Journal of Kerbala University , Vol. 8 No.4 Scientific . 2010

Design and Implementation of Java Media Player

Muhanad Hayder
University of Karbala – College of Science
Department of Computer Science
E-mail: muhanadeng2005@gmail.com

Abstract

As the technology evolving the need for media communication and interchangeability increased, Result in introduce a vast variety of media files with different type and format incurring different software for processing these files.

As the name suggests this paper presents the design and implementation of Media Player which is software used to play different media files (audio and video). It's accept media file format like (mp3, wav, mpg, mpeg, mov,…etc). This Media Player can works on any operating system (Windows, Linux, Mac,…..,etc) because it is written in J2SE (Java 2 Standard Edition) language which is a platform independent language, along with JMF (Java Media Framework) class library for supporting a wide rang of media files.

The aim to design media player as possible as like Windows Media Player but it have owns characteristics likes changing the speed of the player with different speed, can also change the size of the video screen in different scale (big, small, full screen), can also playing online stream media file from Internet, and other features. The prerequisite of the System JMF install for respective operating System (JMF available for most operating system).

Keywords: *Java Media Player, JMF, Platform Independent, and J2SE.*

الخلاصة

كلما تتقدم تكنلوجيا المعلومات كلما يزداد الاحتياج الى تطور وسائل الاتصال والتبادل المعلوماتي والتي تتطلب التعامل مع ملفات الصوت والصورة ومعالجتها بصورة فعالة مما ادى الى تنوع مثل هذه الملفات واختلاف طرق معالجتها. وكما نلاحظ اليوم مدى انتشار مثل هذة البرامج كمشغلات الوديو والفيديو وبرامج المحادثة والاتصال المرئي.

الغرض من هذا البحث هو تصميم وتنفيذ مشغل لجميع انواع ملفات الصوت والصورة على اختلافها. بالاضافة الى اهم خاصية يتمتع بها هذا البرنامج كونه (Platform Independent) بسبب استخدام لغة جافا اصدار(J2SE) لبرمجة هذا السوفتوير حيث يمكن استخدامة في اي نظام تشغل مثل Windows, Mac, Linux, Solaris. وهي من اهم المشاكل التي يعالجها فهناك بعض نظم التشغل التي لاتحتوي على مشغل لملفات الصوت والصورة مثل (RedHat Linux).

يستخدم البرنامج (JMF class Library) التي تستخدم لمعالجة جميع انواع ملفات الميديا (Media Files). ولهذا البرنامج يحتاج الى تنصيب JMF لنظام التشغيل المعني قبل استخدامة (JMF متوفر لجميع نظم التشغل) ولهذا السبب للبرنامج القابلية على العمل عدد كبير من نظم التشغيل ومعالجة عدد كبير من انوع ملفات الصوت والصورة.

1. Introduction

In media player world the most popular one is Windows Media Player (WMP). It is a digital media player and media library application developed by Microsoft that is used for playing audio, video and viewing images on personal computers running the Microsoft Windows operating system, as well as on Pocket PC and Windows Mobile-based devices. Editions of Windows Media Player were also released for Mac OS, Mac OS X and Solaris but development of these has since been discontinued [1].

As WMP is create software for player audio and video file but being tightly couple with windows platform and the limited numbers of file types that it's can play. Many venders take the responsibility to designing media player that could run almost any type of media file and can run on different operating system like Apple and SunMicrosystem.

This paper address this problem for being design a platform independent media player (since its implements in java language which itself platform independent) and can play many types of media file (since it use JMF classes which support a vast number of media files).

2. Media files format

There is many media files format using in multimedia application, but not all media players types can supported it like (compare between all three media players QuickTime, RealPlayer and Windows Media Player). File Formats Supported by Windows Media Player 6.4,
.AVI, .WAV, .MP3, .M3U,.IVF, .MPEG, .MPG, .MPE,.M1V, .MP2, .MPV2, .MP2V,.MPA, .MID, .MIDI, .QT,.MOV, .AIFF, .AIFC, .AU,.SND, .ASF,.ASX, .WM,.WMA, .WAX, .WMV, .WVX, and others
File Formats Supported by Apple QuickTime 4 is:-
.QT, Flash, .AVI,.MP3,.WAV, .AIFF, .DV, .GIF,.MPEG, .JPEG, .TIFF,.MIDI, .SD2, .AU, .FLC,.MOV, .BMP, .FPX, Photoshop, Motion .JPEG, Open DML, .PNG, .GX,.TGA, .SGI, .PICT, and others.
File Formats Supported by Real Player G2:-
.AIFF, .ASF, .AU, .AVI,.MIDI, .MOV, .MPEG-1,.RA, .RV, .RM, .SwF,WAV
Real Jukebox:- Basic: .AIF, .AU, .RA,.RM, MP3, .PLS, .M3U,CDs, Real Jukebox, A2B,Liquid Audio [3].
As we can see almost all media players could not play all the file types, here is the need for player which can player as much files as possible for this purpose JMF classes are coming to the picture which has the ability to handle most media file type.
We done nothing greet concern this aspect other than gathering these classes in one module for handling most media files in this player.

3. JMF

- The Java Media Framework (JMF) is an application programming interface (API) for incorporating time-based media into Java applications and applets.
- Current version JMF 2.1.1.
- JMF contains classes that provide support for RTP (real-time Transport Protocol).
    - RTP enables the transmission and reception of real-time media streams across the network.
    - RTP can be used for media-on-demand applications as well as interactive services such as Internet telephony [2].

JMF install on top of JVM (Java Virtual Machine) with architecture as shown in figure (1).

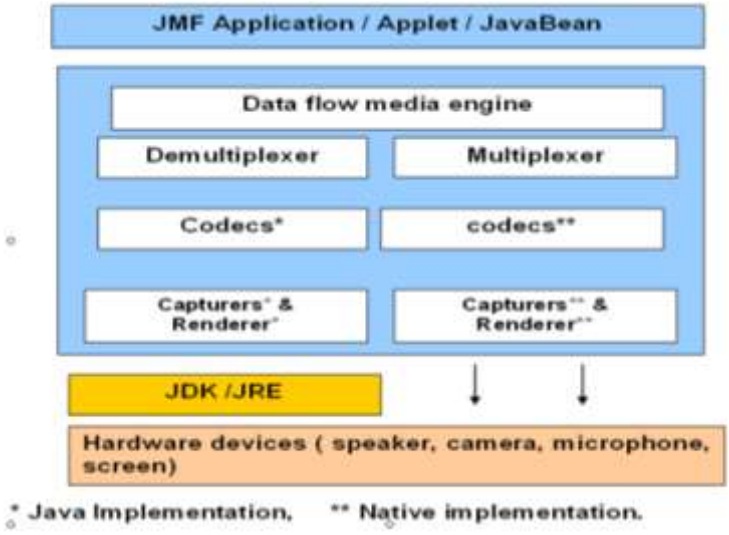


Figure (1) JMF Application [2]

Following are main JMF classes used in this paper:-

3.1 Player

Player is a Media Handler for rendering and controlling time based media data. Player extends the Controller interface. Player provides methods for obtaining AWT components, media processing controls, and a way to manage other Controllers. Player relaxes some restrictions that a Controller imposes on what methods can be called on a Started Controller. It also provides a way to manage groups of Controllers. Methods Restricted to Stop Players [2].

3.1.1 Player Life cycle

The life cycle of the player is shown in figure (2) and it is consisted of the following states:
1- ***UnRealized*** state : Player has no knowledge of the media it is supposed to handle.
2- ***Realizing*** state : Player identifies and acquires the resources it needs to playback the media.
3- ***Prefetching*** state : Player perfected the media data[5].

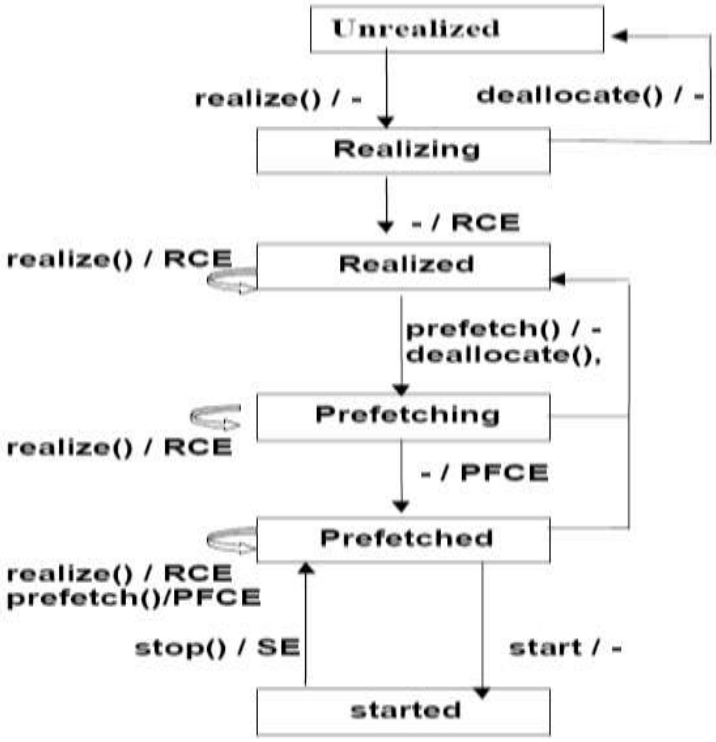


Figure (2) Player Life Cycle [5]

3.3 Manger

JMF uses intermediary objects called managers to integrate new implementations of key interfaces that define the behavior and interaction of objects used to capture, process, and present time-based media.

- JMF uses four managers:
    1. ***Manager***: handles the construction of Players, Processors, Data Sources, and Data Sinks.
    2. ***Package Manager:*** maintains a registry of packages that contain JMF classes, such as custom Players, Processors, Data Sources, and Data Sinks.
    3. ***Capture Device Manager***: maintains a registry of available capture devices.
    4. ***Plug in Manager***: maintains a registry of available JMF plug-in processing components, such as Multiplexers, Demultiplexers, Codes, Effects, and Renders [5].

4. System design and Implementation

4.1 System Design

The UML diagrams show below depicted all the software functionality from OOP point of view. We have been drawing them with the help of [6,7] because all software projects represented in terms of these diagrams(no longer using flow chart and other primitive way for representing the software). The Java Media Player System design is showing in figures (3, 4, 5, 6, 7, 8 and 9).

4.1.1 Class Diagram: There are many classes but we have represented the main ones. Class ***MediaPlayer*** inherited from ***Player*** class for gaining all the properties and methods for performing player functionalities. As well as inherited from class ***EventHandler*** for catching all the events triggered on ***Player*** class. There is association relationship between ***File*** class and ***MediaPlayer*** class as shown in figure (3).

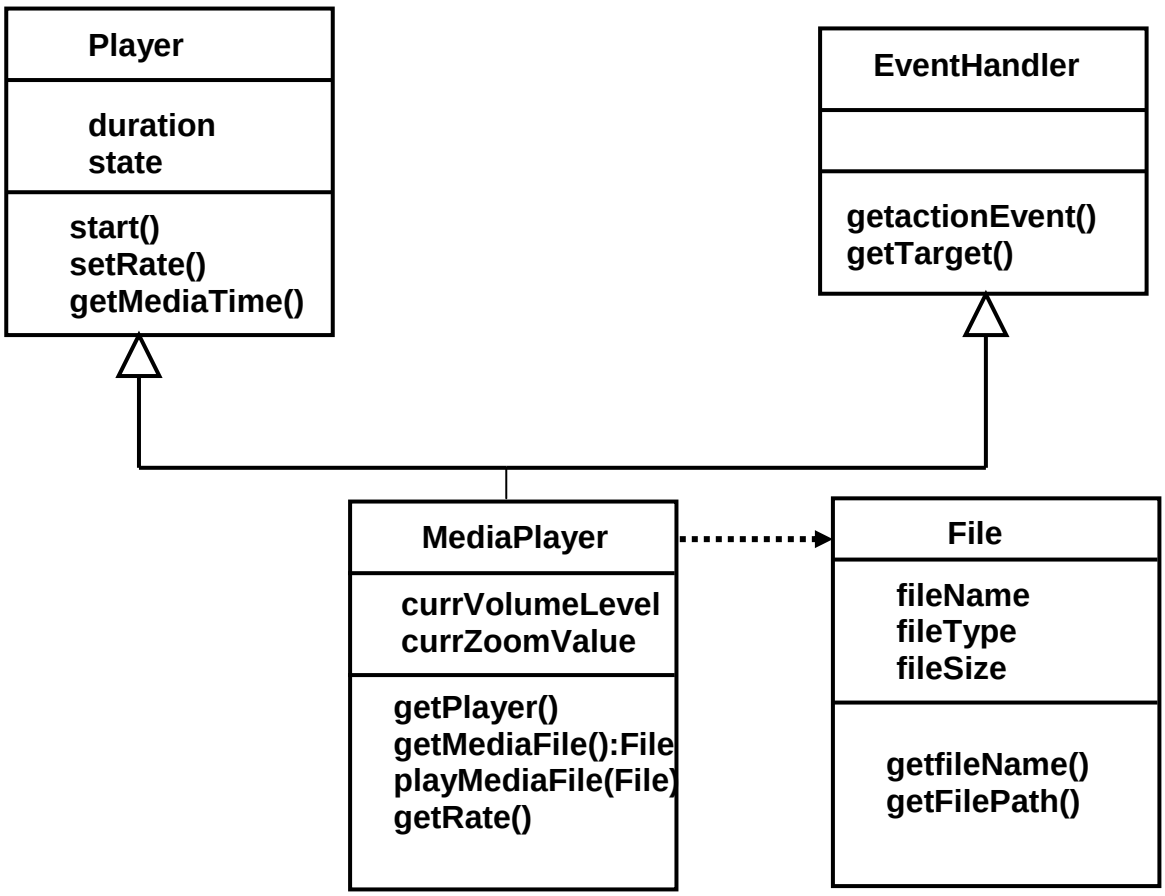


Figure (3): Class Diagram

4.1.2 Object Diagram: showing the instances of classes above along with their instantaneous values and variables name. ***MediaPlayer*** object (m) has ***one to many*** relationship with ***File*** object. As shown in figure (4).

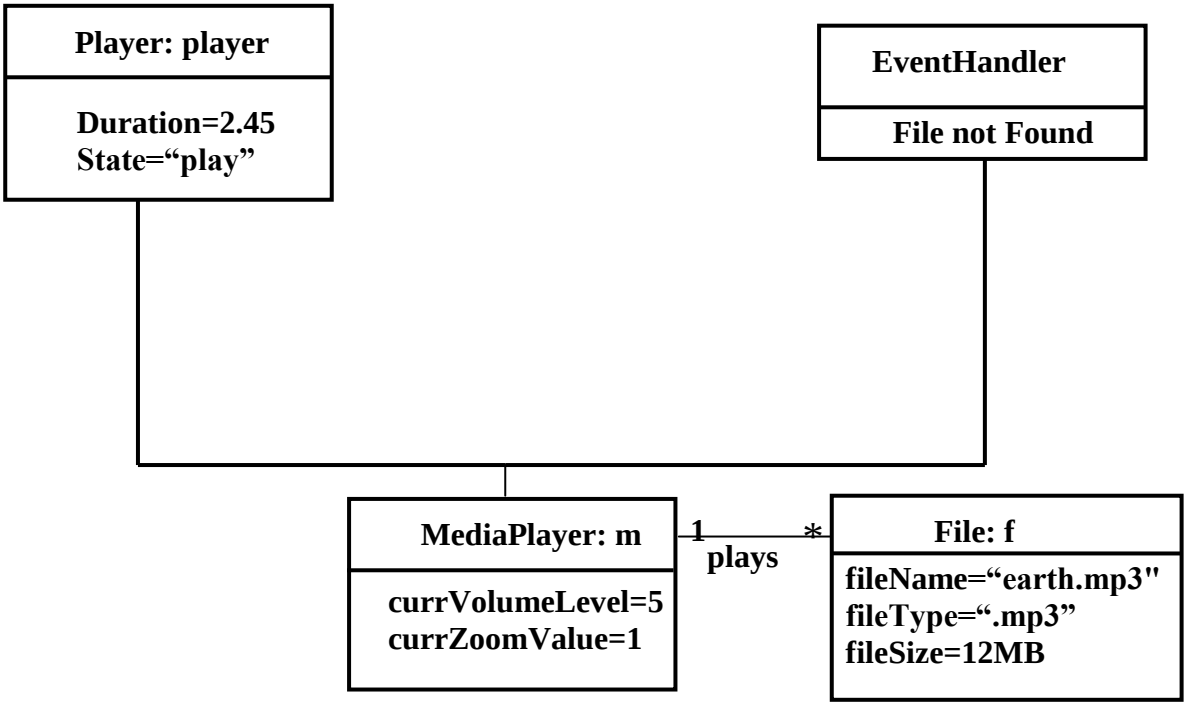


Figure (4): Object Diagram

4.1.3 Use Case Diagram: showing different scenarios the user (actor) interact with the system from behavior (functional) point of view, as shown in figure (5).

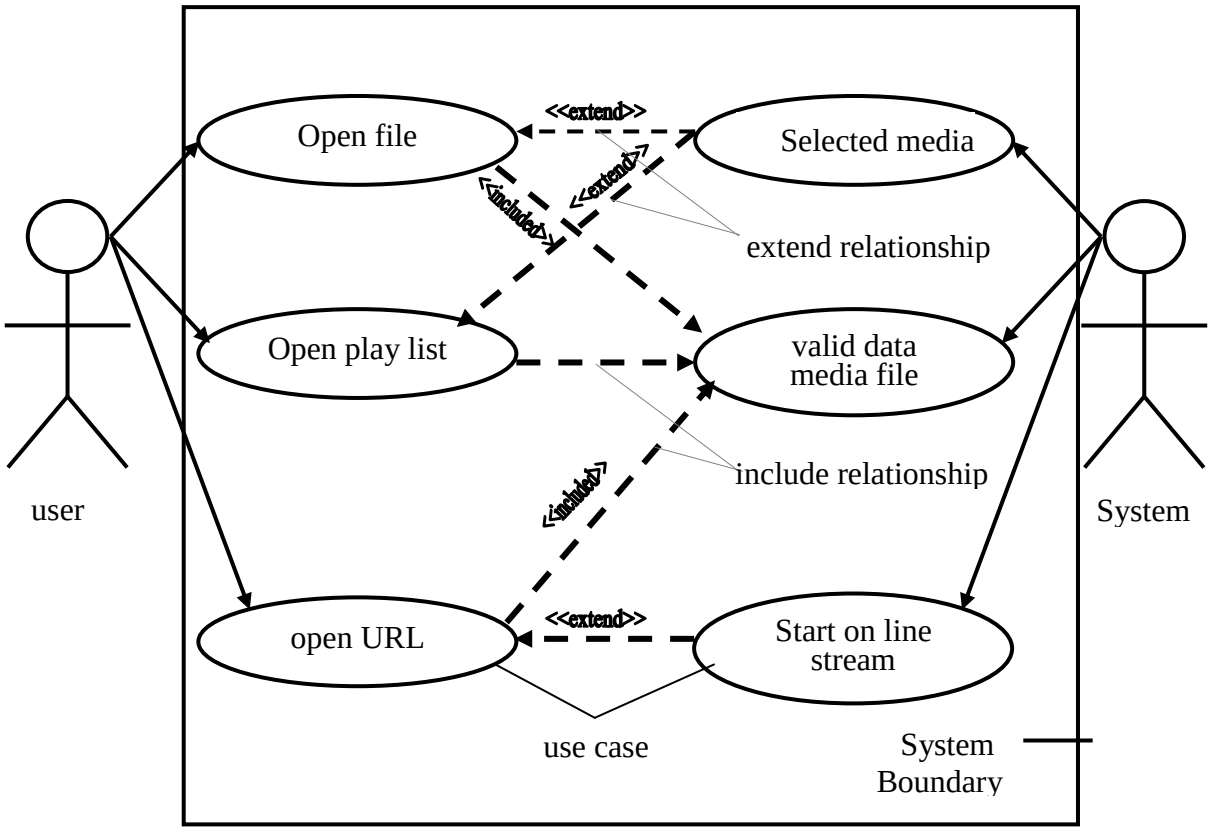


Figure (5): Use Case

4.1.4 Sequence Diagram: showing the sequence of interaction process between ***User*** and ***MediaPlayer*** instance (m) base on their event, the dashed lines hanging from the boxes are called object lifelines, representing the life span of the object during the scenario being modeled. The long, thin boxes on the lifelines are activation boxes, also called method-invocation boxes as shown in figure (6).

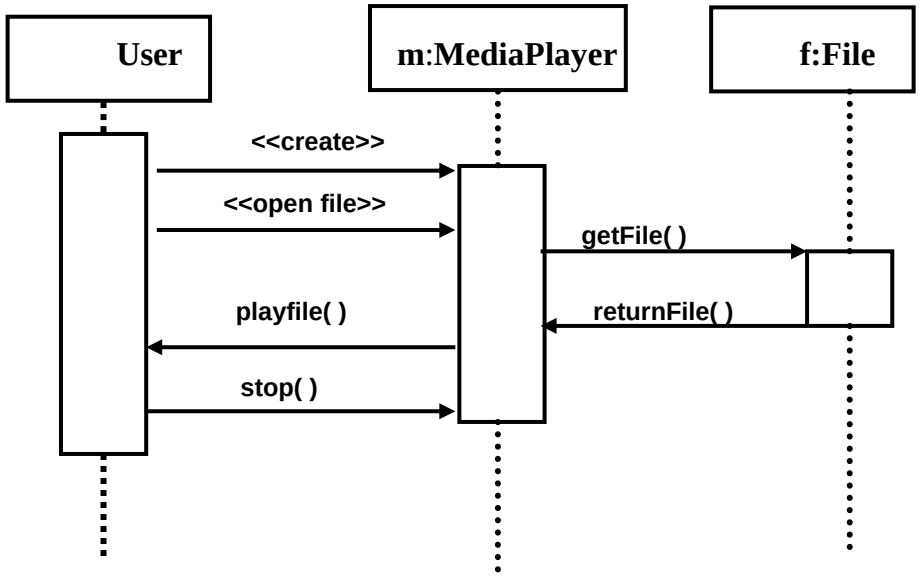


Figure (6): Sequence Diagram

4.1.5 Collaboration Diagram: demonstrate the relationship between objects and the order of messages passed between them. The numbers next to the messages are called sequence numbers. They show the sequence of the messages as they are passed between the objects (***User***, ***MediaPlayer*** and ***File***). As shown in figure (7).

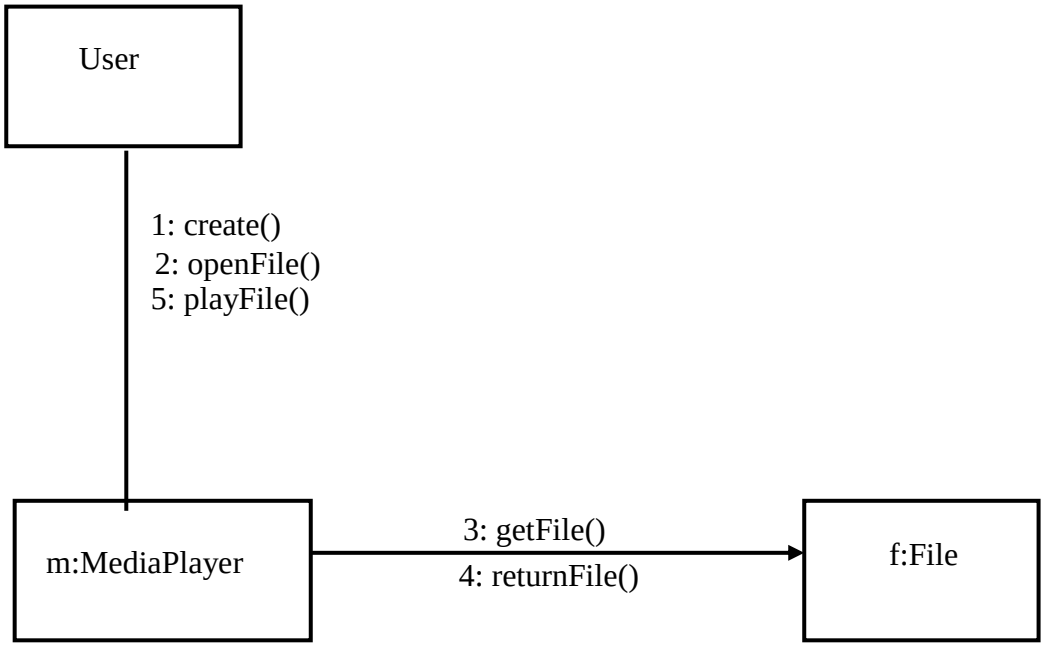


Figure (7): Collaboration diagram

4.1.6 State Diagram: demonstrate the behavior of an object through many use cases of the system which describe all of the possible states of an object as events occur. Rounded boxes representing the state of the object and arrows indicting the transition to the next state. As shown in figure (8).

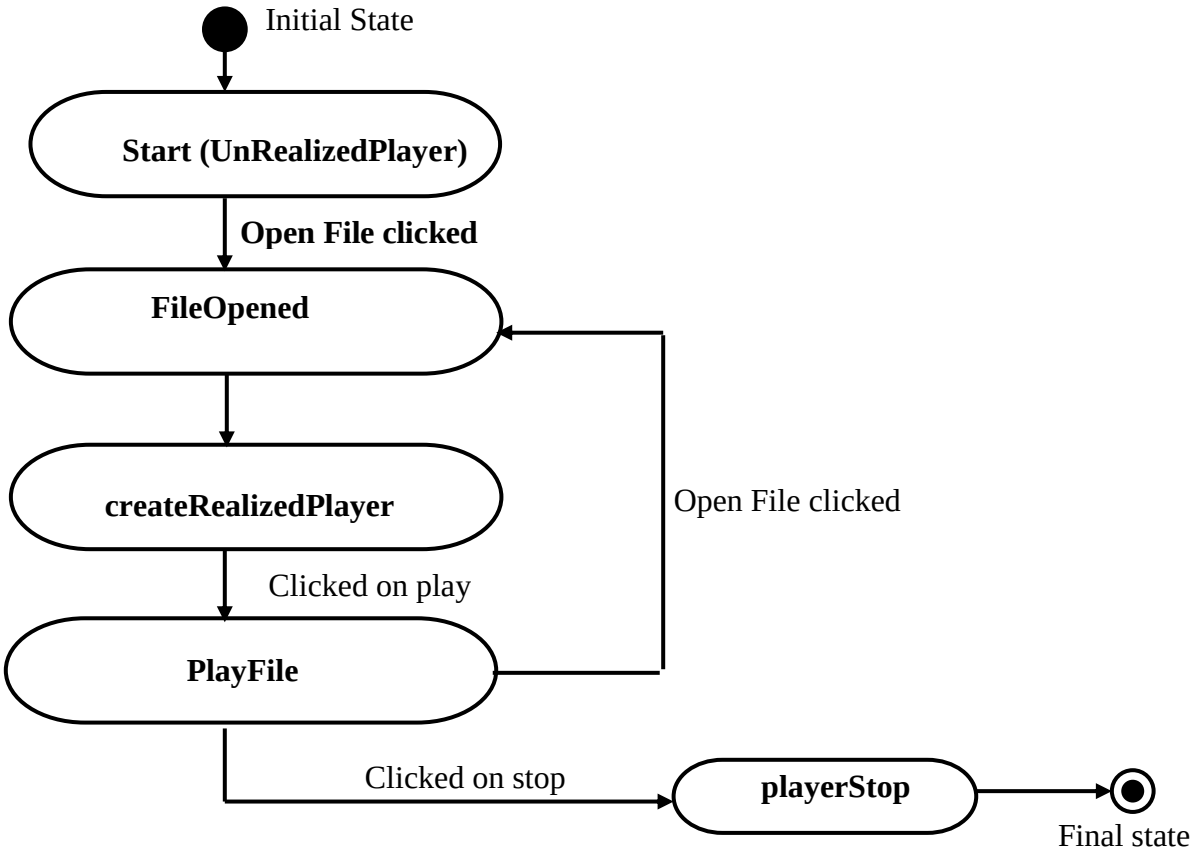


Figure (8): State diagram

4.1.7 Activity Diagram: demonstrate the workflow behavior of a system which describes the state of activities by showing the sequence of activities performed in different order. The diagram has branches and forks to describe conditions and parallel activities. As shown in figure (9).

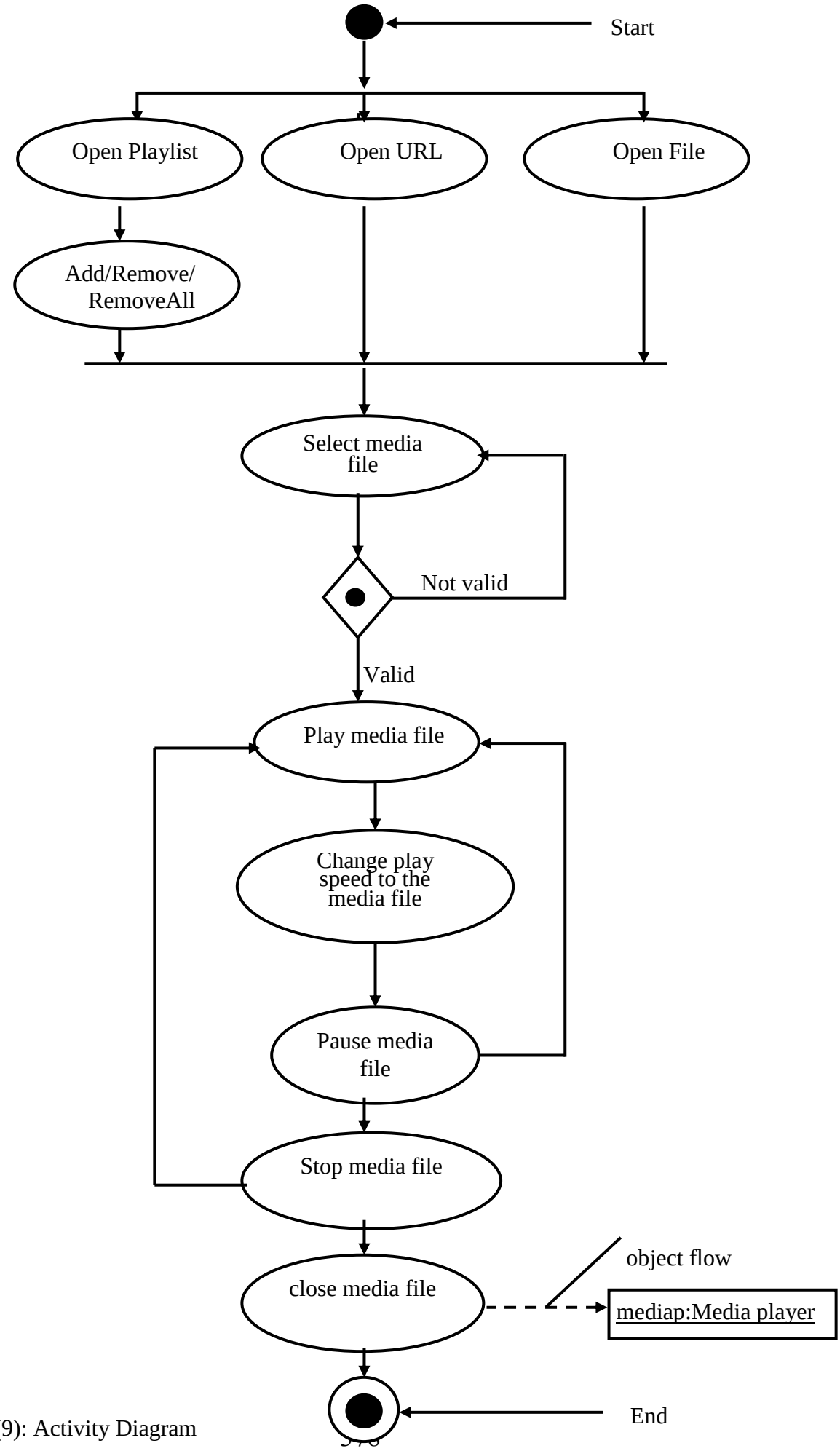


Figure (9): Activity Diagram

4.2 System Implementation

The implementation makes the heavy use of java API for supporting media file the main java package which are used in java media player implementations:

1. javax.swing.[8]
2. java.awt.[8]
3. java.awt.image.BufferedImage.[8]
4. javax.imageio.ImageIO.[8]
5. java.awt.event.[8]
6. java.io.[8]
7. javax.media.[2]
8. java.net.[8]
9. javax.swing.ListSelectionModel.[8]

The most media functions and events are programmed with the help of [8,9].

In this software implementation we try our best to make the GUI (look and feel) as much as most popular media players which makes the interaction of the user with the system very soft and easy that is why the system is very user friendly.

The general interface of the system illustrates in the figure (10), and most of menu functionalities are listed below along with their figures.

Figure (11) represented the content of the file menu.

File menu which have:

- **Open**:- to open media files (audio , vedio), this menu dsiplay a **JFileChooser** to user which has filter to chocie only media files.
- **OpenURL**:- to play online stream media files from the Internet so we can intergrates the system with any browser as plug in. Figure(12) showing this facility.
- **OpenPlayLis**t:- to open multimedia files form local machine the list has the faciliy of add any numbers of media files(audio or vedio) even you can remove one or group of file from the list as show in figuer(13).
- **Close**:- to close the already playing media file without terminating the media player .
- **Exit**:- to terminate the media player (process exit).

Figure (14) represent the view menu interface and its content.

- **FullMod**:-to make full media player GUI screen means display the media player with its menu and buttons as full screen.
- **NormalMod**:-to return to the normal size media player GUI.
- **ZoomTo**:- changes video screen size with different scale as show in figure(15).

Figure (16) represent the **Player** menu interface and its content.

- **Play/Pause**:- play or pause the player.
- **Back**:- start from the beginning of the player.
- **Stop**:- stop the player.
- **Repeat**:- repeat mode of the player when it finished.
- **Next**:- play the next player from the list of files which are already selected by the user in **PlayList** in Figure (13).
- **Previous**:- play the previous player from **PlayList**.
- **Play Speed**:- change the speed of the player with different speed as show in the figure(17).

Most these functions are available in the buttons below the media player so the user can select either one (from **Player** menu or from the buttons below).

Figure (18) represent the Help menu interface and its content

- **About Java Media Player**.- explain the content of the Software, along with user manual.
- **About As**:- explain about the author.

The Software support the pop up menu when the user right click on the player screen as shown in the figure (19) which has the most menu items given above.
The Software also has all the error validations as the system catches all the error situations with proper exception handling classes so the user has no way to play any file type other than media files and when no media files are select to play the alert message box will be displayer as shown in the figure (20).

Figure (10):General interface of the project.

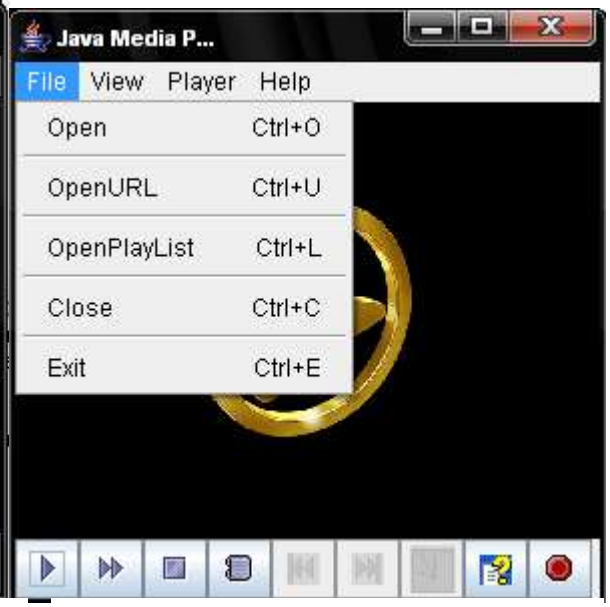


Figure (11): File Menu

Figure (12):OpenURL

Figure (13):OpenPlayList

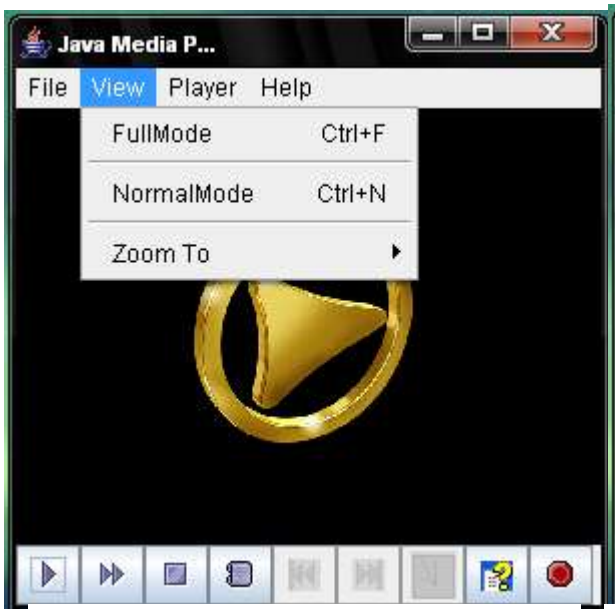


Figure (14):View Menu

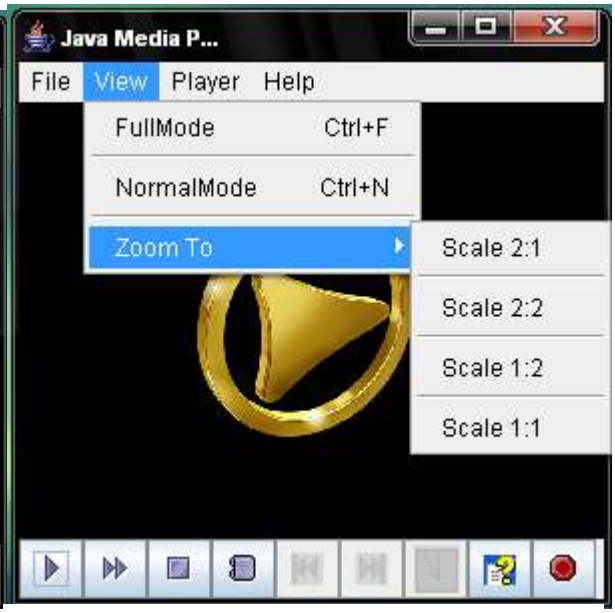
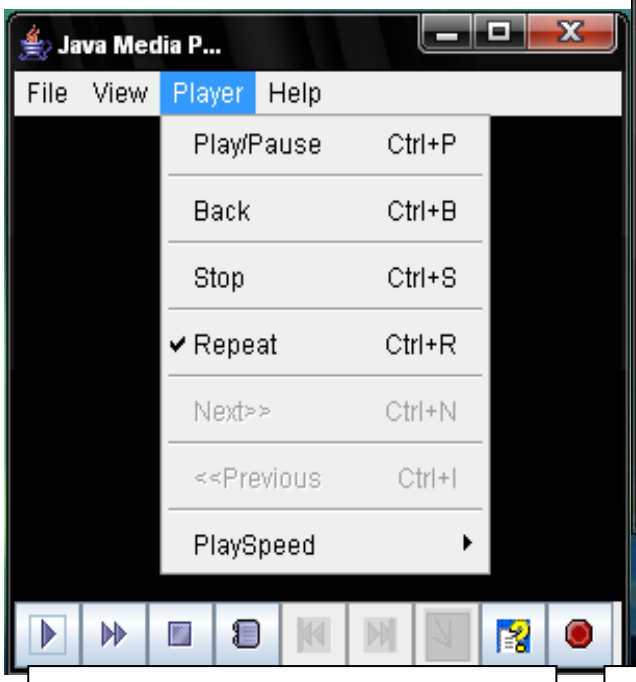
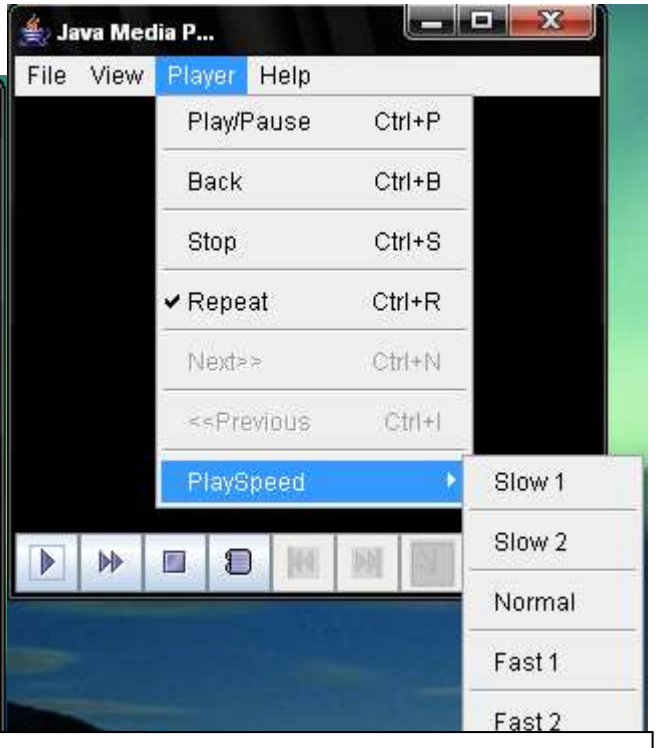


Figure (15):Zoom To



Figure(16):Player Menu



Figure(17):Play Speed

Figure(18):Help Menu

Figure(19):Pop up menu

Figure(20):Alert message box

5. Conclusion

The media player application provides following features: -

A) To Single Platform Users: -

1. Playing various media files format.
2. Changing speed of playing media file.
3. Create your own Playlist.
4. View elapsed time of playing media file.
5. Playing real time media files from internet.

B) To Multiple Platform Users: -

1. Easy access to GUI of application.
2. Support different files format.
3. Different plug-in for operating system platform like Encoder and Decoder.

6. Feature Enhancement

The media player application needs the following feature enhancement:

1- Capturing the image from playing video file.
2- Changing format of media file.
3- Supports for compressing media file.
4- Playing file backwards with multiple speeds.
5- Live broadcast over Internet
6- Zooming in and out video file.
7- Supports for capturing media device.
8- Cutting the media files into small sub media files.
9- Integration with some web browser as plug in.

7. References

1. http://www. Wikipedia .com\Windows Media Player, the free encyclopedia.mht
2. http://java.sun.com/products/java-media/jmf2.1.1, (2010).
3. http://www.ki-multimedia.com/sample.html, (2010).
4. Herbert Schildt; "The JAVA 2 Complete Reference", Fifth Edition McGraw Hill/Osborne, (2002).
5. http://java.sun.com/products/java-media/jmf,(2010).
6. Dennis de Champeaux Douglea, and Penelope Faure,"Object-Oriented System Development**",** (1993).
7. Grady Booch, James Rumbaugh and Ivar Jacobson;" The Unified Modeling Language User Guide",Tata India,(2000).
8. http://java.sun.com/j2se/1.5.0/docs,(2010).
9. Mike Cohn, Bryan Morgan, Michael Morrison, Michael T. Nygard, Dan Joshi, and TomTrinko;" JAVA Developers Reference", (1996).