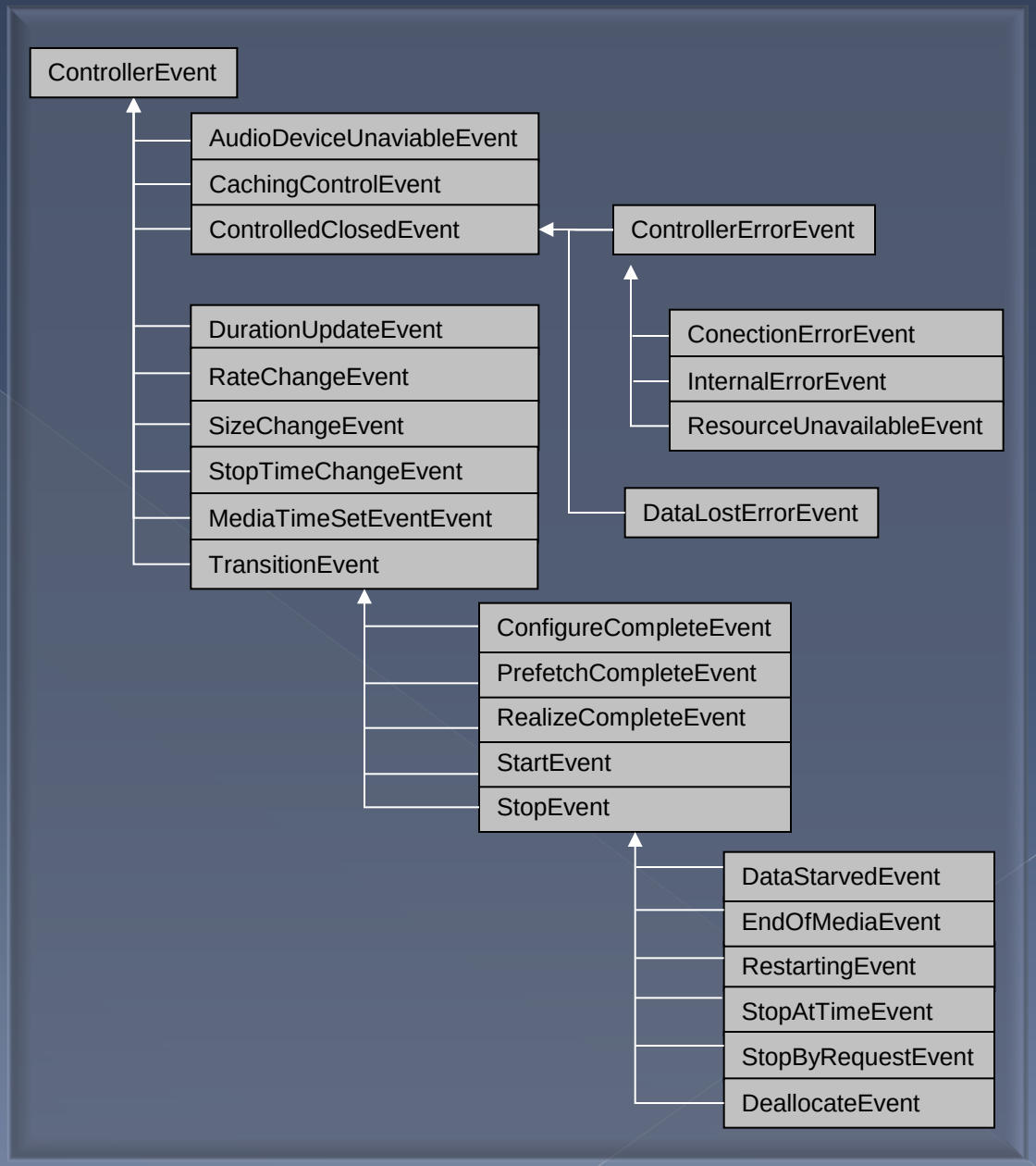


No realizado	Estado inicial. No se sabe nada sobre el medio a reproducir
Realizándose	Determina y reúne los recursos necesarios para reproducir
Realizado	Conoce el tipo de medio y los recursos que necesita
Cargándose	Trayendo el medio al búfer de memoria
Cargado	El medio está en memoria. Listo para reproducir
Activo	Reproduciendo el medio

Eventos

Player

- Cambio en atributo
- Cierre del Player
- Transición de estado



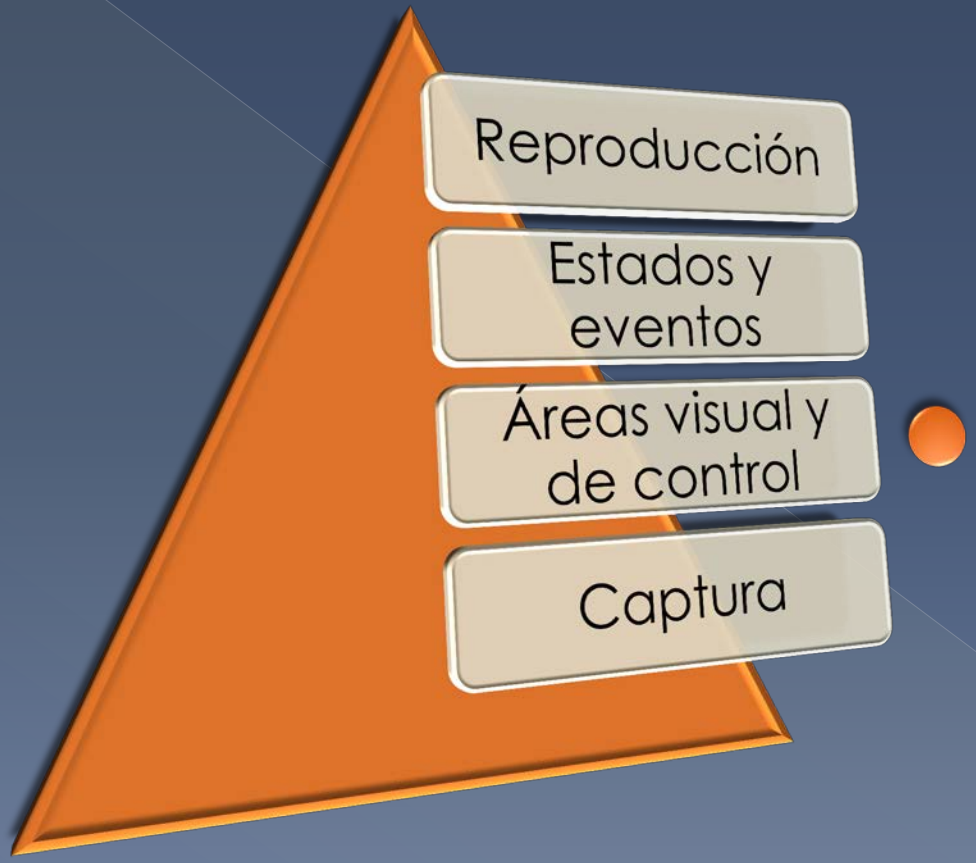
Estados y eventos

Manejador de eventos

```
class miManejadorMedia extends ControllerAdapter {

    public void realizeComplete (RealizeCompleteEvent e) {
        /* Código */
    }
    public void prefetchComplete (PrefetchCompleteEvent e) {
        /* Código */
    }
    public void start(StartEvent e) {
        /* Código */
    }
    public void stop(StopEvent e) {
        /* Código */
    }
    .
    .
    .
}
```

```
player.addControllerListener(new  miManejadorMedia());
```

Índice


Reproducción
Estados y eventos
Áreas visual y de control
Captura

Áreas visual y de control

Componentes

```
Component aVisual = player.getVisualComponent();
contenedor.add(aVisual,...);
```

Área Visual

```
Component aControl = player. getControlPanelComponent();
contenedor.add(aControl,...);
```

Área Control

Para acceder a las áreas visual y de control, el reproductor ha de estar ***realizado****. Por ello, habrá que:*

1. *Gestionar eventos e introducir el código anterior en el método realizeComplete()*
2. *O usar el método Manager.createRealizedPlayer(MediaLocator) para crear el reproductor*

Áreas visual y de control

Añadir áreas usado manejador

```
public void play(){
  try {
        MediaLocator ml = new MediaLocator(``c:/sonido.wav");
        Player p = Manager.createPlayer(ml);
        p.addControllerListener(new miManejadorMedia());
        p.start();
  } catch(Exception e) {}
}
```

```
class miManejadorMedia extends ControllerAdapter{
   public void realizeComplete (RealizeCompleteEvent e) {
      Component  areaVisual = p.getVisualComponent();
      if(areaVisual!=null) contenedor.add(areaVisual,...);
      Component  panelControl = p.getControlPanelComponent();
      if(panelControl!=null) contenedor.add(panelControll,...);
   }
}
```

Áreas visual y de control

Añadir áreas usando createRealizedPlayer

```
public void play(){
  try {
        MediaLocator ml = new MediaLocator(``c:/sonido.wav");
        Player p = Manager.createPlayer(ml);

        p.start();
  } catch(Exception e) {}
}
```

Áreas visual y de control

Añadir áreas usando createRealizedPlayer

```
public void play(){
  try {
        MediaLocator ml = new MediaLocator(``c:/sonido.wav");
        Player p = Manager.createRealizedPlayer(ml);

        p.start();
  } catch(Exception e) {}
}
```

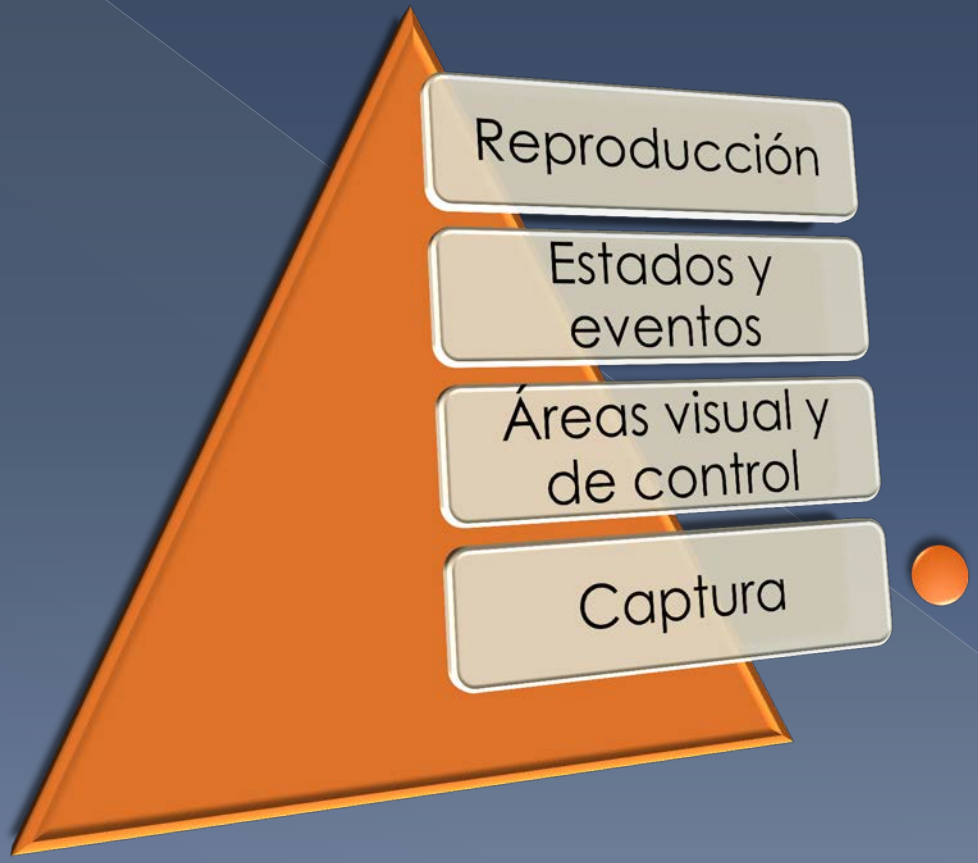
En este punto, podemos asumir que el Player está ya realizado

Áreas visual y de control

Añadir áreas usando createRealizedPlayer

```
public void play(){
  try {
        MediaLocator ml = new MediaLocator(``c:/sonido.wav");
        Player p = Manager.createRealizedPlayer(ml);
        Component  areaVisual = p.getVisualComponent();
        if(areaVisual!=null) contenedor.add(areaVisual);
        Component  panelControl = p.getControlPanelComponent();
        if(panelControl!=null) contenedor.add(panelControll);
        p.start();
  } catch(Exception e) {}
}
```

Índice


Reproducción
Estados y eventos
Áreas visual y de control
Captura

Captura
Punto de partida

```
public void play(){
  try {
      MediaLocator ml = new MediaLocator(``c:/sonido.wav");
      Player p = Manager.createRealizedPlayer(ml);
      Component  areaVisual = p.getVisualComponent();
      if(areaVisual!=null) contenedor.add(areaVisual);
      Component  panelControl = p.getControlPanelComponent();
      if(panelControl!=null) contenedor.add(panelControll);
      p.start();
  } catch(Exception e) {}
}
```

Hasta ahora, el localizador del medio lo hemos asociado a un fichero, pero también puede asociarse a un dispositivo, por ejemplo, una cámara

Captura

Nombre de dispositivo conocido

```
public void play(){
  try {
      CaptureDeviceInfo deviceInfo;
      String dName="vfw:Microsoft WDM Image Capture (Win32):0"
      deviceInfo = CaptureDeviceManager.getDevice(dName);


      MediaLocator ml = deviceInfo.getLocator();
      Player p = Manager.createRealizedPlayer(ml);
      Component  areaVisual = p.getVisualComponent();
      if(areaVisual!=null) contenedor.add(areaVisual);
      Component  panelControl = p.getControlPanelComponent();
      if(panelControl!=null) contenedor.add(panelControll);
      p.start();
  } catch(Exception e) {}
}
```

Acceso a cámara

Captura

Acceso a lista de dispositivos

```
public void play(){
  try {
      CaptureDeviceInfo deviceInfo;
      List<CaptureDeviceInfo> deviceList →
      = CaptureDeviceManager.getDeviceList(new YUVFormat());
      deviceInfo = deviceList.get(0);

      MediaLocator ml = deviceInfo.getLocator();
      Player p = Manager.createRealizedPlayer(ml);
      Component areaVisual = p.getVisualComponent();
      if(areaVisual!=null) contenedor.add(areaVisual);
      Component panelControl = p.getControlPanelComponent();
      if(panelControl!=null) contenedor.add(panelControll);
      p.start();
  } catch(Exception e) {}
}
```

Acceso a cámara

Captura de un frame

Código

```
public BufferedImage getFrame(Player player){
    FrameGrabbingControl fgc;
    String  claseCtr = "javax.media.control.FrameGrabbingControl ";
    fgc = (FrameGrabbingControl)player.getControl(claseCtr );
    Buffer bufferFrame = fgc.grabFrame();
    BufferToImage bti;
    bti=new BufferToImage((VideoFormat)bufferFrame.getFormat());
    Image img = bti.createImage(bufferFrame);
    return (BufferedImage)img;
}
```

Graba frame en buffer

Convertimos buffer a imagen

Suena bien, pero…
JMF

- Aunque JMF permite la gestión de video/audio de una forma sencilla, su última actualización es de 2003, lo que plantea una serie de problemas:
  - No soporta los códecs y formatos más actuales (mp4, mkv, etc.), solo a aquellos previos al 2003.
  - Da problemas a la hora de detectar dispositivos de captura (webcam, micrófono, etc.) en los sistemas más modernos.
  - Está desarrollado para arquitecturas de 32 bits, lo que provoca incompatibilidades en sistemas de 64 bits.
- Por este motivo, se suele recurrir a otras bibliotecas de código abierto más actualizadas que, si bien no están estandarizadas por Oracle, permitan solventar los problemas anteriores.

Webcam Capture API

Cámara

- Esta API permite, de manera muy sencilla, acceder a la webcam e integrarla en nuestra aplicación.

```
Webcam camara = Webcam.getDefault();
JPanel areaVisual = new WebcamPanel(camara);
.
.
contenedor.add(areaVisual, ...);
```

Devuelve la primera cámara disponible en el sistema. Se puede obtener la lista de cámaras mediante el método getWebcams();

```
BufferedImage img = camara.getImage();
```

https://github.com/sarxos/webcam-capture

Biblioteca VLCj

VLC

- VLC Media Player es un software de reproducción multimedia, libre y de código abierto, desarrollado por VideoLAN.

- La biblioteca VLCj, también de código abierto, ofrece la posibilidad de llamar al código nativo de VLC y crear un reproductor embebido en un panel
  - La principal ventaja de este enfoque es que tenemos acceso a formatos y códecs actualizados de una forma sencilla
  - El inconveniente es que depende de librerías nativas, por lo que hay que tener instalado VLC en nuestro sistema. Además, es obligatorio ejecutar nuestra aplicación usando una máquina virtual de JAVA con la misma arquitectura (32 o 64 bits) que la del VLC instalado.

https://github.com/caprica/vlcj

Biblioteca VLCj

VLC

```
boolean ok = new NativeDiscovery().discover();
.
.
EmbeddedMediaPlayerComponent areaVisual;
areaVisual = new EmbeddedMediaPlayerComponent();
contenedor.add(areaVisual,...);
.
.
EmbeddedMediaPlayer player;
player = areaVisual.getMediaPlayer();

    player.playMedia(String)
    player.play()
    player.pause()
    player.stop()
    ...
```

A

B

C

https://github.com/caprica/vlcj