

Rajob raihan Monmoy

University of Greenwich

[Date]



Table of Contents

[1 COMP1148 Coursework Feedback sheet 3](#_Toc476222528)

[2 Introduction 5](#_Toc476222529)

[3 Design and Development 6](#_Toc476222530)

[4 Testing and Faults 7](#_Toc476222531)

[5 Conclusions, further development and reflection 8](#_Toc476222532)

[5.1 Future innovations 8](#_Toc476222533)

[5.2 Reflection 8](#_Toc476222534)

[6 Appendix A – The commented code for deliverable A 9](#_Toc476222535)

[7 Appendix B – White Box Testing 10](#_Toc476222536)

# COMP1148 Coursework Feedback sheet

Name: Overall mark: %

There are **no specific marks for each stage** and your overall mark is based on **your** **performance as a whole**. The following table indicates which Grading Band your coursework falls into and why. You may receive marks in a higher band if your tutor feels some of the work justifies it.

Remember: You cannot get marks for a particular stage unless you have made a reasonable attempt at all of the previous stages. For example, if you miss out the testing (stage 4) you cannot get a mark over 60%, regardless of how good your innovations are.

|  |  |  |
| --- | --- | --- |
| **Grading Band** | **Indicative Grading Criteria** | **Your work** |
| **71-100%** | a very good attempt at all stages, with an excellent report |  |
| **61-70%** | a good attempt at all stages, with a good report |  |
| a very good attempt at all stages, but the report is weak |  |
| a very good attempt at all stages, but the program is hard to use |  |
| a very good attempt at all stages, but the code contains minor runtime errors |  |
| **41-60%** | a good attempt at stages 1-4, with a good report |  |
| a reasonable attempt at all stages, but the report is missing a section |  |
| a reasonable attempt at all stages, but the code contains serious runtime errors |  |
| **21-40%** | an attempt at stages 1-3, with a reasonable report |  |
| a reasonable attempt at further stages, but the report is missing several sections |  |
| a reasonable attempt at further stages, but the code cannot be run by your tutor |  |
| **11-20%** | an attempt at stages 1-2 |  |
| **0-10%** | an attempt at stage 1 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Assessment Criteria** | Achieved well | Partially achieved | Not achieved |
| **The functionality of the programme.** |  | | |
| Does the system do what it is supposed to do? |  |  |  |
| **Usability:** |  | | |
| Is your system straightforward and easy to use? |  |  |  |
| Is it obvious to the user what to do? |  |  |  |
| Are all messages clear and unambiguous? |  |  |  |
| Is the output formatted appropriately? |  |  |  |
| Is bad input data handled appropriately? |  |  |  |
| Is the system free from crashes and uncaught exceptions? |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Quality of the Java code:** | |  | | |
| Inclusion of meaningful comments (other than for Stage 1). | |  |  |  |
| Use of sensible naming standards. | |  |  |  |
| Clear code layout and formatting. | |  |  |  |
| **Quality and completeness of the report:** | |  | | |
| Is the design documentation clear and concise? | |  |  |  |
| Have you included evidence of appropriate testing? | |  |  |  |
| Have you discussed any faults or failures? | |  |  |  |
| Have you reflected on the development process? | |  |  |  |
| **Feedback Comments** | | | | |
| **Interim Deliverables** | (Brief feedback on your work as the term progressed) | | | |
| **Final Deliverable** | (Feedback on your work as a whole – typically: a summary; an evaluation of your report; a suggestion of what could be improved in the program; and an indication of the best features of your coursework) | | | |

# Introduction

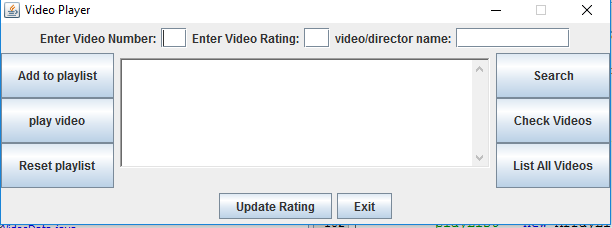
The program should be a simulation of a virtual video player with no video playing functionality. Here is a list of things the program should achieve:

1. User can search video by video number.
2. User should be able to see relevant information about the video like movie name, director name, rating and how many times the video is played.
3. User should be able to see the list of all videos.
4. User should be able to make playlist by inputting video number.
5. Since this is a simulation no videos will be played, but play count for each video in the playlist will be incremented by one if user press play video button.
6. User should be able to reset the playlist.
7. User should be able to give rating to a particular video by inputting video name and then the rating.

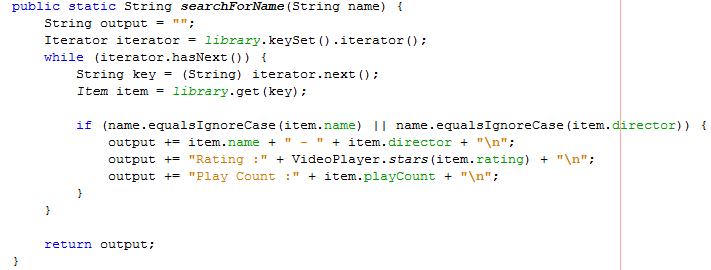
Also the system should be straightforward and easy to use. All the messages should be unambiguous. The bad input data should be handled properly. The system should be free from crashes.

# Design and Development

To develop the final application with applying innovation at first the whole functionality is brought into a single GUI. By applying borderlayout and Gridlayout the main GUI is formed.

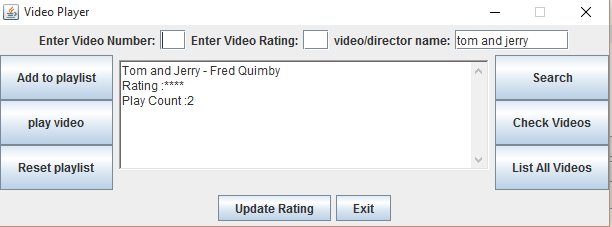


Extra feature such as searching option is also provided here. User has to input the video name or the director’s name to find the desired result. A loop is created which will check the existing library if there is any match with video name or director name. The desired result will be printed into text area if any found. Otherwise it will show an error message.

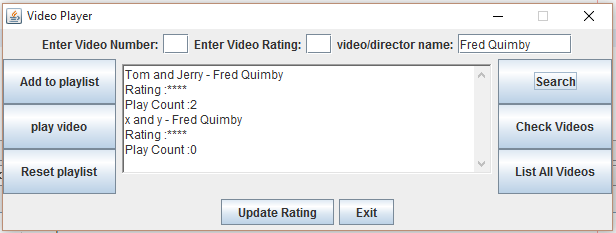


To check a video only video number should be inputted and a search operation will be accomplished with a loop. Output will be printed on textarea.

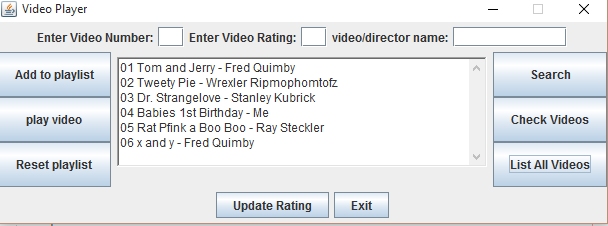
User can only search a video by movie name.



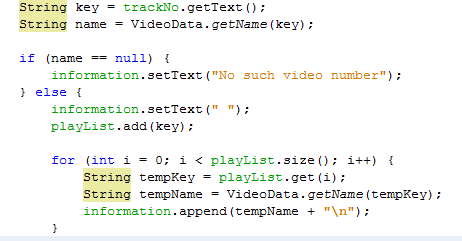
User can also see all the movies by the same director by searching the director name.

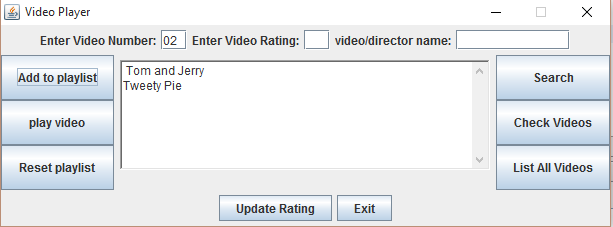


**List All Videos**: function is done with a straightforward method. A loop is created and brought and printed all data stored in the library.



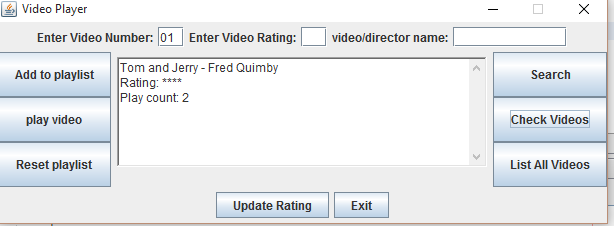
**Add to playlist**: video number will be checked and if found it will be stored into an arraylist and printed on textarea. If any wrong track number then textarea is will be cleared an error message will be generated. And again if any valid number is inputted then previous playlist result and will be shown along with new inputted video. It is possible because every playlist information is stored into the arraylist.



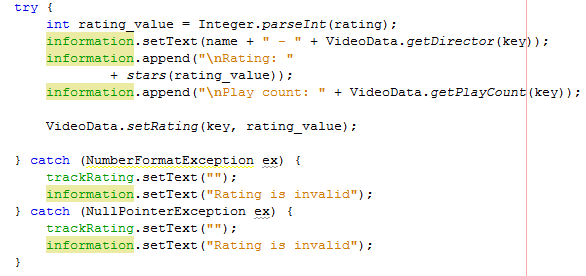


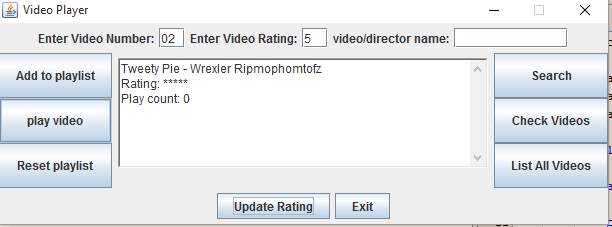
**Reset**: Reset button will simply clear the text area and also clear the arraylist. Arraylist’s built in clear() method is used in this purpose.

**PlayVideo**: The data from arraylist will be taken and with the help of VideoData.java’s public static method incrementPlayCount(), play count will be increased for each video. The incrementPlayCount() method also a user defined method.



**UpdateRating**: video number and rating is checked with if-else condition if it is valid or not. If valid number is provided then VideoData.java’s public static method setRating() is used to update the rating. The rating field is converted into integer form to show the”\*” instead of simple number show. This process is handled between try-catch block to prevent any unwanted runtime crashes.





These are the procedures of designing and developing my final code for this application.

# Testing and Faults

White box testing is testing of a systems internal coding and infrastructure. This testing is based on inner working of an application and revolves around internal testing. White box testing finds internal security holes and tests each statement. It matches desired output with obtained output and finds bug. The testing method used for this project is statement coverage techniques. The actual table and results is listed in appendix B.

Faults and failures:

When a invalid number like “asdf” in place of an actual number is typed the program also searches the file in the database. But the song number cannot be a string. So it needs a validation before checking the static video database if the number entered is valid or not. I fixed the problem after white box testing.

In UpdateVideos.java there was a runtime exception. The code was failed, when user input a random sting which is not actually an integer in “track rating” field. Now I fixed that portion of code and now it works correctly.

# Conclusions, further development and reflection

## Conclusions

This is a very basic video player simulation. User can search video from the static video database and see all the relevant information about the video like director name , movie name , play count , rating. User can also add playlist. Video play count is increased when play video button clicked. User can see how many times the video is played. User can also give the video rating according to their choice. The rating and play count will be saved and can be seen from checking video.

## Future innovations

This is a very basic video player simulation with very few functionality. If I had another three months to work on the program I would like to add video playing functionality. This would make it a complete video player. Also I would like to enhance the playlist facility like allowing playlist to be saved externally and modify later.

## Reflection

**Option A**

**What did I actually achieve with this element of learning?**

I have learned how to code with Java and the basic functionalities of object oriented programming is very much clear to me. How to manipulate data from one class to another class and how actually OOP works is also learned.

**Which were the most difficult parts, and why were they difficult for me?**

Most difficulty part of this project for me was to merge all functionalities into one single GUI. It was difficult because at first the idea to give the application a proper GUI was hard to find out. After giving the GUI to merge the code into one place was also seemed difficult because the length of the code in a single class was getting larger.

**Which were the most straightforward parts, and why did I find these easy?**

Most straightforward part was creating the required two classes.

# Appendix A – The commented code for deliverable A

**Code**: CheckVideo.java

package coursework;

// import the classes of the abstract windows toolkit and

// swing to enable the use of components such as buttons

// and import event to make use of events such as actionPerformed event

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

//Extending Jframe class for using the functionality of the class and

//implementing ActionListener interface

public class CheckVideos extends JFrame implements ActionListener {

//initializing components for the GUI

JTextField trackNo = new JTextField(2);

TextArea information = new TextArea(6, 50);

JButton list = new JButton("List All Videos");

JButton check = new JButton("Check Video");

public CheckVideos() {

//set the layout to border layout

setLayout(new BorderLayout());

//set the jframe size

setBounds(100, 100, 400, 200);

//set the name for the jframe

setTitle("Check Videos");

//This is for disposing jframe by clicking the X button

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

//Create a Jpanel and add the components for top bar

JPanel top = new JPanel();

top.add(new JLabel("Enter Video Number:"));

top.add(trackNo);

top.add(check);

top.add(list);

//adding action listener to buttons

list.addActionListener(this);

check.addActionListener(this);

//adding panel to the north

add("North", top);

//create a panel and then add previously initialized textarea

//information. Then add the panel to the middle of jframe

JPanel middle = new JPanel();

information.setText(VideoData.listAll());

middle.add(information);

add("Center", middle);

//This is used so that user can not resize the jframe

setResizable(false);

//This makes the jframe visible

setVisible(true);

}

//this function is invoked when list or check button is clicked

public void actionPerformed(ActionEvent e) {

//if the clicked button is list button then call lisAll function

//from VideoData class

if (e.getSource() == list) {

information.setText(VideoData.listAll());

}

//if the clicked button is check button

else {

//take the data from trackNo textfield

String key = trackNo.getText();

//get the name from VideoData class using getName(Key) class

String name = VideoData.getName(key);

//getName(Key) class in VideoData returns null if no name found

if (name == null) {

information.setText("No such video number");

} else {

//if name found fill up the information textfield using

//the data gathered from VideoData class

information.setText(name + " - " + VideoData.getDirector(key));

//passes the rating data found from VideoData to the stars function

information.append("\nRating: "

+ stars(VideoData.getRating(key)));

information.append("\nPlay count: " + VideoData.getPlayCount(key));

}

}

}

//Produces stars for each rating. Ex: 5 rating then \*\*\*\*\* will be produced

private String stars(int rating) {

String stars = "";

for (int i = 0; i < rating; ++i) {

stars += "\*";

}

return stars;

}

}

# Appendix B – White Box Testing

Tested the code in stage 3.

**Class**: CheckVideos.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | Clicked | Method | Output Expected | Output Obtained |
| 01 | check | actionPerformed(ActionEvent e) | Tom and Jerry - Fred Quimby  Rating: \*\*\*\*  Play count: 0 | Tom and Jerry - Fred Quimby  Rating: \*\*\*\*  Play count: 0 |
| null | check | actionPerformed(ActionEvent e) | No such video number | No such video number |
| asdf | check | actionPerformed(ActionEvent e) | No such video number | No such video number |
| 1 | check | actionPerformed(ActionEvent e) | No such video number | No such video number |
| null | List | actionPerformed(ActionEvent e) | 01 Tom and Jerry - Fred Quimby  02 Tweety Pie - Wrexler Ripmophomtofz  03 Dr. Strangelove - Stanley Kubrick  04 Babies 1st Birthday - Me  05 Rat Pfink a Boo Boo - Ray Steckler | 01 Tom and Jerry - Fred Quimby  02 Tweety Pie - Wrexler Ripmophomtofz  03 Dr. Strangelove - Stanley Kubrick  04 Babies 1st Birthday - Me  05 Rat Pfink a Boo Boo - Ray Steckler |

**Class**: CreateVideoList.java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | Clicked | Method | Output Expected | Output Obtained |
| 01 | addToPlaylist | actionPerformed(ActionEvent e) | Tom and Jerry | Tom and Jerry |
| 15 | addToPlaylist | actionPerformed(ActionEvent e) | Wrong Track Number! | Wrong Track Number! |
| asdf | addToPlaylist | actionPerformed(ActionEvent e) | Wrong Track Number! | Wrong Track Number! |
| null | addToPlaylist | actionPerformed(ActionEvent e) | Wrong Track Number! | Wrong Track Number! |
| null | play | actionPerformed(ActionEvent e) | playing(check play count in CheckVideos.java is 0) | playing(check play count in CheckVideos.java is 1) |
| null | reset | actionPerformed(ActionEvent e) | null | null |

**Class**: UpdateVideos.java

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Track No | Track Rating | Clicked | Method | Output Expected | Output Obtained |
| null | null | enter | actionPerformed(ActionEvent e) | No such video number | No such video number |
| 15 | null | enter | actionPerformed(ActionEvent e) | No such video number | No such video number |
| 01 | null | enter | actionPerformed(ActionEvent e) | Rating is invalid | Rating is invalid |
| 01 | asad | enter | actionPerformed(ActionEvent e) | Rating is invalid | Rating is invalid |
| 01 | 5 | enter | actionPerformed(ActionEvent e) | Tom and Jerry - Fred Quimby  Rating: \*\*\*\*\*  Play count: 0 | Tom and Jerry - Fred Quimby  Rating: \*\*\*\*\*  Play count: 0 |