Understand, Develop and Evaluate a Java Product

Gheorghe Mitrea 1118495

Programming in Java

12/20/16

Contents

Table of Figures

Introduction

This assignment has been completed in order to understand the concepts of Java and its basic fundamentals. All the answers have been given after careful study of assignment brief.

1. Task 1 Explain the characteristics and features of programming in Java

a) Why Java?

Java is a high-level object-oriented programming language. It is a general-purpose programming language created by Sun Microsystems. For the most part, Java is used to develop web applications but also, can create simple mobile applications to programs for huge data centres. The main features of the language include:

- ✓ ***Simplicity*** - eliminates the overloaded operators, the multiple inheritance and all the facilities that can cause confusing code
- ✓ ***Robustness*** - eliminates sources of common programming errors that occur by eliminating pointers, it is managing the automatic memory, and it is eliminating the memory cracks, through a procedure of automatic release memory those objects that are no longer used. Java is binding functions at runtime and compile information available to the runtime application
- ✓ ***Ease of use*** – as regard of the creation of complex applications using network programming, execution threads, graphical interface, databases, etc.
- ✓ ***Security*** - is the most secure programming language available at this time, ensuring strict mechanisms for the security of the programs resulting in checking the dynamics of the

code for the detection of dangerous sequences, imposing certain strict rules for running the programs launched on the computers located at a distance;

- ✓ ***Portable***, in other words, Java is a programming language independent of the platform on which is running, the same application can run without any change on different systems such as Windows, UNIX or Macintosh, which result in substantial savings for firms who develop applications for the Internet.
- ✓ ***Object oriented*** - completely eliminates the procedural programming style.
- ✓ ***Performance -*** although slower than programming languages that generate native executables for a particular work platform, the Java compiler ensures high performance of the bytecode, so a slightly lower work speed will not be an impediment to the development of complex applications, including 3D graphics, animated, etc.
- ✓ ***Architectural Neutrality -*** The behaviour of a Java application does not depend on the physical architecture of the machine on which it is running.
- ✓ ***Distributed computing*** - Java applications can easily access network, using the calls to a standard set of classes (java.net) and distributed execution (java.rmi);
- ✓ ***Dynamic*** - objects are always dynamically allocated. Java class libraries can be reused very easily.

b) General Characteristics:

Currently, Java is more than just a programming language. It is a collection technology that allows developing and running applications secure, portable and scalable. Java programming language has some general features:

- ✓ ***Java Virtual Machine (JVM)*** - Java programs are compiled and then interpreted. The reason for this system is the introduction of the flow of execution of an additional layer called the Java Virtual Machine. As the name implies, is a processor JVM, but not real, physical, conducted on silicon, but a virtual processor, simulating the host processor. In essence, the virtual machine is another program;

- ✓ ***Fully object-oriented*** - completely remove the style of procedural programming; it is based on the encapsulation, inheritance and polymorphism;
- ✓ ***Compiled and Interpreted*** - Java is a compiled and interpreted programming language. Depending on the mode of execution of the programs, programming language is divided into two categories:
    - ▪ **interpreted:** instructions are read line by line by a program called interpreter and translated in the instructions by the machine;
        - *advantage*: simplicity;
        - *disadvantage*: speed of execution reduced;
    - ▪ **compiled**: the compiler transforms the source code of the programs into a code that can be made directly to the processor;
        - *advantage*: execution;
        - *disadvantage*: no portability, compiled code at a low level cannot be run only on the platform on which has been compiled.
- ✓ ***Applications vs. Applets*** - In straightforward terms, ***applets*** are simply Java programs that run in web browsers, while ***an application*** runs solitary, with the support of a virtual machine.

Practical use of applets

- Applets have serious uses on intranets. Reasons:
    - Corporate Intranets are safe and managed -> many of the normal security restrictions can be relaxed.
    - Ease of administration - a large number of very small applications distributed over many machines can create a management problem (i.e. periodic updating) - less able users can move, damage, or delete them. Central server applets that are loaded on demand do not have this problem (although browsers, plug-ins, etc. need to be maintained).

- The wide bandwidth. Internal networks generally have large bandwidths - this means that even reloading of large applets may not be a problem.

- It is still a viable technology for applications that need a more robust interface than a JavaScript page. Also, Java WebStart - the new alternative allows applications to be downloaded through the browser and running their independent browser, using JVM from the system respectively.

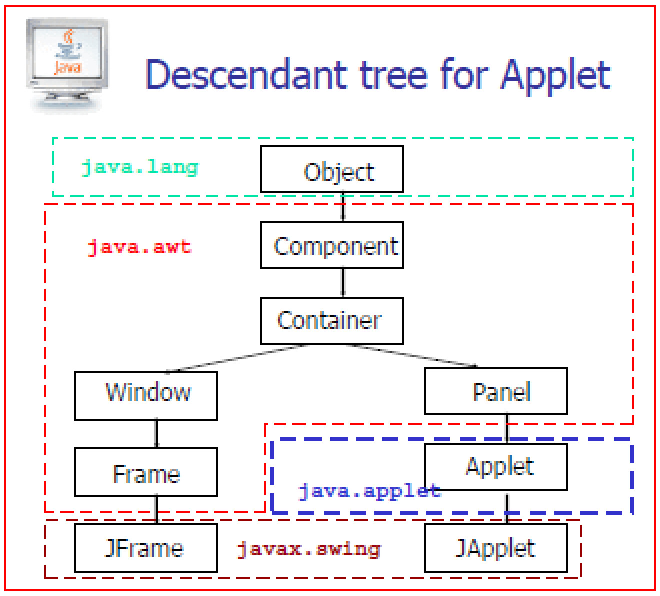


Figure 1 Descendent tree of a Java applet

The limitations of applets

- It cannot run any client executable:
- It cannot communicate with any other host except the server they come from
- It cannot read or write on the client file system
- It can only get limited information about the client machine

Essential differences between applets and stand-alone applications:

- An instance of an applet is always created, and its constructor *init* and the *start* method are always executed.
- Because an Applet instance is a Panel instance (*JApplet* extends the Applet), a visible component is created, *awt* events are treated.
- When invoking a standalone application, it is executed only *public static void main (String []) (and the code called by it)*.

c) Data Types and Variables

In Java, data types are divided into two categories: primitive data types and reference types. Java starts from the premise that everything is an object. So, data types should actually be defined by classes and all variables should actually store instances of these classes.

The most common primitive data types are:

a) Arithmetic

Integers:

- *byte (1 octet) -* Byte data type is an integer with values between -128 and 127 (inclusive). Byte data type can be useful to save memory in large vectors.

- *short (2 octets) -* Short data type is an integer with values between 32 768 and 32 767 (inclusive). As with byte, you can use a short to save memory in large arrays, in cases where the memory savings matter.

- *int (4 octets) -* Int data type is an integer in the range of (-2.147.483.648, 2,147,483,647]. For integer values, this data type is the standard choice.

- *long (8 octets)* - Long data type is an integer between 9,223,372,036,854,775,808 and 9,223,372,036,854,775,807. This type of data using a range of values wider than that provided by int.

Real data types:

- *float (4 octets)* - Data type float represents rational numbers in floating point, simple precision, 32-bit. This type of data should not be used for precise values, such as exchange rate or conversion rate between currencies. For this, use the java.math.BigDecimal class.

- *double (8 octets)* - double datatype is a floating-point representation, simple precision, 64-bit. This type of data is the standard choice for rational values. This type of data should not be used to exact values.

b) Logic

- *Boolean (true and false)* - Boolean data type has only two possible values: true (true) and false (false). Using this type of data for indicators that track conditions true/false. This type of data is a bit of information, but its size is not a specifically defined.

c) Character

- *char* data type - is a Unicode character. It has a minimum value of '\ u0000' (or 0) and a maximum value '\ uffff' (or 65,535 inclusive).

1. The most common Reference data types

Vectors, classes, and interfaces are reference types. The value of a variable of this type is in contrast to the primitive types, a reference (memory address) to the value or set of values represented by that variable.

String - A string is a sequence of characters between two apostrophes and can contain any characters: uppercase and lowercase letters, special characters (‘#', '&', etc.) and delimiters (comma, period, etc.). A constant string is a symbol consecutiveness inserted between apostrophes.

Operators and functions on strings

The operator + is used for concatenation (joining) several rows.

For example, 'Ana' + 'Mary' will return the sequence 'Ana Maria '.

Operators =,>, <,>, <, are strings and are used for comparison.

Variables

Variables are used in the programming to store data of a program. These are created in primary memory of the system. When a class is defined, under its definition some instance variable is declared that will be created with the object. Every object will have its own copy of each of its instance variable.

When we want to use a variable, we must declare it first. While declaring a variable, we have to specify its data type and name. We cannot use a variable without declaration. It will cause a compile time error in our program. There are several data types available in Java.

The variables can have as the type or a primitive type date, either a reference to an object.

Variable Declaration

Any variable must be declared to be able to be used. This declaration must contain a type of values that can be stored in the location of the reserved variable and a name for the variable declared. On the basis of the place in the source of the program in which the variable has been declared, this receives a class of local or static memory.

The ***declaration*** of the variables is made by:

type_the_data name variable;

Variable Initialization

Java allows initialization of variables' values even when they are declared.

Variables ***initialization*** is made by:

name_variable = value;

Variable Name Convention

The variable name can be any Java identifier. Any variable that is not final has a name starting with lowercase letter while the final variables have names that contain all capital letters. If the variable name is not final contains more words, words from the second are written in lower case letters but with the first letter capitalized. Examples of variable names which are not final could be:

numberOfSteps color nextElement

Final variables could have names like:

ORANGE GREEN LIGHT_BLUE

The Operators

1. ***Assignment operator*** = (equal sign)

Example*: a = 9 (a is assigned the value 9)*

2. ***Binary arithmetic operators***: +, -, *, /, %

Example: *s = a + b*

In Java, there are short forms that comprise the assignment operator and a binary arithmetic operator. Are abbreviated operators: + = - =, + =, / =,% =

3. ***Unary arithmetic operators***: +, -, ++ (increment operator), - (decrement operator). The increment and decrement can be prefixed (++ x or --x) or postfix (x ++ or x--).

a) -x is the opposite of x

b) int x = 5, y = 7;

x ++; // X receives value 6

y--; // Y gets value 6

Constants and Literals

A static final variable is effectively a constant. The static modifier causes the variable to be available without loading an instance of the class where it is defined. The final modifier causes the variable to be unchangeable. The constants include the following types:

1. ***Constant Integer***

Three bases numeracy are supported: base 10 base 16 (starting with the characters 0x) and base 8 (starting with 0) and can be of two types:

• normal (it is 4 bytes - 32 bit)

• Long (8 bytes - 64 bits) ends with the character L (or him).

2. ***Real constants***

For a constant to be considered true it must have at least one decimal place after the decimal point to be in exponential notation or have the suffix F or f to normal values (represented by 32 bits) or D or d values long (represented 64 bit).

3. ***Logical constant***

True: Boolean value of true

False: Boolean value false

4. ***Constant Character***

A constant character is used to express the Unicode character code. The representation is done using either a letter or an escape sequence written between apostrophes

5. ***Constant Strings***

A string consists of zero or more characters enclosed in double quotes. Characters that can be formed string graphic characters or escape sequences. If the string is too long it can be written as a concatenation of substrings smaller size. String concatenation operator is + ("Ana" + "is" + "apples"). Null string is "". As we shall see, any string is actually an instance of String, defined in the java.lang package.

The lexical structure of Java

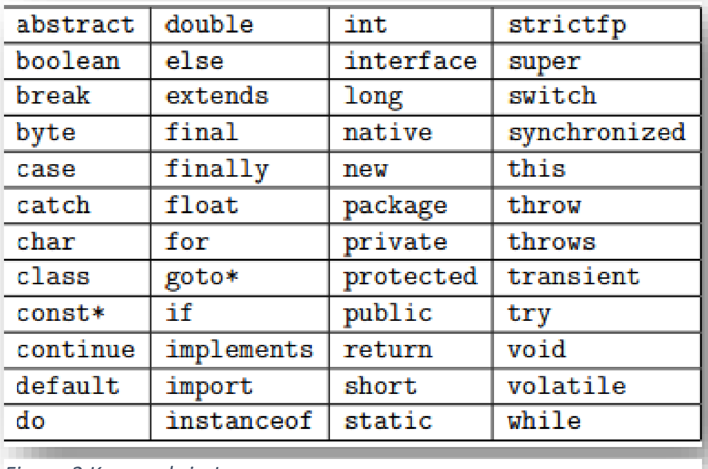
Lexical units in Java are:

- Keywords
- Identifiers
- Literals
- Separators
- Operators

Identifiers

Java Identifiers are unlimited sequences of letters and numbers Unicode, starting with a letter. Identifiers are not allowed to be identical with the words reserved.

Interfaces, variables, methods, true, false, null are not keywords, but they are not used in applications with their name. Words marked with * are reserved but not used:



abstract	double	int	strictfp
boolean	else	interface	super
break	extends	long	switch
byte	final	native	synchronized
case	finally	new	this
catch	float	package	throw
char	for	private	throws
class	goto*	protected	transient
const*	if	public	try
continue	implements	return	void
default	import	short	volatile
do	instanceof	static	while

Figure 2 Keywords in Java

A literal is a basic mode of expression in the source file the values which they may take the primitive types and the type of string of characters. Using literals can enter the constant values in the variables of primitive type or the variables of string type of characters.

In the language of the Java there are the following types of literal:

• Integer literal

• Flotation literal

• Boolean literal

• Literal character

• Literal string of characters

d) *Flow Control*

Java for execution control instructions can be divided into the following categories:

• Instructions decision: if-else, switch-case

• Instructions jump: for, while, do-while

• Instructions for exception handling: try-catch-finally, throw

• Other instructions: break, continue, return, label:

1. ***Boolean Operators***

The Boolean logical operators are: ***, & , ^ , ! , , && , == , !=*** . Java supplies a primitive data type called Boolean, instances of which can take the value ***true*** or **false** only, and have the default value ***false*.**

2. ***Flow control instructions*** can be divided into the following categories:

• Decision Instructions: ***if-else, switch-case***

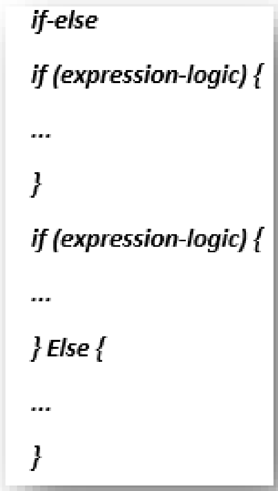
• Loops Instructions: ***for, while, do-while***

• Instructions for handling the exception: ***try-catch-finally, throw***

• Other instructions: ***break, continue, return, label***.

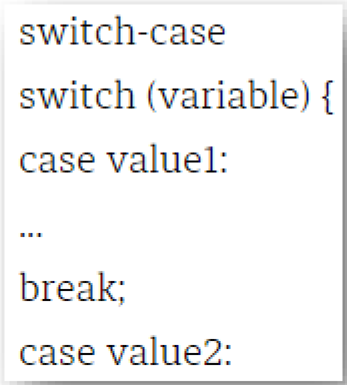
a) Instructions decision

Examples:



```
if-else
if (expression-logic) {
...
}
if (expression-logic) {
...
} Else {
...
}
```

Figure 3 If and else statements



```
switch-case
switch (variable) {
case value1:
...
break;
case value2:
```

Figure 4 switch case statements

b) Loops Instructions

Examples:

```
for
for (initialization; logical-expresion; step-iteration) {
// Loop body
}
for (int i = 0, j = 100; i <100 && j> 0; i ++, j--) {
...
}
```

*Figure 5 **for** statement*

Both initialization and in iteration can be more instructive separated by commas.

```
while
while (logical-expression) {
...
}
do-while
do {
...
}
while (logical-expression);
```

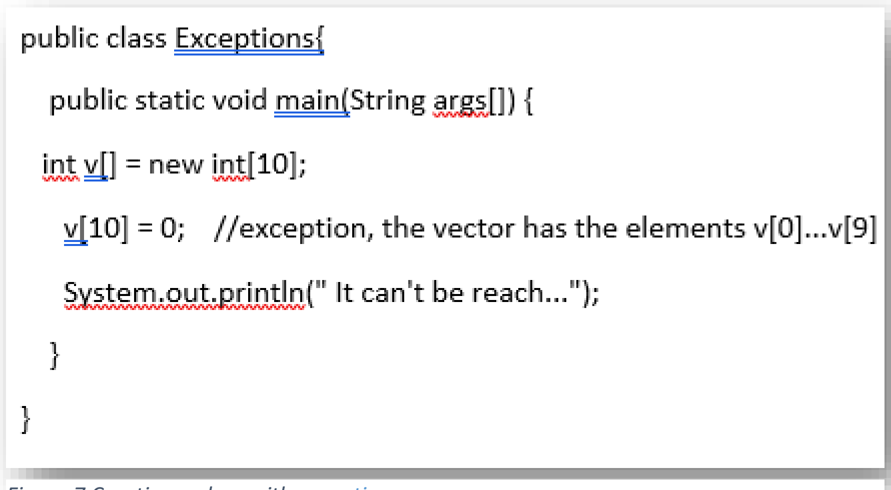
Figure 6 while, do-while statements

3. Exceptions and Errors

An exception is an event that occurs during the execution of a program and causing the interruption of the normal execution. When such an error occurs during a program execution system automatically generates an exception object that contains:

- information about the exemption;
- program status at the time of the exception

Example:



```
public class Exceptions{
    public static void main(String args[]) {
   int v[] = new int[10];
     v[10] = 0;    //exception, the vector has the elements v[0]...v[9]
     System.out.println(" It can't be reach...");
    }
}
```

Figure 7 Creating a class with exceptions

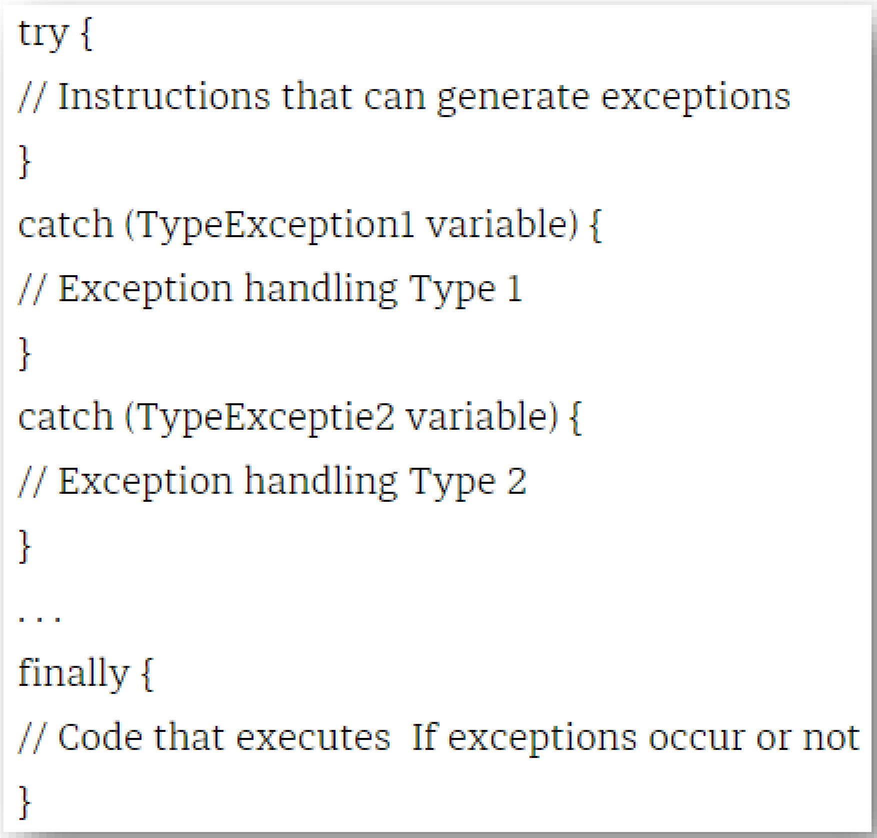
When the program is running will generated an exception and will display the next message:

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 3

at Exceptions.main (Exceptions.java:4)

Creating an exception object is called throwing an exception. When the method throws an exception, execution system is responsible for finding a sequence of code of a method to treat it.

Exception handling is done by blocks of instructions ***try, catch and finally***. A sequence of code that treats certain exceptions. It should look like this:



```
try {
// Instructions that can generate exceptions
}
catch (TypeException1 variable) {
// Exception handling Type 1
}
catch (TypeExceptie2 variable) {
// Exception handling Type 2
}
. . .
finally {
// Code that executes  If exceptions occur or not
}
```

Figure 8 Treat the exceptions with try, catch and finally

e) Data Structures

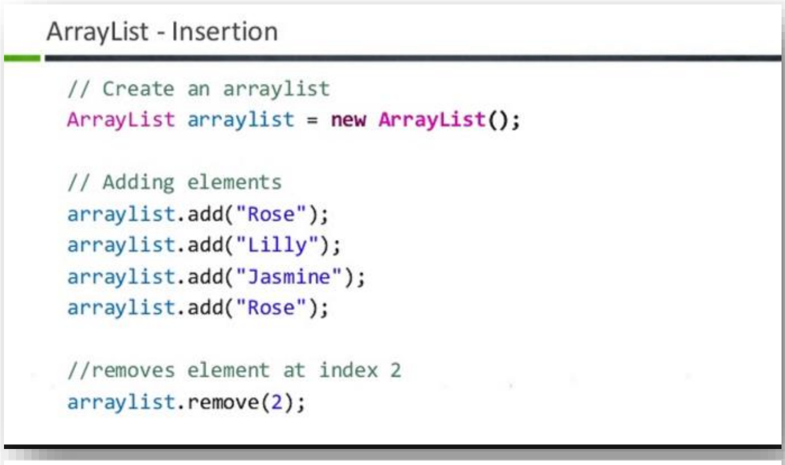
The data structure is a systematic method of storing information and data in a computer so that they can be used effectively.

A data structure is, therefore, a collection of data of different types, on which has defined a specific organization and which is specific to a particular way of identification of the components.

Data structures can be created to be stored in the internal memory (these data structures are called internal structures) or in the external memory (are called external structures, or ***files***). The internal structures have a character of the temporary data (they disappear once the activity of processing has stopped) and these files are permanent data (on a long time long time).

Elementary data structures:

- ***Lists*** - A list is a collection of information elements (nodes) arranged in a certain


ArrayList - Insertion

```
// Create an arraylist
ArrayList arraylist = new ArrayList();

// Adding elements
arraylist.add("Rose");
arraylist.add("Lilly");
arraylist.add("Jasmine");
arraylist.add("Rose");

//removes element at index 2
arraylist.remove(2);
```

Figure 9 Example of a Array List of nodes

order.

- ***Stacks and Queues*** - A stack is a linear list of property that the operations of insertion/extraction nodes are to/from bottom of the list.
- ***Graphs***
- ***Binary Tree***
- ***Heaps***

f) Standards of formatting source code

Formatting Source Code includes:

- Indentation
- Placement of braces
- Whitespace
- Blank lines
- New lines
- Control
- Line wrapping statements
- Comments

1. Use of Comments

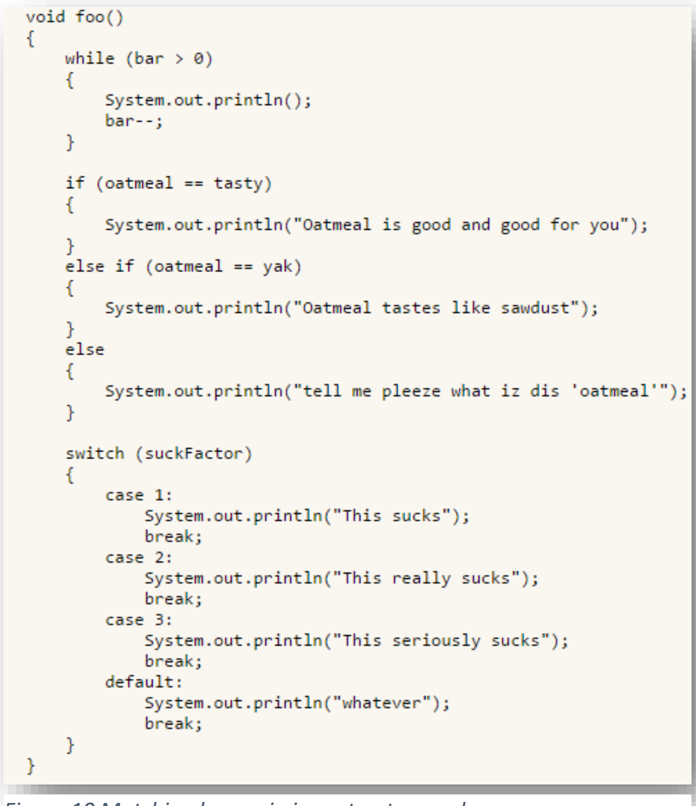
In Java, there are three types of comments:

- Comments on a single line: I begin with //.
- Comments on multiple lines closed between /* and */.
- Comments on more lines forming the documentation closed between /** and **/. The text of the two sequences is automatically moved to the documentation of the application by the Javadoc generator automatically.

2. ***Indentation***

- Indentation follows the structure of the program,
- All indenting is made with spaces, not with tabs,
- Matching braces always line up vertically in the same column as their construct.

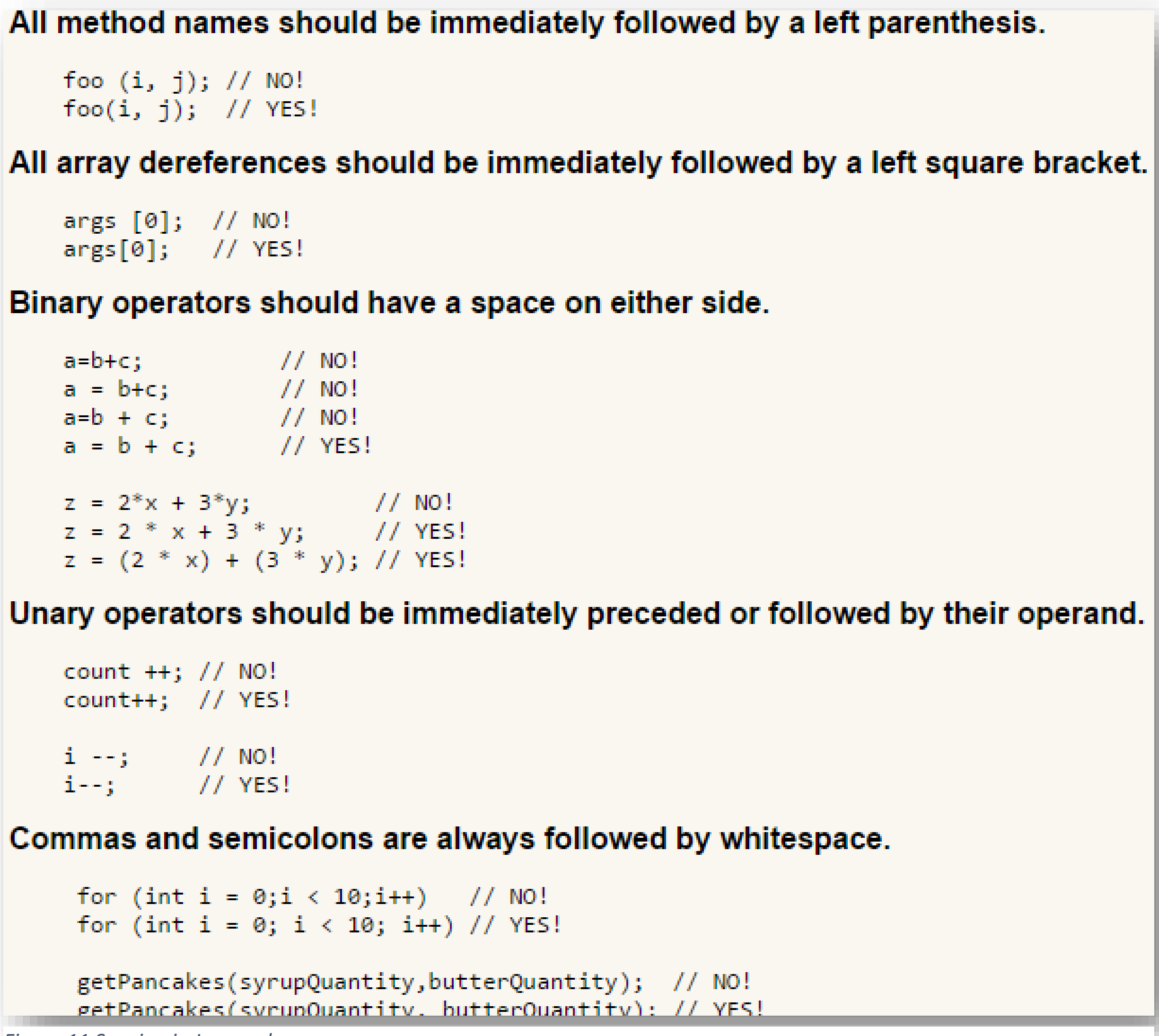
Example:



```
void foo()
{
    while (bar > 0)
    {
        System.out.println();
        bar--;
    }

    if (oatmeal == tasty)
    {
        System.out.println("Oatmeal is good and good for you");
    }
    else if (oatmeal == yak)
    {
        System.out.println("Oatmeal tastes like sawdust");
    }
    else
    {
        System.out.println("tell me pleeze what iz dis 'oatmeal'");
    }

    switch (suckFactor)
    {
        case 1:
            System.out.println("This sucks");
            break;
        case 2:
            System.out.println("This really sucks");
            break;
        case 3:
            System.out.println("This seriously sucks");
            break;
        default:
            System.out.println("whatever");
            break;
    }
}
```

Figure 10 Matching braces in java structure code

3. ***Spacing***



All method names should be immediately followed by a left parenthesis.

```
    foo (i, j); // NO!
    foo(i, j);  // YES!
```

All array dereferences should be immediately followed by a left square bracket.

```
    args [0];  // NO!
    args[0];   // YES!
```

Binary operators should have a space on either side.

```
    a=b+c;          // NO!
    a = b+c;        // NO!
    a=b + c;        // NO!
    a = b + c;      // YES!

    z = 2*x + 3*y;         // NO!
    z = 2 * x + 3 * y;     // YES!
    z = (2 * x) + (3 * y); // YES!
```

Unary operators should be immediately preceded or followed by their operand.

```
    count ++; // NO!
    count++;  // YES!

    i --;     // NO!
    i--;      // YES!
```

Commas and semicolons are always followed by whitespace.

```
     for (int i = 0;i < 10;i++)   // NO!
     for (int i = 0; i < 10; i++) // YES!

     getPancakes(syrupQuantity,butterQuantity);  // NO!
     getPancakes(syrupQuantity, butterQuantity); // YES!
```

Figure 11 Spacing in Java code

2. Task 2 Critical Analysis of the Environmental Flexibility in Java

a) Programming in Java Using Simple Text Editor

The Java programs can be coded through simple text edition like Notepad. It has no requirement to use IDE built for java programming. Using IDE has a much easier way of doing java programming. The java programs can be created using just a simple text editor and can be compiled and run using the command line on windows and on the terminal in Linux. The only requirement is that you must have JDK installed on the system.

b) JDK Tools

The tools used for the java application are:

javac	The compiler used for java is known as javac. The compiler call is the file that contains the main application class. The compiler creates a separate file for each class of the program; They have the .class extension and are placed in the same directory as the source files.
java	IT is the launcher for Java applications. It is used to launch compiled java classes and executed compiled java code. Java applications are running with the help of Java interpreter, called for the main compiler, but the .class extension associated with it is omitted.
	Running an application that does not use a graphical interface will be done in a window system.
Javadoc	It is the API documentation generator. Java source code documentation is documented by JavaDoc projects, a collection of HTML pages describing the classes and methods in the code.

To generate the JavaDoc documentation, use a JDK tool, javadoc.exe, and the code comments are written according to strict rules.

Generate JavaDoc in Netbeans

In the Projects window, right-click on the project name to select Generate JavaDoc; another possibility is through the Run -> Generate JavaDoc option

applet viewer It can be used to run and debug applets without a web browser.

Applet viewer is a display area (container) that can be included in a web page and managed by a Java program. A program like this is also called gadget. The fundamental difference between an applet and an application is that an applet cannot be executed independently, but it will be executed by the browser where the applet is loaded.

jar JAR format (Java Archive) allows multiple files to be packaged in a single archive. A JAR file typically contains .class files and other auxiliary files (for example, images, audio files, etc.). It is used to bundle java application into a compressed file. It is used to distribute jar files to the application uses.

Here are some of the features of JAR files:

Security: The content of a JAR file can be digitally signed, so only those who recognise the signature can benefit from security privileges.

Compression: The JAR format allows compression of files (with ZIP format).

Package version information: A JAR file can retain data about the files it contains, such as the version, the manufacturer, etc.

Portability: This feature applies to the Java language in general, with everything related to it, so this archiving mechanism is a standard part of the Java API platform.

c) What are the Major Advantages and Disadvantages of Using an IDE

Integrated Development Environments (IDEs) are essentially projected to generate programs and code. It can be specified the ones of the most used IDEs which are NetBeans and Eclipse.

Advantages of using these application development tools:

The using of such IDE like NetBeans or Eclipse for example, helps the efforts of a programmer in creating a program. Some of the advantages are as follows:

- These applications development tools are universal, open-source platforms that integrate various application development tools.
- Independence of development language and operating environment
- Various content types HTML, Java, C, JSP, EJB, XML, GIF, ...
- Facilitates easy integration of various tools
- Tools can access various platform core functions
- It can be added new tools(plugins) to products already installed
- Location in source text on error lines
- Help (eventually context dependent) for class and method names, work
- Particularly important regarding the continuous increase in the number of classes and methods
- Facilitating the specification of search paths and libraries used
- Automatically create subdirectories for package classes

- Troubleshooting features using the graphical interface
- Different colouring of syntactic units (keywords, comments, etc.)
- Create and use directories for all application files
- Generating a program skeleton, depending on the type of application
- Autocomplete: Automatically add long names based on a prefix entered by user
- Large community of developers
- Based on the popularity Java enjoys among application developers
- Minimal time and efforts
- Sustain company's gauges
- Prolong administrative work

Drawbacks of Using an IDE

The programmer needs to be a bit cautious while working using IDE as it doesn't suit in every circumstance and may results in adverse effects.

1. To do programming via IDE is a complex mechanism and requires time to understand it.
2. For the beginners it is not suggestive to implement the program using advance IDE as the results may be very disappointing.
3. Any IDE can't update codes or resolve issues from the application

d) Discuss the Flexibility of a Chosen IDE, such as Eclipse and NetBeans

The time required for the development of larger applications can be greatly reduced by using an Integrated Development Environment (IDE) that provides some facilities for editing programs, debugging them, building applications and testing them with a user-friendly and easy-to-use graphical interface. Such a product integrates several different programs used in the command line: text editor, compiler, debugger, unit test support (JUnit), application builder, web server, etc.

For editing (writing and modifying sources) an IDE is helpful by reporting errors before compilation (a syntactic analysis is performed as instructions are entered), by completing some class or method names, by using help with the documentation Classes by refactoring to build applications, Ant builds files are automatically created for each project. The work unit of an IDE is a project that groups a variety of files into a folder structure: sources, compiled files, libraries, property files, build files. In general, a project corresponds to an application, but an application may be developed by several projects, some of the class libraries in the application and a project to start the application (with the main () method).

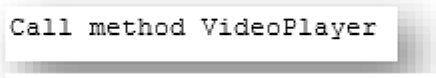
The most used IDE products for Java are Eclipse, NetBeans and IntelliJ IDEA, which enable the development of applications in several languages: Java, C, C ++, PHP, Groovy, and so on. Also, they can also be used as visual environments to create graphical user interfaces without manual programming. The user selects the graphics components, configures their properties and places them on a surface that represents the main application window and the IDE generates the source code which creates the user-friendly graphical user interface and helps to handle the events generated by the graphics components.

3. Task 3 Detailed Design of the System

a) Pseudo Code

1) Pseudo Code for VideoPlayer class:

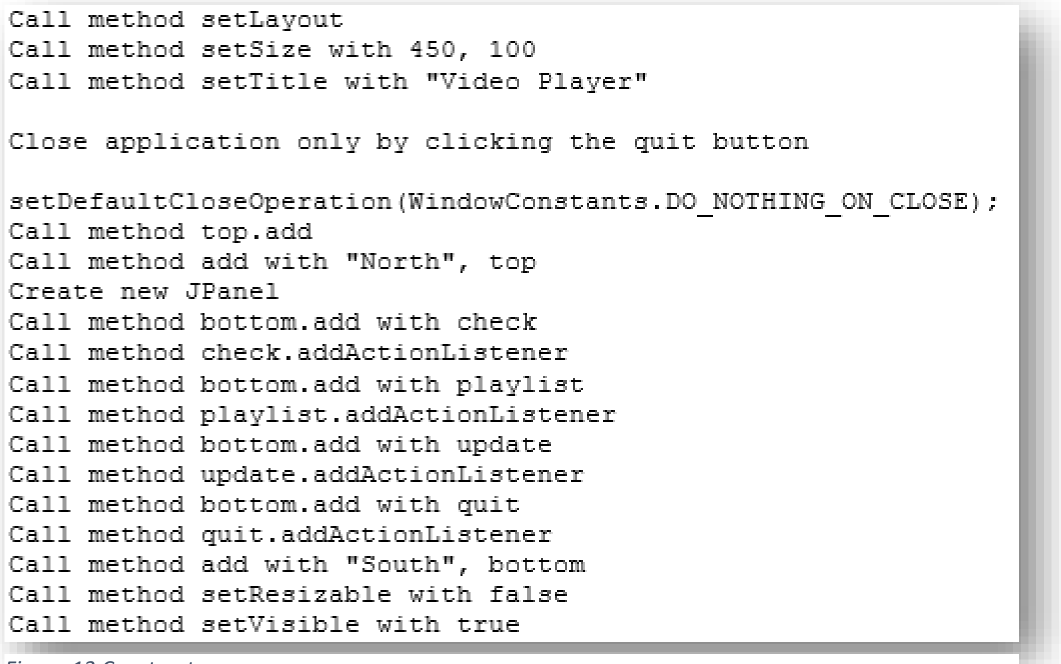
main



```
Call method VideoPlayer
```

Figure 12 Main method of the application

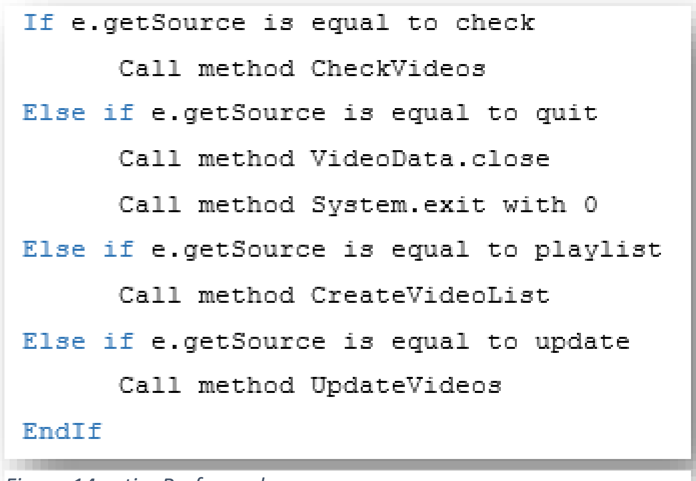
Constructor



```
Call method setLayout
Call method setSize with 450, 100
Call method setTitle with "Video Player"

Close application only by clicking the quit button

setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);
Call method top.add
Call method add with "North", top
Create new JPanel
Call method bottom.add with check
Call method check.addActionListener
Call method bottom.add with playlist
Call method playlist.addActionListener
Call method bottom.add with update
Call method update.addActionListener
Call method bottom.add with quit
Call method quit.addActionListener
Call method add with "South", bottom
Call method setResizable with false
Call method setVisible with true
```

Figure 13 Constructor

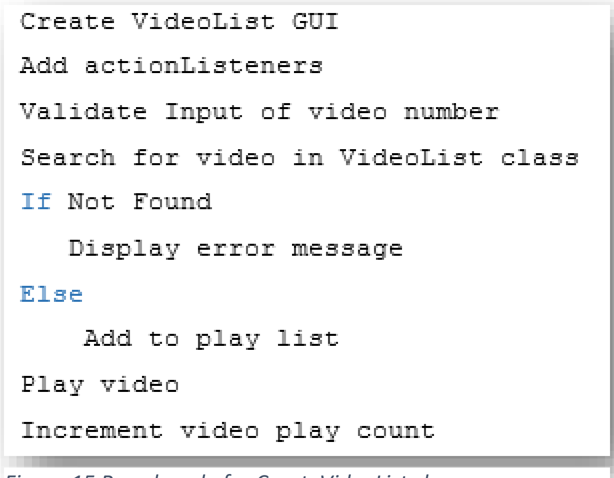
actionPerformed:



```
If e.getSource is equal to check
      Call method CheckVideos
Else if e.getSource is equal to quit
      Call method VideoData.close
      Call method System.exit with 0
Else if e.getSource is equal to playlist
      Call method CreateVideoList
Else if e.getSource is equal to update
      Call method UpdateVideos
EndIf
```

Figure 14 actionPerformed

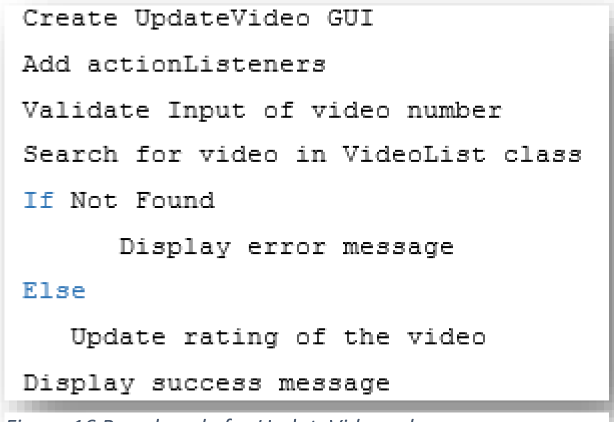
2) Pseudo Code for CreateVideoList class:



```
Create VideoList GUI
Add actionListeners
Validate Input of video number
Search for video in VideoList class
If Not Found
   Display error message
Else
    Add to play list
Play video
Increment video play count
```

Figure 15 Pseudocode for CreateVideoList class

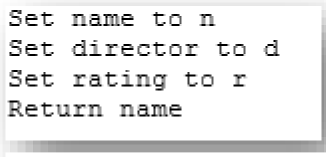
3) Pseudo Code for UpdateVideos class:



```
Create UpdateVideo GUI
Add actionListeners
Validate Input of video number
Search for video in VideoList class
If Not Found
      Display error message
Else
   Update rating of the video
Display success message
```

Figure 16 Pseudocode for UpdateVideos class

4) Pseudo Code for VideoData class:

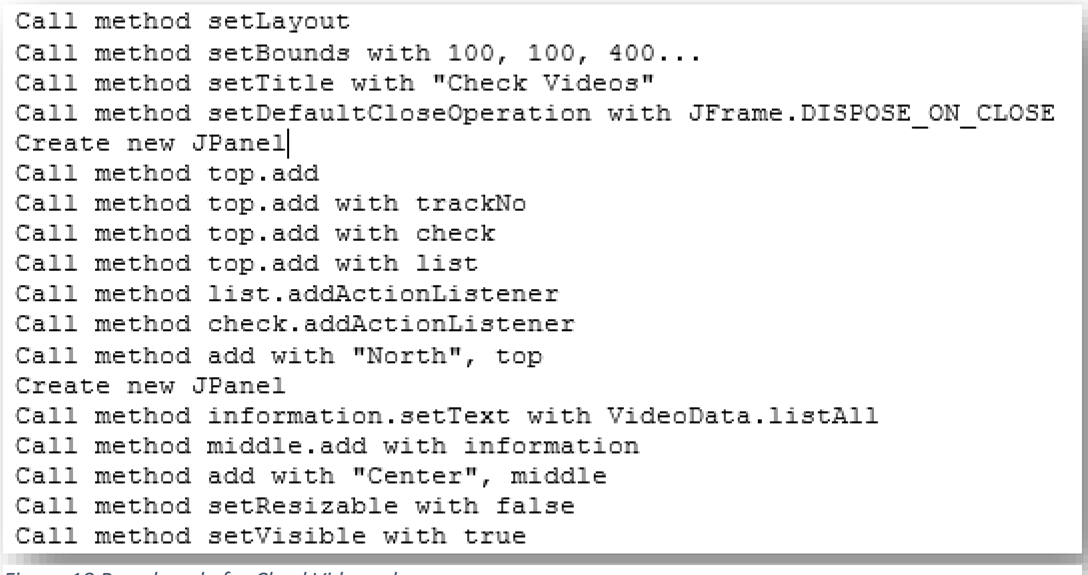


```
Set name to n
Set director to d
Set rating to r
Return name
```

Figure 17 Pseudocode for VideoData class

5) Pseudo Code for CheckVideos class:

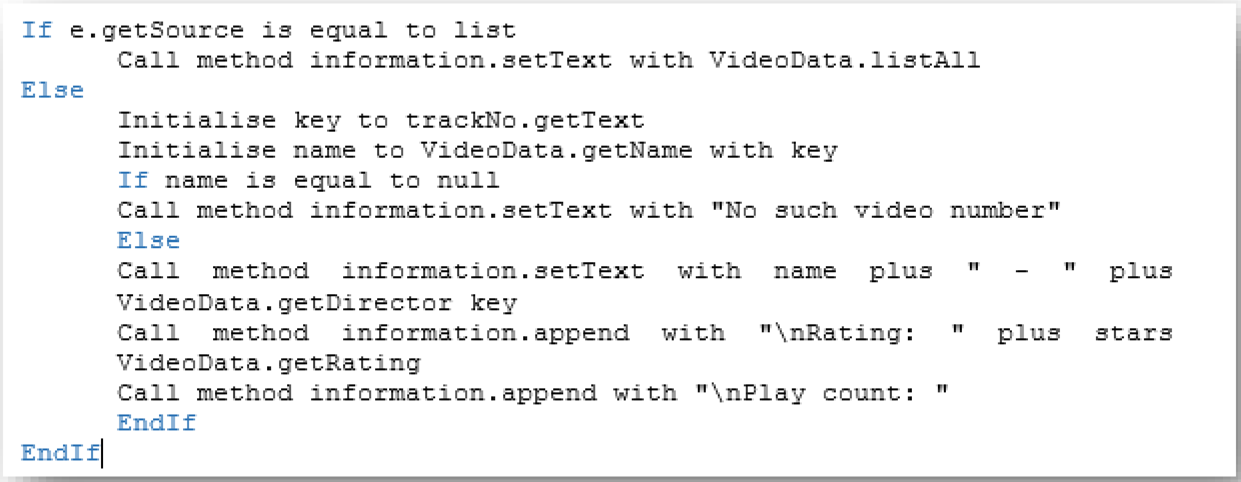
Constructor:



```
Call method setLayout
Call method setBounds with 100, 100, 400...
Call method setTitle with "Check Videos"
Call method setDefaultCloseOperation with JFrame.DISPOSE_ON_CLOSE
Create new JPanel
Call method top.add
Call method top.add with trackNo
Call method top.add with check
Call method top.add with list
Call method list.addActionListener
Call method check.addActionListener
Call method add with "North", top
Create new JPanel
Call method information.setText with VideoData.listAll
Call method middle.add with information
Call method add with "Center", middle
Call method setResizable with false
Call method setVisible with true
```

Figure 18 Pseudocode for CheckVideos class

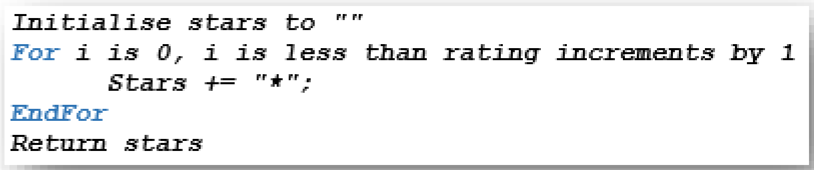
actionPerformed



```
If e.getSource is equal to list
      Call method information.setText with VideoData.listAll
Else
      Initialise key to trackNo.getText
      Initialise name to VideoData.getName with key
      If name is equal to null
      Call method information.setText with "No such video number"
      Else
      Call  method  information.setText  with  name  plus  "  -  "  plus
      VideoData.getDirector key
      Call  method  information.append  with  "\nRating:  "  plus  stars
      VideoData.getRating
      Call method information.append with "\nPlay count: "
      EndIf
EndIf
```

Figure 19 Code of actionPerformed

stars



```
Initialise stars to ""
For i is 0, i is less than rating increments by 1
      Stars += "*";
EndFor
Return stars
```

Figure 20 Pseudocode for stars rating video

b) Virtual Video Player Class Diagram

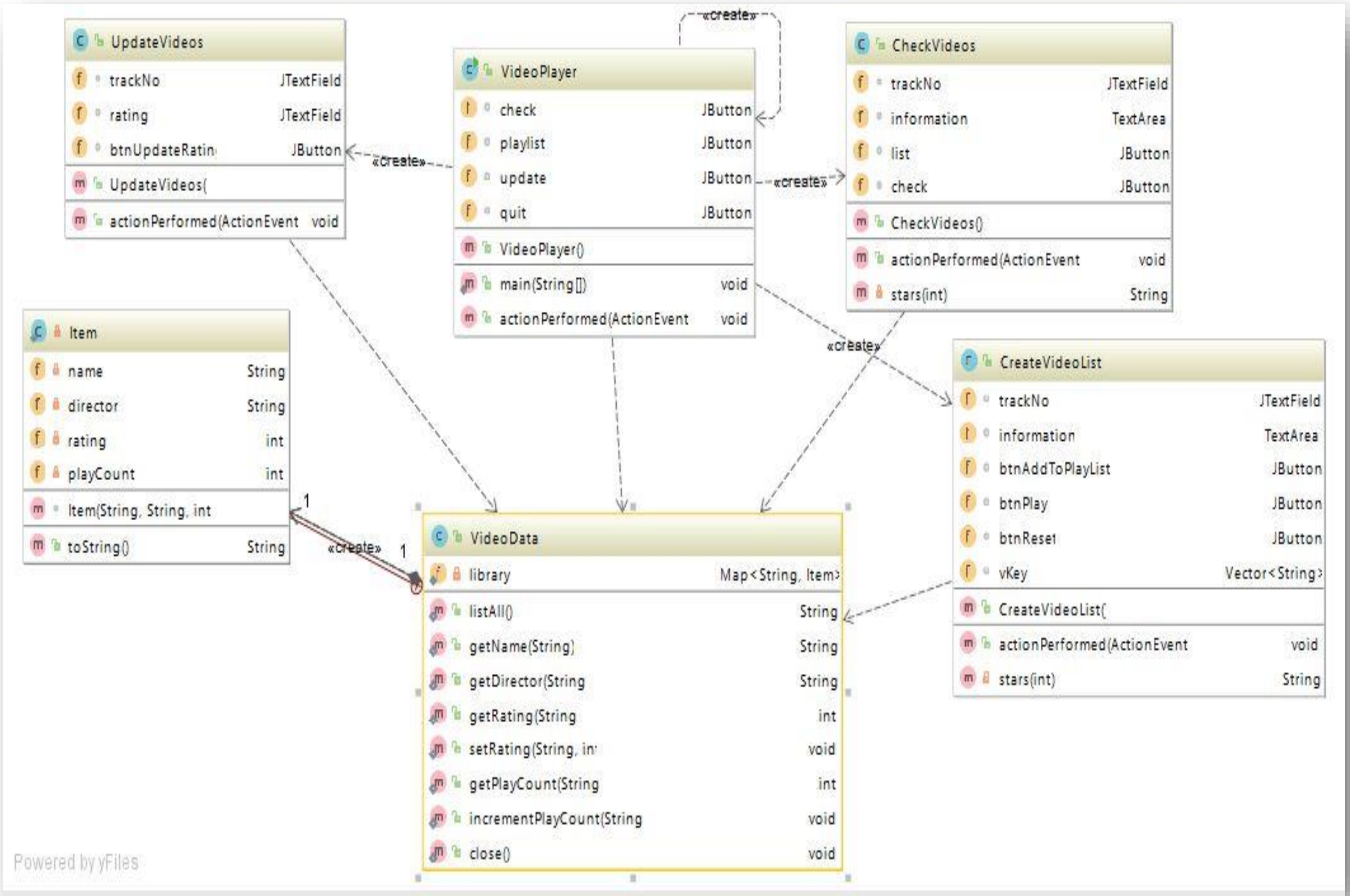


Figure 21 Class Diagram for virtual video player application

c) Data Dictionary of Virtual Video Player system

Purpose	Identifier	Data Type	Storage (bytes)	Reason
Input variable	movieSelection	int	4	2 digits
Input variable	movieRating	int	4	Number of stars
Input variable	movieNumber	int	4	2digits
Output variable	movieList	String	As required	Digits and characters (text)
Output variable	movieTitle	String	As required	Digits and characters (text)
Output variable	movieRating	char	4	Star Rating - special character
Output variable	movieCouting	int	As required	Counting value
Input variable	movieSelection	int	4	2 digits
Input variable	movieRating	int	4	Number of stars
Input variable	movieNumber	int	4	2digits

Figure 22 Table of Data Dictionary of Virtual Player system

d) Inputs and Outputs dialog designs for VideoPlayer program:

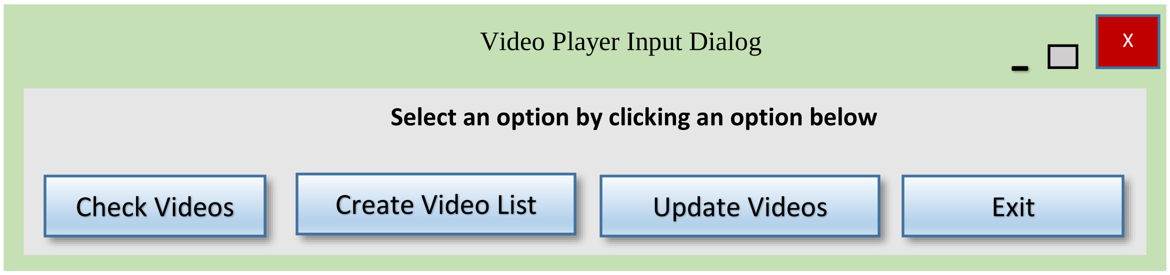

Video Player Input Dialog
X
Select an option by clicking an option below
Check Videos
Create Video List
Update Videos
Exit

Figure 23 Video Player Input Dialog

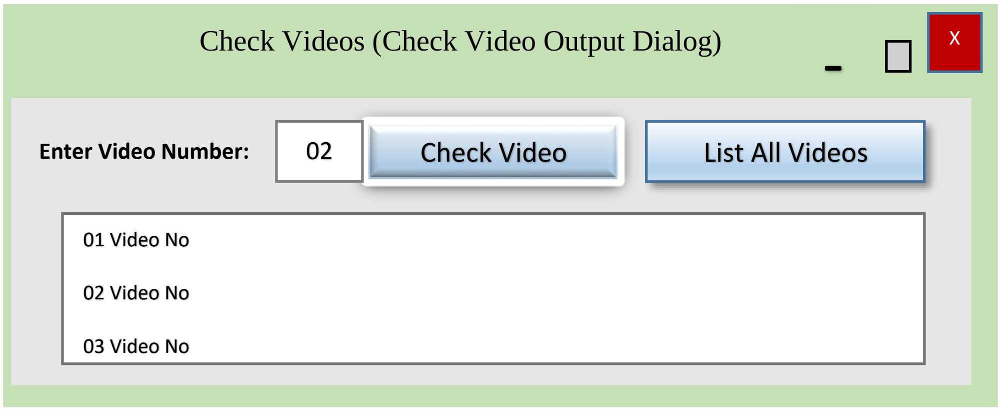
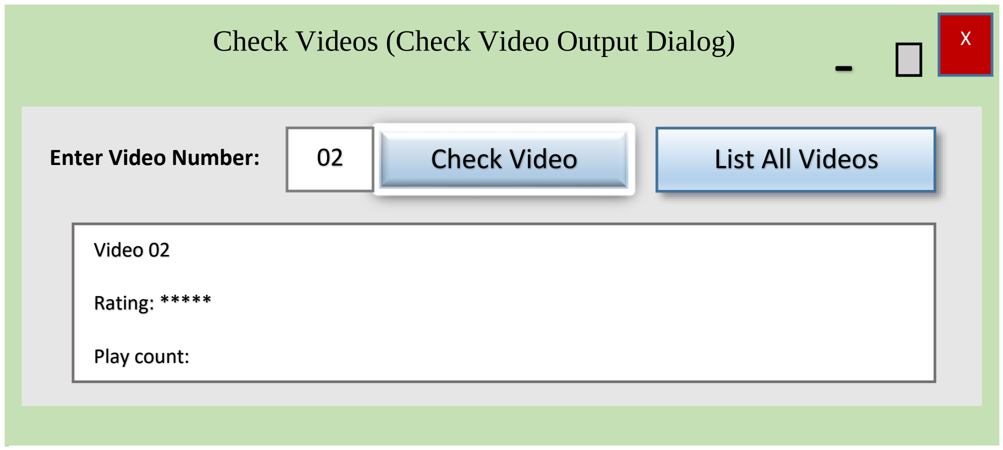


Figure 24 Check video number input dialog

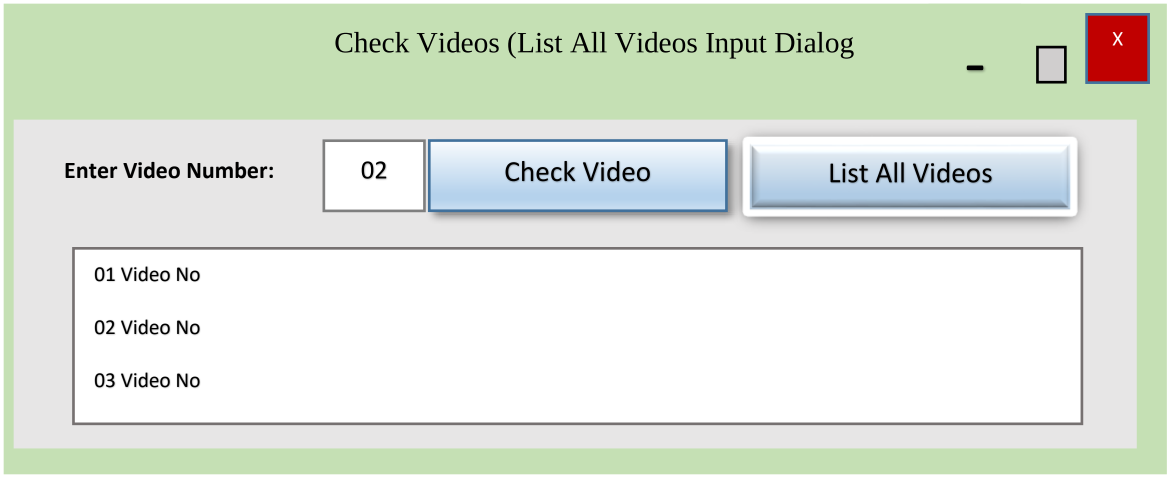


Video 02

Rating: *****

Play count:

Figure 25 Check video output dialog



01 Video No

02 Video No

03 Video No

Figure 26 List all the videos output dialog

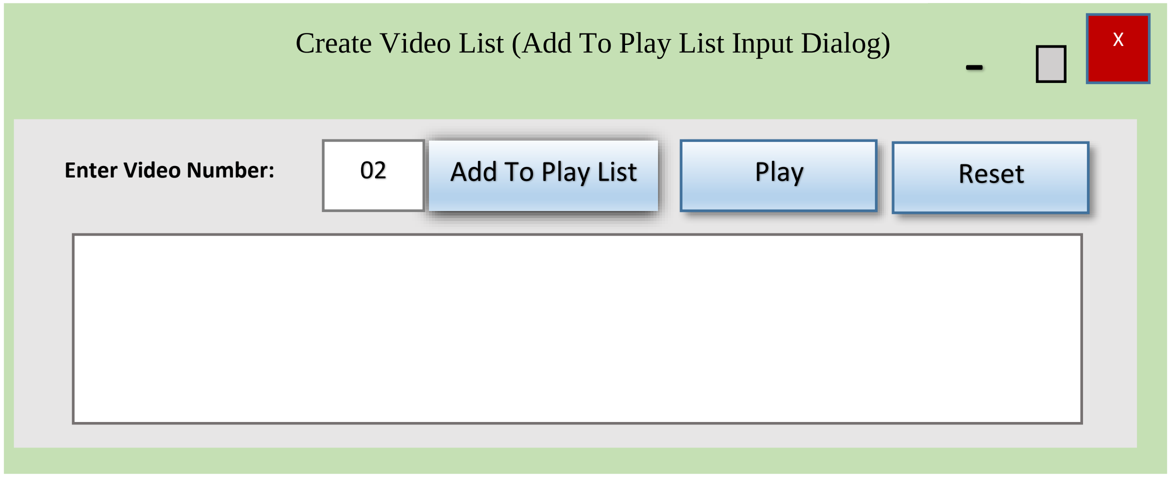


Figure 27 Add to Play List Input Dialog

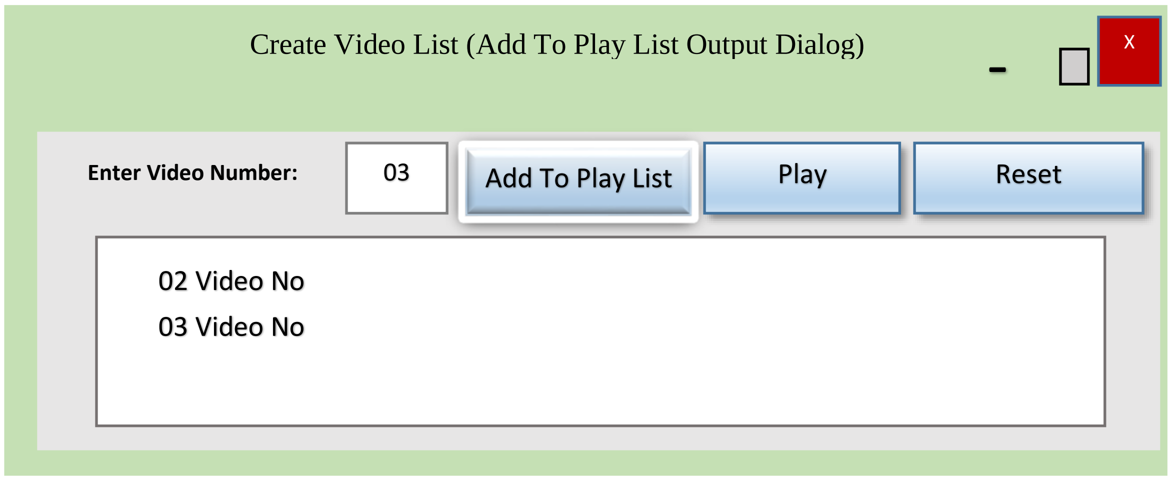


Figure 28 Add to Play List Output Dialog

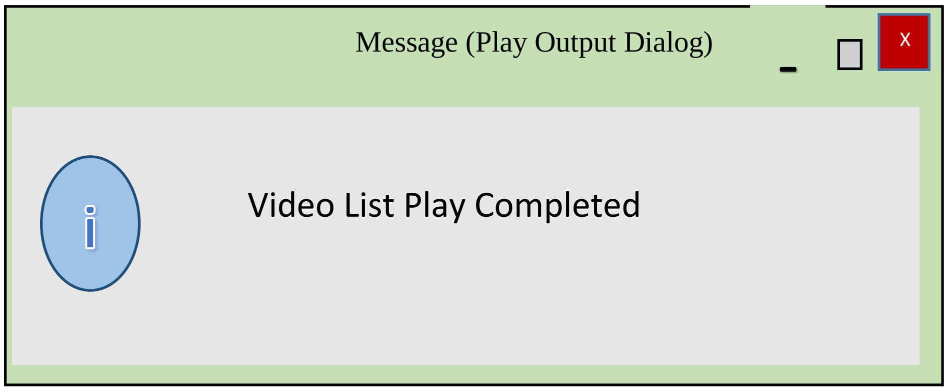


Figure 29 Play Output Dialog

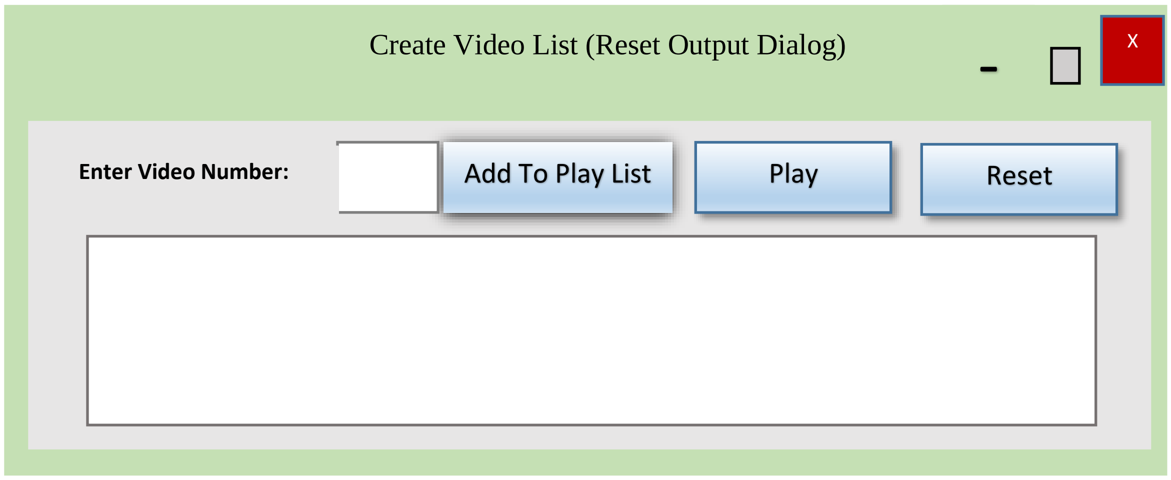
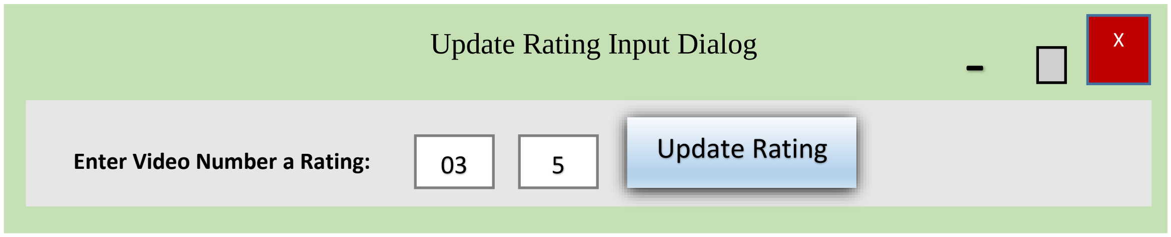
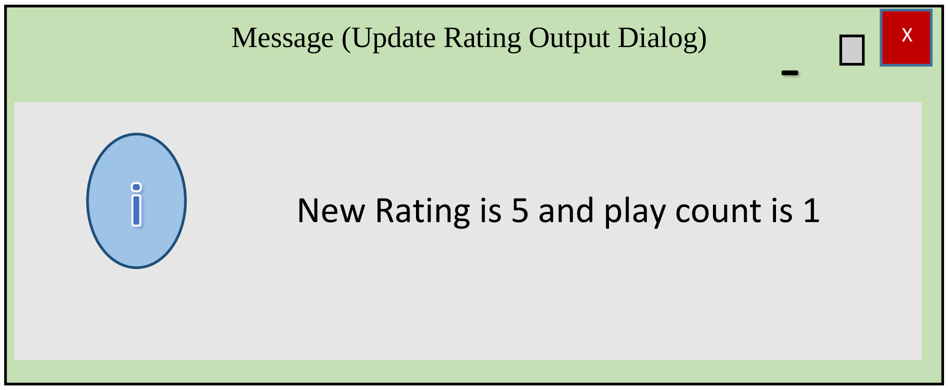


Figure 30 Reset output dialog



Enter Video Number a Rating: 03 5

Figure 31 Update Rating Input Dialog



New Rating is 5 and play count is 1

Figure 32 Update Rating Output Dialog

4. Task 4 Implementation of the System

CreateVideoList

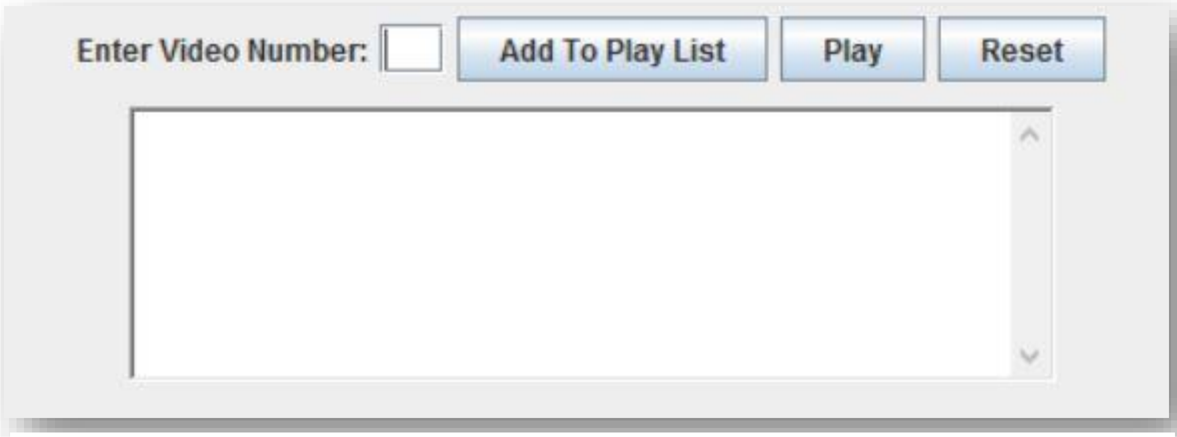


Figure 33 Output for CreateVideoList

Code for CreateVideoList class

```
import java.awt.*;
import java.awt.event.*;
import java.util.Vector;
import javax.swing.*;

public class CreateVideoList extends JFrame implements ActionListener {

    JTextField trackNo = new JTextField(2);
    TextArea information = new TextArea(6, 50);
    JButton btnAddToPlayList = new JButton("Add To Play List");
    JButton btnPlay = new JButton("Play");
    JButton btnReset = new JButton("Reset");

    Vector<String> vKey = new Vector<String>();

    public CreateVideoList() {
        setLayout(new BorderLayout());
        setBounds(100, 100, 500, 200);
        setTitle("Create Video List");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        JPanel top = new JPanel();
        top.add(new JLabel("Enter Video Number:"));
        top.add(trackNo);
        top.add(btnAddToPlayList);
        top.add(btnPlay);
        top.add(btnReset);

        btnAddToPlayList.addActionListener(this);
        btnPlay.addActionListener(this);
        btnReset.addActionListener(this);
        add("North", top);
        JPanel middle = new JPanel();
        middle.add(information);
        add("Center", middle);




     setResizable(false);
    setVisible(true);
}

public void actionPerformed(ActionEvent e) {
    if (e.getSource() == btnAddToPlayList) {
        if (trackNo.getText().trim().equals("")) {
            JOptionPane.showMessageDialog(this, "Enter Video Number");
        } else {
            String key = trackNo.getText();
            String name = VideoData.getName(key);
```

```
                    if (name == null) {
                        JOptionPane.showMessageDialog(this, "No such video number");
                    } else{
                         if (information.getText().length() != 0) {
                        information.setText(information.getText() + "\n" + key + " " + name + " - " +
          VideoData.getDirector(key));
                    } else {
                        information.setText(key + " " + name + " - " + VideoData.getDirector(key));
                    }
                         vKey.add(key);
                         trackNo.setText("");
                    }
                }
            }
            else if (e.getSource() == btnPlay) {
                for(String s:vKey){
                    VideoData.incrementPlayCount(s);
                }
                JOptionPane.showMessageDialog(this, "Video List Play Completed");
            }else if (e.getSource() == btnReset) {

                vKey.clear();
                information.setText("");
            }

        }

        private String stars(int rating) {
            String stars = "";
            for (int i = 0; i < rating; ++i) {
                stars += "*";
            }
            return stars;


    }


    }
```

UpdateVideos

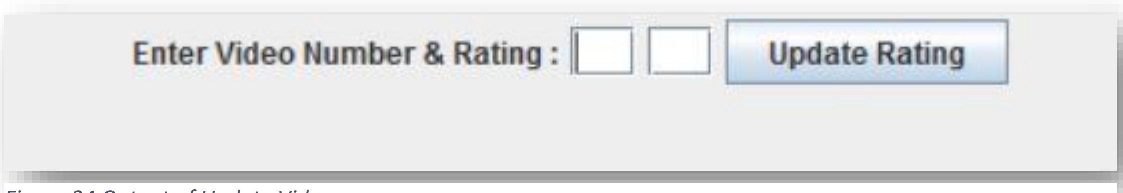
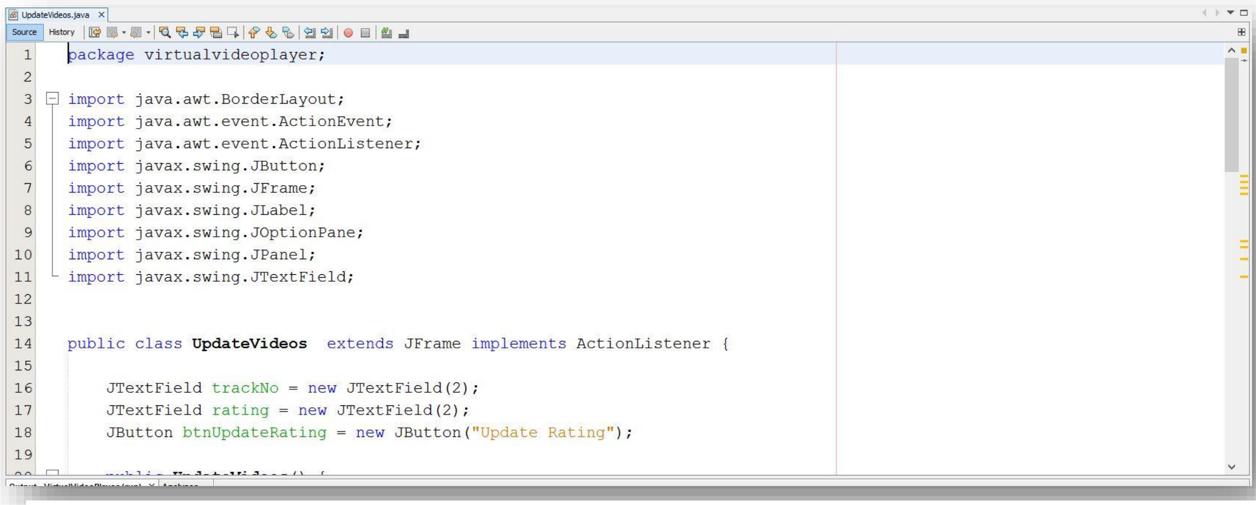


Figure 34 Output of Update Videos

Code for UpdateVideos class

```
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;


public class UpdateVideos  extends JFrame implements ActionListener {

    JTextField trackNo = new JTextField(2);
    JTextField rating = new JTextField(2);
    JButton btnUpdateRating = new JButton("Update Rating");

    public UpdateVideos() {
        setLayout(new BorderLayout());
        setBounds(100, 100, 500, 100);
        setTitle("Update Rating");
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        JPanel top = new JPanel();
        top.add(new JLabel("Enter Video Number & Rating :"));
        top.add(trackNo);
        top.add(rating);
        top.add(btnUpdateRating);


        btnUpdateRating.addActionListener(this);
        add("North", top);
         setResizable(false);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {

        if (e.getSource() == btnUpdateRating) {
            if (trackNo.getText().trim().equals(""))  {
                JOptionPane.showMessageDialog(this, "Enter Video Number");
            }else if (rating.getText().trim().equals(""))  {
                JOptionPane.showMessageDialog(this, "Enter Rating Number");
            } else {
                String key = trackNo.getText();
                String name = VideoData.getName(key);
                if (name == null) {
                    JOptionPane.showMessageDialog(this, "No such video number");
                } else{
                     VideoData.setRating(key, Integer.parseInt(rating.getText()));
                     JOptionPane.showMessageDialog(this,          "New          Rating
"+VideoData.getRating(key)+" and play count is "+VideoData.getPlayCount(key));
                     trackNo.setText("");
                     rating.setText("");

                }
            }
        }
    }

}
```

<u>The program is developed using Net Beans</u>



```
package virtualvideoplayer;

import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;


public class UpdateVideos  extends JFrame implements ActionListener {

    JTextField trackNo = new JTextField(2);
    JTextField rating = new JTextField(2);
    JButton btnUpdateRating = new JButton("Update Rating");

```

Figure 35 Using NetBeans to create the code for the application

5. Task 5 Test the System

5.1 Test Plan

The system was tested based on the below test cases

Table 1 –Specification Testing and Black Box Functional Tests

Test Case	Expected Result	Current Result	Functional or non-functional		Date of the test	Comments	Action required
Initiating and running the program	Initiating and running the program	Opened and running correctly	✓		10/04/2017	Open, load and run VirtualVideoPlayer project with NetBeans IDE	Test Passed
Test to resize the window	The windows of the application should not be resizable	The test found that the windows are not resizable	✓		10/04/2017	The test is checking if the windows are resized when the user is dragging the sides or the corners of the windows	Test Passed
Press X button to close the window	The window should be close by pressing X from the right-top of the window	The test found that the X button close the window	✓		10/04/2017	The test is verifying the functionality of X button (top-right of windows)	Test Passed
Test the functionality of "Check Videos" button	Is opening a list with all the recorded videos	Is opening a dialog box with all the current videos	✓		10/04/2017	The test is intended to check the functionality of "Check Videos" button by click on it and should show the videos list	Test Passed

Validating the video ID	Try to click check video button with the blank field	This test was performed to check if field validation on the application was working correctly	✓		10/04/2017	The test video button was pressed without any input on the field, the validation worked correctly and a message was prompted saying: "Please Enter Video Number".	Test Passed
Produce a video playlist	Provide a new video playlist	All listed videos were added to a new playlist and were added correctly	✓		10/04/2017	The meaning of the test is to check if by pressing "Create Video List" button is created a new playlist with the chosen videos	Test Passed
Checking functionality against requirements	The program must correctly store and display video data	The program is storing and displaying data correctly	✓		10/04/2017	All the inputs for the desired video data should give the required results	Test Passed
Display Error Message	The program must display the correct error messages dialog	The program is displaying correct error message dialogs.	✓		10/04/2017	The user wants to check how is responding the application to the inputs which should give error messages	Test Passed
Reset button practicality	Should clear a playlist	By pressing "Reset" button from "Create Video List" menu is clearing the current playlist	✓		10/04/2017	By clicking the "Reset" button under "Create Video List" menu is deleting the actual playlist	Test Passed
Testing the "Update Videos" button practicality for rating a video	Press "Update Videos" button to update the rating for a video number	By add a video number and a number of starts for selected video is confirmed the correct updated rating	✓		10/04/2017	This test is checking the functionality of updating the rating for a selected video	Test Passed
"Play" button practicality	Every time by pressing "Play" button on a desired video number is increased the play count	Every time by pressing "Play" button is increasing the play number of a selected video number	✓		10/04/2017	This test is intended to verify if the counting of playing a video is works according to the task requirements	Test Passed

	White box Tests						
Comment code	The code must be indented and well commented	The code is indented and well commented	✓		10/04/2017	The code must be explained in a proper comment format	Test Passed
Input testing	The program must validate user input	The program validated user input	✓		10/04/2017	Validation of a video's data must be correct when entering the requested inputs	Test Passed
Valid declaration	There must be meaningful variable declaration and methods names.	Classes, methods, and variables are meaningful and expressive	✓		10/04/2017	This test is checking if all the variables and methods are writing correctly	Test Passed

- There was some compile time error initially when the program was created. But after taking a closer look at the code and interpreting the compile time error message, it finally was able to successfully compile the code.
- While running the application, there was some input validation that was not implemented and was creating runtime errors. But after doing the necessary changes in the code the application is successfully generating the desired output.
- While first running the application, it was found that the application is not doing anything while entering the video number to create a playlist. So, after changing the code, now the application is generating a dialog box if any of the video number entered is not found.

5.2 Feedback from a Cirimpei Andrei, a colleague course

Programmer: How satisfied are you with the quickness and effectiveness at which the application is able to respond to your requests?

Programmer: How do you think is looking the design of the application?

Cirimpei Andrei: *The design is simple and easy to memorise it, good for a video player to be used day by day.*

Programmer: The Inputs/Outputs met the requirements of the application?

Cirimpei Andrei: *All the I/O of the application are responsive to the expected results and easily to manage them.*

Programmer: What improvements can be made in the future for the application?

Andrei Cirimpei: *A database can be created for the video lists and integrated in the Video Player java application.*

Programmer: What is your overall rating related to the requirements compared to the results of the application?

Andrei Cirimpei:

	Below Anticipations	**Same Expectations**	**More than expected**
Appealing	(X)		
Suitable		(X)	
Expert	(X)		
User Friendly		(X)	

Recommendation for Improvements

Taking into account both the opinion of lecturer Andre Beckley and the colleague Andrei Cirimpei in the development of this Java application as well as the overall image of the developer who developed this application it was concluded that it fulfils all the specifications and requirements of the Assignment Brief. However, there are some aspects that are recommended to be improved:

- Adding an external database to the source code that includes all the videos or inserting a line of code that directly extracts the path of that video from the local station
- Improving application graphics for greater user appeal
- Comparing the current version of the application with other applications of this kind in real life and trying to implement some useful functions in using the application by the users

6. Task 6 Be able to Document the Java Solution

a) Create a Technical Manual of the System with Guidance

1. Installation and set-up

The user can use the application using the jar file from the application’s folder. It is needed to distribute the application bundled in the .jar file. There is no requirement to distribute the source code. The only requirement to run the .jar file on the user system is that the JRE (Java Runtime Environment) must be installed on the user's computer.

The next steps are taking into consideration when the user wants to run Virtual Video Player application:

Step 1 Ensure that JRE is installed on the local system

Step 2 The application can be installed on the user’s system by save it in a zipped format on an external optical storage (USB, CD) or can be sent it to the user online.

Step 3 After the download or transfer of it from the external source the files’ applications are unzipped in the directory where the user wants to have it.

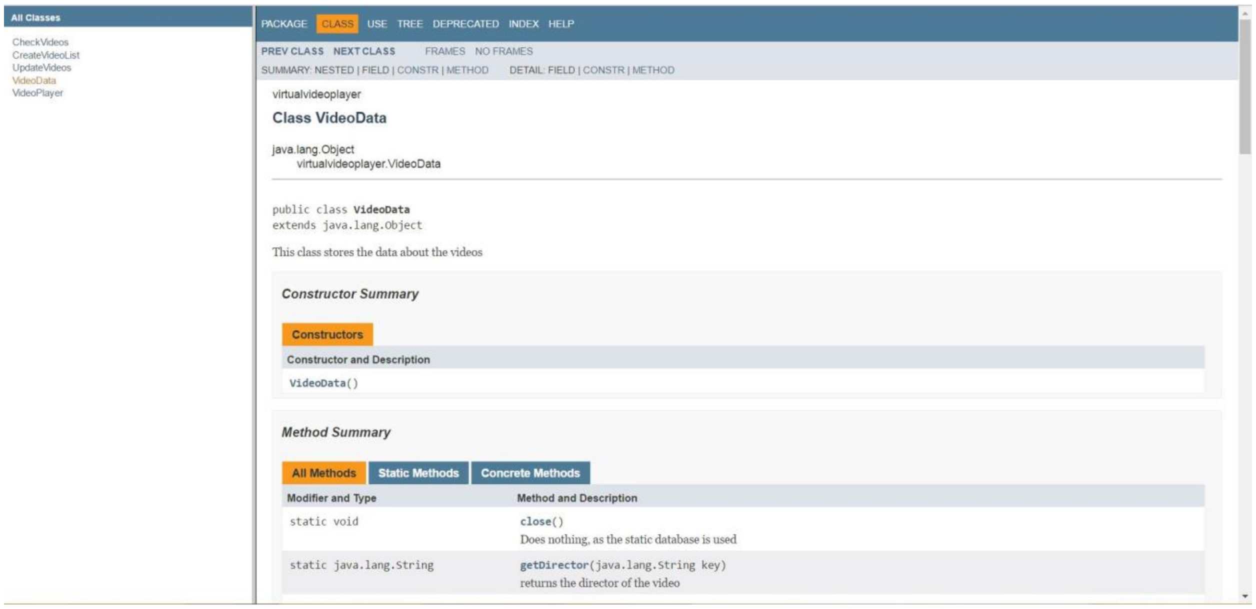
Step 4 After the download or transfer of it from the external source the application’s files are unzipped in the directory where the user wants to have it.

Step 5 Run ***videoplayer-1.0.jar*** file from the ***virtualplayer folder*** where it is located

If the user wants to export the project then he needs to go to ***File*** menu from NetBeans IDE and click ***Export*** option to export it as a ZIP file.

2. API Documentation

The appropriated java doc is being generated using the standard java doc syntax


All Classes
CheckVideos
CreateVideoList
UpdateVideos
VideoData
VideoPlayer
PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

virtualvideoplayer

Class VideoData

java.lang.Object
    virtualvideoplayer.VideoData

```
public class VideoData
extends java.lang.Object
```

This class stores the data about the videos

Constructor Summary

Constructors

Constructor and Description
`VideoData()`

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method and Description
`static void`	`close()` Does nothing, as the static database is used
`static java.lang.String`	`getDirector(java.lang.String key)` returns the director of the video

Figure 36 Generate the documentation for VirtualVideoPlayer application

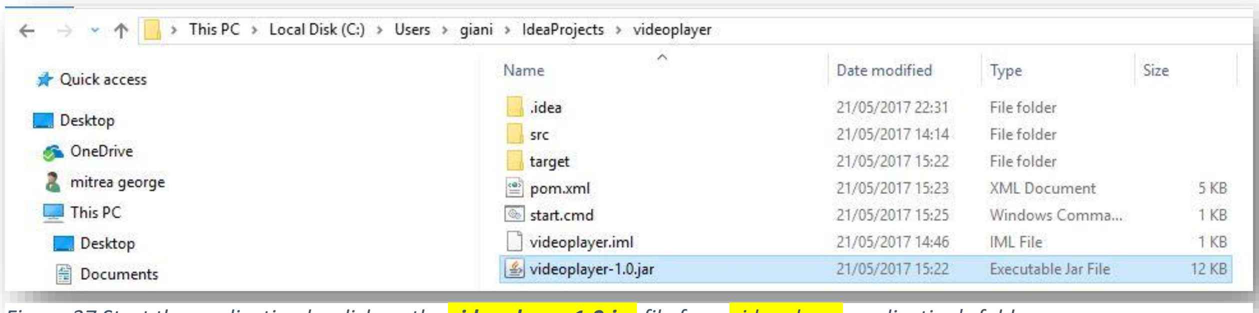
b) Create a Technical Manual of the System with Guidance

1. What is the Purpose of the System?

The meaning of this application development is to bring on the entertainment market a tool for watching the videos in a simple way of use by the users, to create playlists with the desired videos most played by the users and all of these in an opened source of using.

2. How to start the application?

To run the application the user needs to click on ***videoplayer-1.0.jar*** file from the ***virtualplayer folder*** where it is located:


This PC > Local Disk (C:) > Users > giani > IdeaProjects > videoplayer
Quick access
Desktop
OneDrive
mitrea george
This PC
Desktop
Documents

Name	Date modified	Type	Size
.idea	21/05/2017 22:31	File folder	
src	21/05/2017 14:14	File folder	
target	21/05/2017 15:22	File folder	
pom.xml	21/05/2017 15:23	XML Document	5 KB
start.cmd	21/05/2017 15:25	Windows Comma...	1 KB
videoplayer.iml	21/05/2017 14:46	IML File	1 KB
videoplayer-1.0.jar	21/05/2017 15:22	Executable Jar File	12 KB

*Figure 37 Start the application by click on the **videoplayer-1.0.jar** file from videoplayer application's folder*

3. What are the features of the application?

The application provides a quick access to the video list

4. How to use the system?

The main screen

It is the main screen of the application.

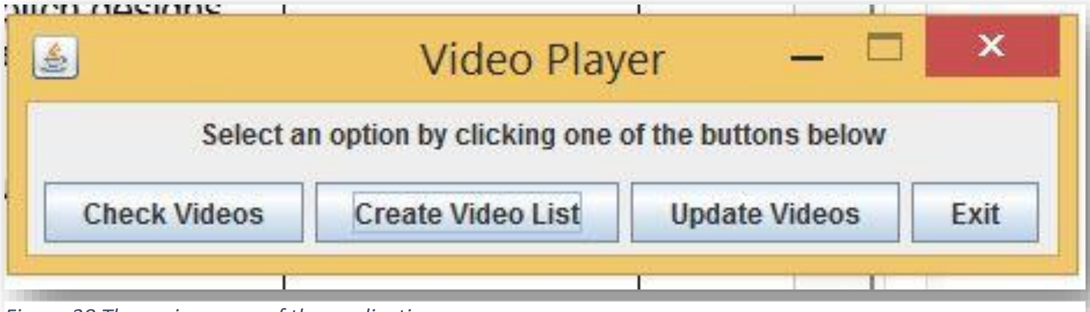
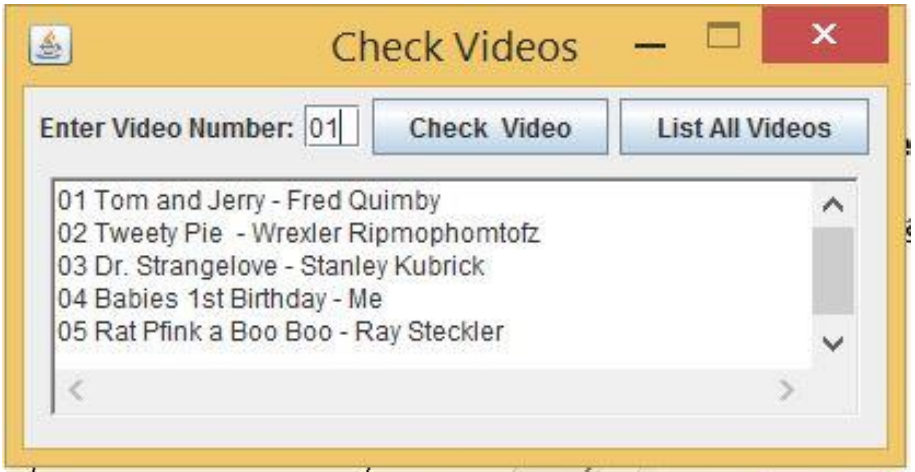


Figure 38 The main screen of the application

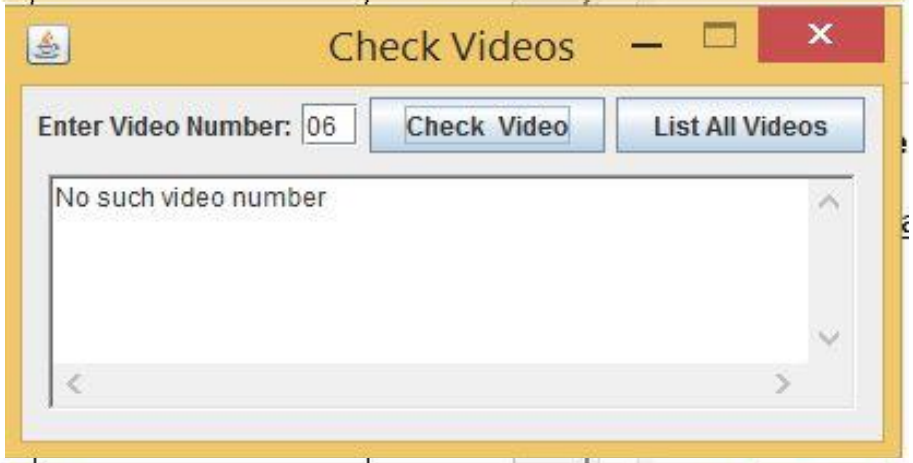
Check Video

This window can be used to check video details. Details like play count and rating.



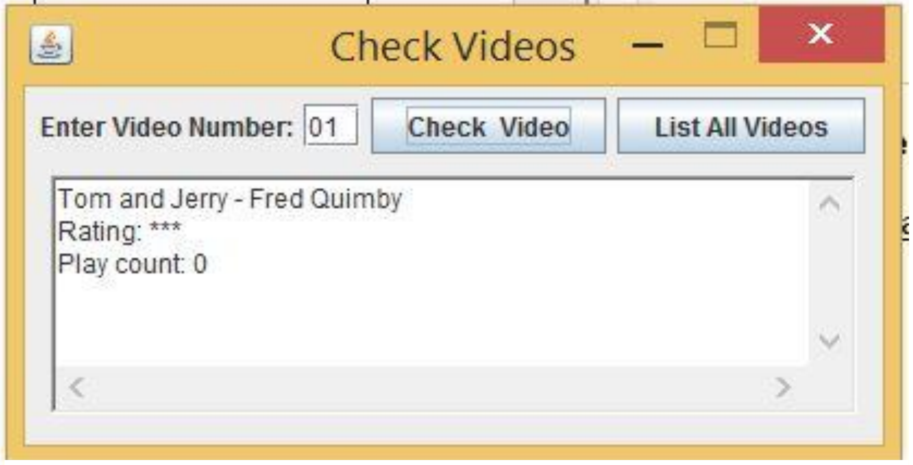
Enter Video Number: 01

```
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
```



Enter Video Number: 06

```
No such video number
```



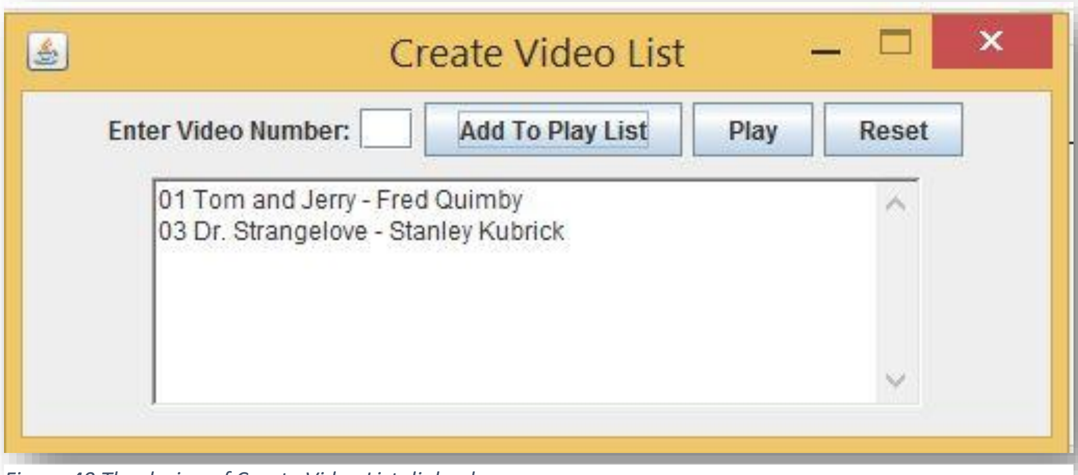
Enter Video Number: 01

```
Tom and Jerry - Fred Quimby
Rating: ***
Play count: 0
```

Figure 39 The interface of Check Videos menu

Create Video List

This screen will provide the functionality to the user to create video list. The user can enter the video number and if the video with that number found, it will be added to the list, otherwise the application will display an error message. The user can play the videos using the play button and can clear the list using the reset button.

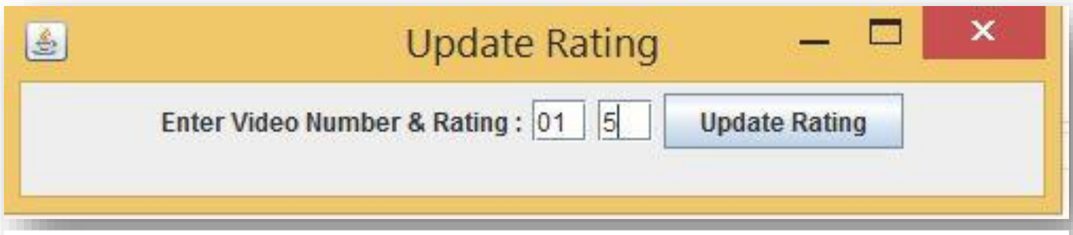


```
01 Tom and Jerry - Fred Quimby
03 Dr. Strangelove - Stanley Kubrick
```

Figure 40 The design of Create Video List dialog box

Update Video

This can be used to update the rating of an item. If the item associated with that number finds, it will update its rating, otherwise the application will display an error message.



Enter Video Number & Rating : 01 5

Figure 41 The screenshot of Update Rating to update the rating for a video number

5. Other operational features

All the necessary features required for this application were included and described in question 3 from Create a Technical Manual of the System with guidance.

Bibliography

(Etutorials.org, 2015) Netbeans.org, (2015). *Welcome to NetBeans*. [online] Available at: https://netbeans.org/ [Accessed 7 Dec. 2015].

Docs.oracle.com, (2015). *Advantages of Exceptions (The Java™ Tutorials > Essential Classes > Exceptions)*.

[online] Available at: https://docs.oracle.com/javase/tutorial/essential/exceptions/advantages.html [Accessed 7 Dec. 2015].

www.tutorialspoint.com, (2015). *Java Inheritance*. [online] Available at: http://www.tutorialspoint.com/java/java_inheritance.htm [Accessed 7 Dec. 2015].

Umlet.com, (2015). *UMLet - Free UML Tool for Fast UML Diagrams*. [online] Available at: http://www.umlet.com/ [Accessed 10 Dec. 2015].

Everywhere, J. (2016). *Java's top 20: The most used Java libraries on GitHub*. [online] JavaWorld. Available at: http://www.javaworld.com/article/2924315/open-source-tools/javas-top-20-the-most-used-java-libraries-on-github.html [Accessed 5 Jan. 2016].

Torchiano, M. (2016). *Patterns for Java Program Testing*. [online] Trondheim, Norway. Available at: http://softeng.polito.it/torchiano/papers/WTiXP2002.pdf [Accessed 6 Jan. 2016].

Libraries, 4. (2014). *45+ Most Useful Java Libraries | FromDev*. [online] Fromdev.com. Available at: http://www.fromdev.com/2014/10/most-widely-used-java-libraries.html [Accessed 6 Jan. 2016].