UNIVERSIDAD DE GRANADA

E.T.S. de Ingenierías Informática y de Telecomunicación

Práctica Final Sistemas Multimedia

Curso 2017-2018

Alumno: Javier Galera Garrido

Profesor: Jesús Chamorro M.

Índice:

En esta documentación se explica el funcionamiento y justificación de mi PaintFinal.

Como presentación para este proyecto, podemos dividir el funcionamiento de mi programa en tres apartados fundamentales: "Gráficos e Imágenes" , "Sonido" y "vídeo".

En el primer caso, **"Gráficos e Imágenes"**, abordamos distintas formas como el punto, la recta, el rectángulo y el óvalo entre otros y la posibilidad de editar imágenes, ya sea aplicando un filtro de brillo, rotación, etc.
Para las distintas formas se ha creado una clase propia como puede ser "MyRectangle.java" de tal forma que obtenemos una abstracción propia cambiando la típica idea de pintar en un lienzo formas, para obtener distintas formas que se pintarán en un lienzo, es decir, nuestras formas serán independientes de las otras, teniendo así sus propios atributos y pudiendo ser modificadas sin verse afectadas el resto de ellas. Por último, es necesario añadir que también es posible seleccionar y modificar las distintas formas una vez han sido pintadas.

En el segundo caso, **"Sonido"**, se pretende que el usuario pueda reproducir un sonido o grabarlo. Y que este sonido pueda ser seleccionado mediante una lista desplegable.

Para el tercer caso, **"vídeo"**, la aplicación permitirá la reproducción y captura de vídeo. Es decir, podremos reproducir un vídeo en función de su extensión e incluso podremos encender nuestra propia WebCam y tomar una instantánea, la cual, podrá ser tratada como una imagen, haciendo posible utilizar tantas operaciones como tengamos en el apartado "Imágenes".

1.Requisitos Funcionales:

En nuestra entrega se piden ciertos requisitos funcionales a la hora de realizar los "Gráficos":

Debemos de poder pintar:
1.Punto ----------> En la clase "MyLine.java"
2.Linea Recta ----------> En la clase "MyLine.java"
3.Rectángulo ----------> En la clase "MyRectangle.java"
4.Elipse ----------> En la clase "MyOval.java"
5.Curva con un punto de control -----> En la clase "MyQuadCurve.java"
6.Trazo libre ----------> En la clase "FreeDraw.java"
7.Una forma persona ----------> En la clase "MyForm.java"
Además hemos añadido:
8.Una espiral ----------> En la clase "MySpiral.java"

De los cuáles deben tener sus propios **atributos**:
·Color: Se podrá elegir el color del trazo y del relleno. Deberán aparecer una serie de colores predeterminados(en nuestro caso negro para el borde y blanco para el relleno).
·Trazo: Se podrá modificar el grosor y el tipo de discontinuidad del trazo. En nuestro caso, podemos dibujar línea continua y dos tipos diferentes de líneas discontinuas.
·Relleno: Se podrán rellenar las formas, de 3 formas diferentes:no rellenable, rellenado con un color liso o rellenado con un degradado.
·Alisado de bordes: se podrán activar o desactivar el alisado.
·Transparencia: se podrá establecer una transparencia asociada a una forma.

Respecto a "Imágenes" debemos tener las siguientes opciones:

1.Duplicar la imágen.
2.Modificar el brillo.
3.Filtros de emborronamiento, enfoque y relieve.
4.Contraste normal, iluminado y oscurecido.
5.Negativo (invertir colores).
6.Extracción de bandas.
7.Conversión de los espacios RGB, YCC y GRAY.
8.Giro libre mediante deslizador.

9.Sepia.
10.Un operador "LookUpOp" definida por el estudiante.
11.Operación aplicada componente a componente.
12.Operación aplicada pixel a pixel.

Opcionalmente tenemos:
·Tintado.
·Ecualizador.
·Umbralización.

Para "Sonido":

Debemos poder reproducir y grabar sonidos. Además la reproducción se aplicará sobre los formatos y códecs reconocidos por Java.

Para "vídeo":

Se debe poder reproducir un vídeo y mostrar la secuencia que esté captando la WebCam.

2.Descripción y análisis de la práctica

2.1. "Tipo de Ventanas"

Es importante explicar cómo funcionan las diferentes ventanas que encontraremos en la aplicación. Para ello empezaremos explicando quién contiene a quién.

Al abrir la aplicación nos encontramos con un escritorio principal en el cual se encuentran las distintas herramientas, formas, filtros, etc. que han sido implementados es lo que el programa conoce como "mainWindow", objeto proveniente de mainWindow.java. En dicha clase se implementan los distintos eventos en función del botón seleccionado. A su vez será el contenedor de las distintas ventanas internas que tiene la aplicación.

Para explicar las distintas ventanas internas las describiremos a partir de su jerarquía:

"InternalWindowSM"

Esta clase padre de todas las ventanas internas sirve como abstracción de todas ellas. De tal forma que no es necesario especificar el tipo de ventana(Camera,Image o vídeo).

Esta clase abstracta cuenta con los métodos que todas las demás deben tener para que no haya ningún problema a la hora de diferenciarlas como (documentadas en javadoc):
~~~~
public getCam()
public getShot()
public getImage()
public getLienzo()
~~~~

De tal explicación podemos decir que las clases hijas de "VentanaInternaSM.java" son: "InternalImageWindow.java","InternalvídeoWindow.java" y "InternalCameraWindow.java".

Explicaremos ahora cada una de ellas.

"InternalImageWindow"

Esta ventana interna es la encargada de mostrar las distintas figuras que pintemos y las imágenes que carguemos para modificar. Cuenta con los 4 métodos ya comentados arriba.

Es necesario explicar que nuestra InternalImageWindow contendrá un Lienzo, pero no controla nada relacionado con el vídeo o la webcam. Es por eso que aunque los cuatro métodos estén implementados, tan solo dos devuelven algo; quedará mejor explicado a continuación:

Estas son las funciones importantes referidas a la InternalImageWindow.

```
public Lienzo2DImagen getLienzo(){
        return this.lienzo2DImagen1;
    }

public BufferedImage getImage() {
            return this.getLienzo().getImage(true);
        }
```

Las siguientes funciones heredadas de su padre InternalWindowSM son obligatorias implementarlas, por ser, el padre, de tipo abstract. Aunque, no es necesario que tengan una funcionalidad como podemos comprobar aquí. Simplemente nos sirve para poder diferenciarla del resto de ventanas internas.

```
public EmbeddedMediaPlayer getShot(File f) {
        return null;
    }

public Webcam getCam() {
        return null;
    }
```

"InternalvídeoWindow"

Ventana interna encargada de la reproducción de vídeos según la extensión soportada por java (.avi y .mp4). Al igual que su hermana "InternalImageWindow" también contiene entre otros los métodos heredados de su padre(InternalWindowSM). En esta ocasión encontramos una implementación distinta pero similar a la de su hermana:

```
Esta sería la función importante de la ventana:
public EmbeddedMediaPlayer getShot(File f) {
        return vlcplayer;
    }
```

Y las siguientes referidas al buen funcionamiento entre todas las ventanas:

```
public Lienzo2DImagen getLienzo() {
        return null;
    }


public Webcam getCam() {
        return null;
    }

public BufferedImage getImage() {
        return null;
    }
```

"InternalCameraWindow"

Esta ventana interna por su parte, se ocupará del buen funcionamiento de la WebCam, cuando nos refiramos a la captura de la cámara, dicha funcionalidad queda relevada a la ventana interna de imágenes(InternalImageWindow), aún así es la propia cámara la que establece la imágen. Por supuesto, al igual que el resto de sus hermanas tiene implementada los cuatro métodos heredados de su padre:

```
Métodos importantes de la clase:
public BufferedImage getImage() {
        return camara.getImage();
    }

public Webcam getCam() {
        return camara;
    }
```

El resto de métodos referentes al buen funcionamiento entre ventanas:

```
public Lienzo2DImagen getLienzo() {
        return null;
    }

public EmbeddedMediaPlayer getShot(File f) {
        return null;
    }
```

Una vez explicadas las distintas clases de ventanas que nos encontraremos en nuestra aplicación, se explica todo lo referente al funcionamiento de esta: Gráficos, Imágenes, Sonido y vídeo.

2.2. "Gráficos e Imágenes"

2.2.1 Apartado Gráficos:

Para el apartado Gráficos, el cual se explicará por separado de "Imágenes", se presenta primeramente una idea global:

La idea principal es poder pintar en un "Lienzo" las distintas formas aquí presentes:

· Punto
· Línea
· Rectángulo
· Elipse
· Curva con un punto de control
· Trazo libre
· Forma propia

Y además la posibilidad de pintar:

· Una Diana

Para ello hemos dispuesto la clase "Lienzo2D" la cual se encargará de mantener en un vector de "MyShape" las distintas formas. Siendo así posible realizar distintas operaciones: establecer un grosor, un color(borde y relleno), relleno, transparencia y alisado.

Presentaremos las distintas formas explicando y justificando el por qué se ha realizado así:

2.2.1.1 Jerarquía de Formas

Primeramente debemos explicar la jerarquía elegida y por qué esa. Nuestra jerarquía consta de una primera clase padre "MyShape.java" de la que heredaran **dos** subclases "WithFilled.java" y "WithOutFilled.java".

"MyShape.java" contiene los atributos comunes para todas las formas:

·Stroke(grosor)
·Color
·Composite(transparencia)
·RenderingHints(Alisado)

Además contiene las funciones para poder modificar estos atributos las cuales heredarán las distintas formas.
Por último añadir que también contiene cierta funcionalidad como: "saber si el objeto está seleccionado" en el modo "Editar" o "informar" al objeto que lo está.

"WithFilled.java y WithOutFilled.java" son 2 simples subclases que sirven más bien para clasificar nuestras formas en "Con Relleno" o "Sin Relleno" evitando así accesos a este atributo si no puedes tenerlo, como es el caso de la línea, el punto o la curva con un punto de control.

2.2.1.2 Tipos de Formas

· Línea y Punto

Estas dos formas comparten la misma clase, "MyLine.java", debido a que la única diferencia entre una línea y un punto, es que la "línea" se pinta mediante un **punto Inicial** distinto del **punto Final**, y el "punto" mediante un un punto Inicial **igual** a un punto final.

Hereda directamente de la clase "WithOutFilled" ya que no tiene relleno posible.

Métodos comentados en JavaDoc.

-Manual:

·Punto:
Click en "Nuevo" -> Seleccionar la herramiento "Punto" -> Click en el lienzo.
(Pudiendo variar de color,grosor,etc. mediante los distintos atributos de
la barra de herramientas superior).
·Línea:
Click en "Nuevo" -> Seleccionar la herramiento "Linea" -> Click en el lienzo y
arrastra hasta donde quieras llevar la línea.
(Pudiendo variar de color,grosor,etc. mediante los distintos atributos de
la barra de herramientas superior).

· Rectángulo

Definida en la clase "MyRectangle.java" se ha decidido que el rectángulo se pintará a través de 2 puntos, el primero será la esquina superior izquierda y el segundo la esquina inferior derecha, mediante la función "setFrameDiagonal" propio de la clase Rectangle2D.

Hereda directamente de la clase "WithFilled" ya que tiene relleno.

Métodos comentados en JavaDoc.

-Manual:

```
·Rectángulo:
Click en "Nuevo" -> Seleccionar la herramiento "Rectángulo" -> Click en el lienzo
y arrastra hasta donde quieras llevar el rectángulo.
(Pudiendo variar de color,grosor,relleno,etc. mediante los distintos atributos de
  la barra de herramientas superior).
```

· Elipse

Definida en la clase "MyEllipse.java" similar al rectángulo, se ha decidido pintar la elipse a partir de dos puntos, y utilizando también el método setFrameDiagonal de la clase Ellipse2D, que nos permite encajar en un rectángulo imaginario formado por esos dos puntos(el primero esquina superior izquierda y el segundo esquina inferior derecha) y encajando la elipse en dicho rectángulo.

Hereda de la clase "WithFilled" por lo que tiene relleno.

Métodos comentados en JavaDoc.

-Manual:

```
·Elipse:
Click en "Nuevo" -> Seleccionar la herramiento "Elipse" -> Click en el lienzo
y arrastra hasta donde quieras llevar la elipse.
(Pudiendo variar de color,grosor,relleno,etc. mediante los distintos atributos de
  la barra de herramientas superior).
```

· Curva con un punto de control

Definida en la clase "MyQuadCurve.java", en este caso hemos utilizado una línea para pintar la primera parte(definida mediante un punto inicial y otro final) y, mediante un punto dado, después de estar ya pintada la línea, la curva hacia ese punto.

Hereda de la clase "WithOutFilled" por lo que no tiene relleno.

Métodos comentados en JavaDoc.

-Manual:

·Curva con un punto de control
Click en "Nuevo" -> Seleccionar la herramiento "Curva con un punto de control" -> Click en el lienzo y arrastra hasta donde quieras llevar la linea -> click en otro punto y la línea se curvará hacia él.
(Pudiendo variar de color, grosor, relleno, etc. mediante los distintos atributos de la barra de herramientas superior).

· Espiral

La encontramos definida en la clase "MySpiral.java". Primero aclarar que para poder obtener un resultado similar a una diana, debemos tener seleccionado el atributo de trazo: “discontínuo2”. La explicación de cómo hemos obtenido esta figura es: Debido a que a diferencia de el trazo libre, que solo se realiza una vez el punto de inicio cuando hacemos el pressed del ratón, en la Espiral se establece siempre como “punto1” (el pressed) el primer punto, que será el centro de nuestra espiral o diana, y el “punto2” (que será el sitio donde arrastramos el ratón), de tal forma que lo que está pasando es que se están pintando líneas rectas de un punto central hacia afuera(el “punto2”).

Hereda de la clase "WithOutFilled" por lo que no tiene relleno.

Métodos comentados en JavaDoc.

-Manual:

·Espiral
Click en "Nuevo" -> Seleccionar la herramiento "Espiral" ->
Click en el lienzo y arrastra hasta donde quieras llevar la línea (movimiento
circular alrededor del punto para observar el resultado).
(Pudiendo variar de color, grosor, etc. mediante los distintos
atributos de la barra de herramientas superior).
NOTA: Se observa mejor el resultado si tenemos seleccionado el trazo:”Discontínuo2”.

· Trazado libre

Queda definido en la clase "FreeDraw.java". Consta de un objeto General Path en el cual se insertarán los puntos que vamos añadiendo a 2 vectores de coordenadas.Uno utilizado para almacenar las **coordenadas X** y otro para almacenar las **coordenadas Y**. Tal que, se establece el primer punto inicial(setFirstPointGP -> moveTo) y se van concatenando el resto a medida que se añadan(setGP -> lineTo).

Hereda de la clase "WithOutFilled" por lo que no tiene relleno.

Métodos comentados en JavaDoc.

-Manual:

·Trazado libre
Click en "Nuevo" -> Seleccionar la herramiento "Trazado libre" ->
Click en el lienzo y arrastra hasta donde quieras llevar el trazo.
(Pudiendo variar de color, grosor, etc. mediante los distintos
atributos de la barra de herramientas superior).

· Mi Forma

Contenida en la clase "MyForm.java". Se trata de poder realizar cualquier tipo de forma, sin importar cuantos lados tenga. Solo hay que ir seleccionando donde se deben colocar los distintos vértices.

Hereda de la clase "WithFilled" por lo que tiene relleno.

Métodos comentados en JavaDoc.

- Manual:

·Mi forma:
Click en "Nuevo" -> Seleccionar la herramiento "Mi Forma" ->
Click izquierdo para establecer el primer punto de control.
Click derecho para establecer el resto de vértices, tantos clicks en los distintos sitios como queramos.
Click izquierdo para cerrar la forma.
(Pudiendo variar de color, grosor, etc. mediante los distintos atributos de la barra de herramientas superior).

Por último añadir, ya que no estaba seguro de que “Mi Forma” pudiese ser aceptado como forma propia, añadí la forma espiral.

2.2.2 Apartado Imágenes:

En esta sección se explicará qué funcionalidad tiene nuestra aplicación para el apartado de imágenes.

Primero aclarar qué tipos de operaciones actuarán sobre una imagen, pudiendo diferenciarlas por el trato sobre ésta:

· El brillo

Sobre un slider aumenta o disminuye la luz de la imagen. Se ha utilizado para ello la clase RescaleOP que mediante un factor de escala y un desplazamiento se logra obtener ese efecto.
Para lograr que la imagen fuente no se vea afectada por el filtro, no se modifica la imagen fuente y se trabaja sobre una copia de ésta internamente, siendo la imagen fuente reemplazada por la copia una vez soltemos el slider.

· Filtros

Los distintos filtros vistos en clase funcionan a través de un objeto Kernel, utilizado en la clase ConvolveOP. El funcionamiento de este operador viene dado por una convolución espacial, esto quiere decir que el color de cada píxel de la imagen destino viene determinado por una combinación de colores del píxel fuente y de los píxeles vecinos.

En nuestro caso tenemos **5 tipos de filtros** en este apartado:

·Media: aplica un emborronamiento a la imágen.
·Binomial: efecto similar al “Media”.
·Enfoque: deshace el emborronamiento de una foto.
·Relieve: realza la imagen.
·Laplaciano: actúa sobre una imagen mostrando las fronteras.

·Contrastes

Para estas operaciones utilizamos un objeto de tipo LookUpTable para usarlo en la clase LookUpOp, la cual permite definir una transformación sobre los niveles de color mediante el objeto LookUpTable, tal que para cada componente de cada pixel se aplica la transformación.

Los **4 filtros** que podemos ver aquí son:

·Contraste : aplica un efecto contraste a la imagen.
·Negativo : en el caso del negativo picado a mano, se crea una tabla de transformación,la cual recorre un vector de Bytes de 256 elementos, a los cuales les aplica el valor de la resta de 255 menos la posición en el array y aplica la transformación a una imágen.
·Iluminación : ilumina la imagen.
·Oscuridad : oscurece la imagen.

· Operaciones

En este apartado encontramos distintos operadores:

· Operador Sin: en un vector de bytes de 256 elementos multiplica el valor 255.0 por el valor del seno de 180/255 por cada uno de los bytes y se queda con el valor absoluto de esta operación, para después aplicar la transformación mediante un objeto LookUpTable.

· Sepia: Recorre los pixeles de una imagen y por cada banda de color(RGB) aplica el filtro, estableciendo como máximo valor posible 255, ya que no es posible la representación de un valor superior.

· Tinta: Esta operación tiñe la imagen según el color que tengamos seleccionado. Aplicando para cada píxel una transformación

· Ecualizador: trata de obtener un histograma lo más “plano” posible, mejorando con ello el contraste.

· Color

El primer botón obtiene por separado las 3 bandas de una imagen RGB, podemos observar que al modificar cualquiera de ellas, la imagen se verá afectada.
En el segundo se nos permite obtener las bandas de una imagen en escala de grises o las bandas YCC.

·Rotación

Permite girar la figura sobre sí misma, ya sea de forma libre con un slider o en grados (90º,180º,270º). Se aplica una transformación en base a un objeto AffineTransform, en el que imagen fuente y destino deben de ser distintas,la cual mantiene los puntos , líneas rectas y planos. Además, los conjuntos de líneas paralelas permanecerán así después de la transformación

·Escalado

Estas operaciones permiten hacer zoom tanto en **aumento** como en **disminución** de una imagen. Se aplica una transformación en base a objetos AffineTansfrom, al igual que en la “Rotación”.

·Umbralización

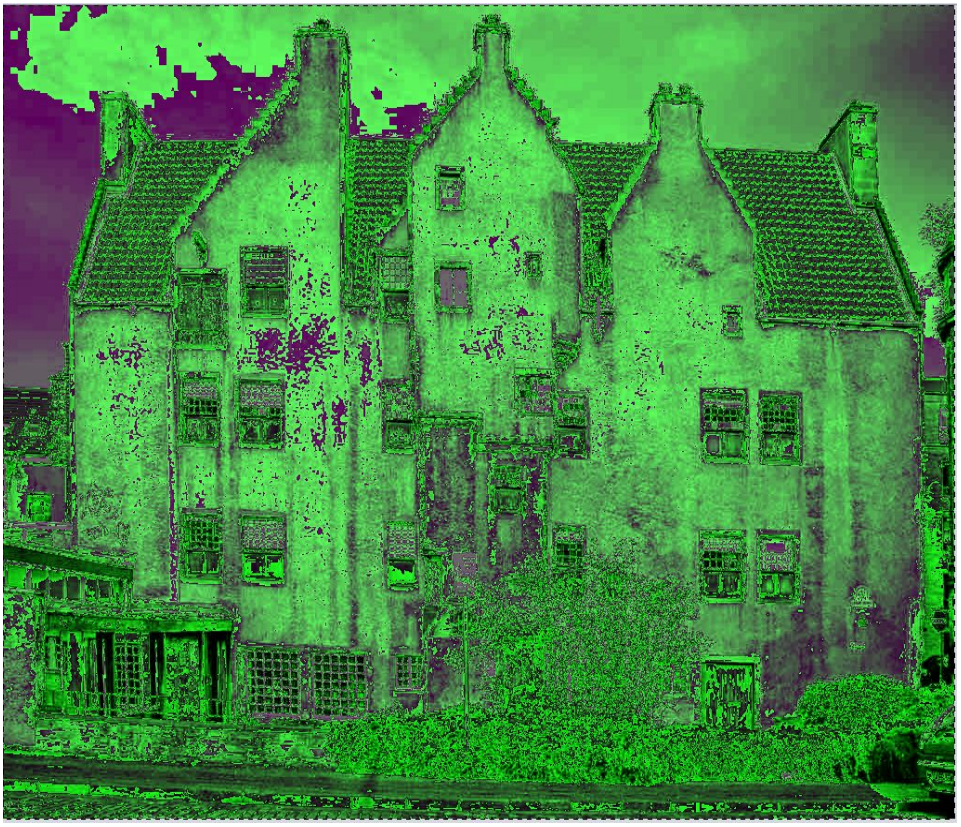
Esta operación está basada en pixeles, usando las propiedades de éstos(color,textura) para agruparlos sin considerar conectividad espacial. Genera una imagen binaria en la que según una condición, a cada píxel se le asigna uno o cero.

·Mis propios filtros:

· Número áureo: A partir de una transformación definida mediante un objeto de la clase LookUpTable,la cual actúa sobre cada componente de cada píxel.

Lo que se realiza en esta operación es una multiplicación del número áureo por una serie de números(del 1 al 255) y los almacena en un vector de Bytes para crear una tabla de transformación para aplicarse posteriormente a la imágen.

· Número PI : Pixel a pixel , realiza un recorrido de la imagen fuente y multiplica cada valor de su R,G y B por una matriz definida(3x3 , la cual mantiene en el centro en número PI y a los alrededores los decimales de éste).Mostrando un efecto tenebroso al realizarse.



· Filtro Fantasma: Componente a componente. Se ha utilizado para la implementación de este operador la clase RescaleOP que mediante un factor de escala y un desplazamiento logra obtener ese efecto.
Al utilizarse aclara la foto hasta el punto de hacerla similar a un tono fantasmal.

2.2.3 Sonido

Respecto al apartado de sonido, en relación a lo dado en clase nuestro objetivo es permitir la **reproducción** y la **captación**(grabación) de sonidos, usando la Sound API de java.

Para ello partimos de una lista de reproducciones inicialmente vacía, la cual se llenará a partir de archivos con extensión .au y .wav(a las que da soporte la API) de dos formas:

1. Abriendo un archivo de estas características.
2. Grabando un audio mediante el micrófono del ordenador.

A la hora de reproducirlo tenemos un botón “Play” el cual nos facilita dicha reproducción. Por otro lado, tenemos un botón “Stop” el cual pausa la reproducción o la reinicia hasta el punto de inicio.

Para el buen funcionamiento de estas dos funciones hay dos manejadores, uno para la reproducción y otro para la grabación, que se encargan de que no exista ningún tipo de conflicto a la hora de reproducir o grabar un audio.

2.2.4 Reproducción y captura de vídeo

Por último en esta sección, comentamos las distintas posibilidades que tenemos, como encender la webcam y tomar una instantánea de esta.

Mediante unas bibliotecas facilitadas por nuestro profesor, nuestro programa es capaz de reproducir vídeos en formato .avi y .mp4. Esto es posible mediante la apertura de un vídeo que cumpla con los requisitos de las extensiones, el cual se abrirá en una ventana interna propia que cuenta con sus propios botones “Play” y “Pause” que cuentan con el mismo funcionamiento que los de **Sonido**, es decir, reproduzca un vídeo, lo pause, y si ya está pausado, lo puede reiniciar pulsando de nuevo el botón “Stop”.

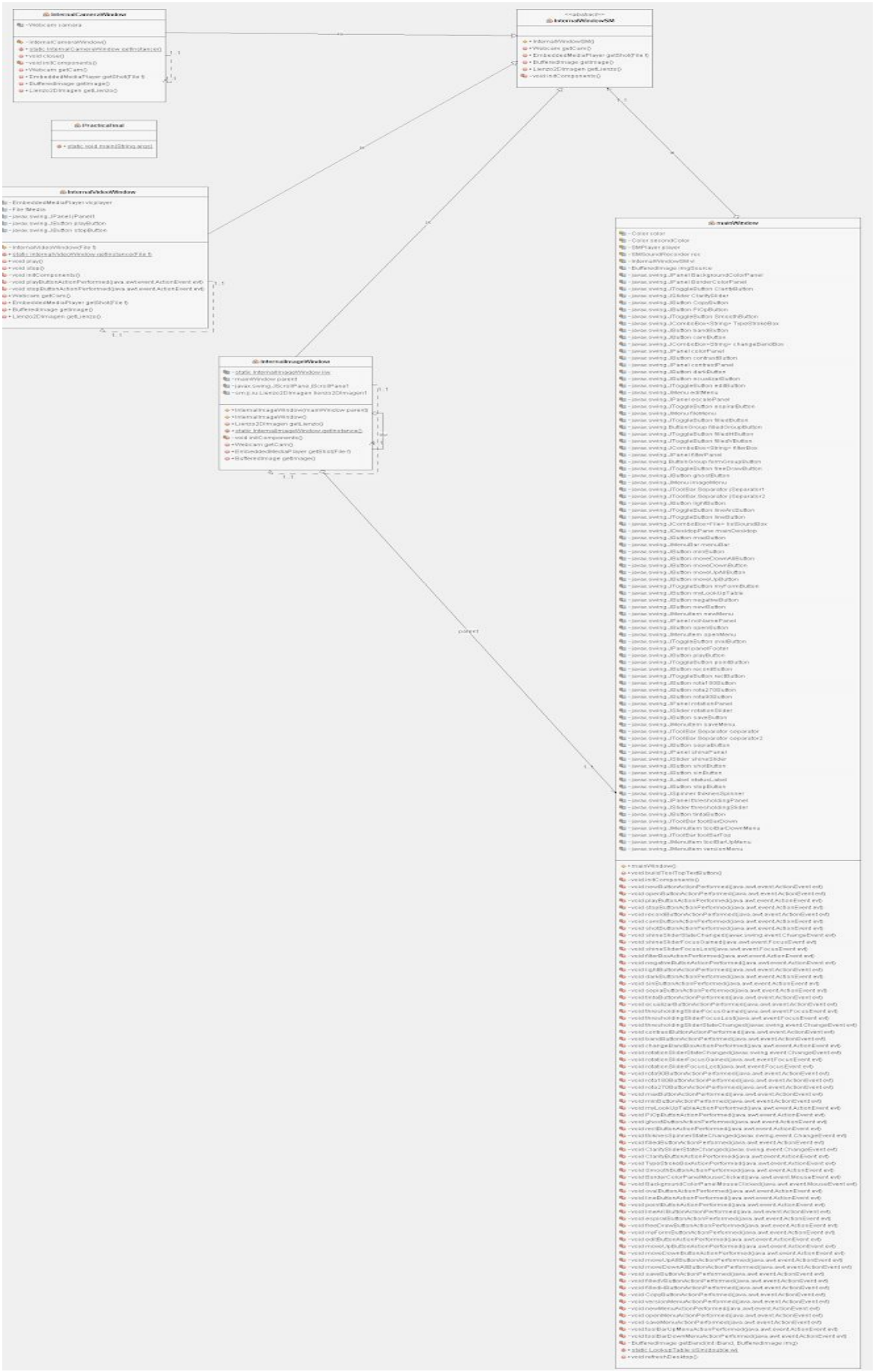
Por otro lado, y como se comenta al principio, nuestra aplicación es capaz de habilitar la WebCam de nuestro ordenador y realizar capturas de dicha filmación.

Para finalizar, es necesario añadir que dichas capturas obtenidas de la WebCam, tienen la posibilidad de ser modificadas por cada uno de los filtros con los que cuenta nuestra aplicación.

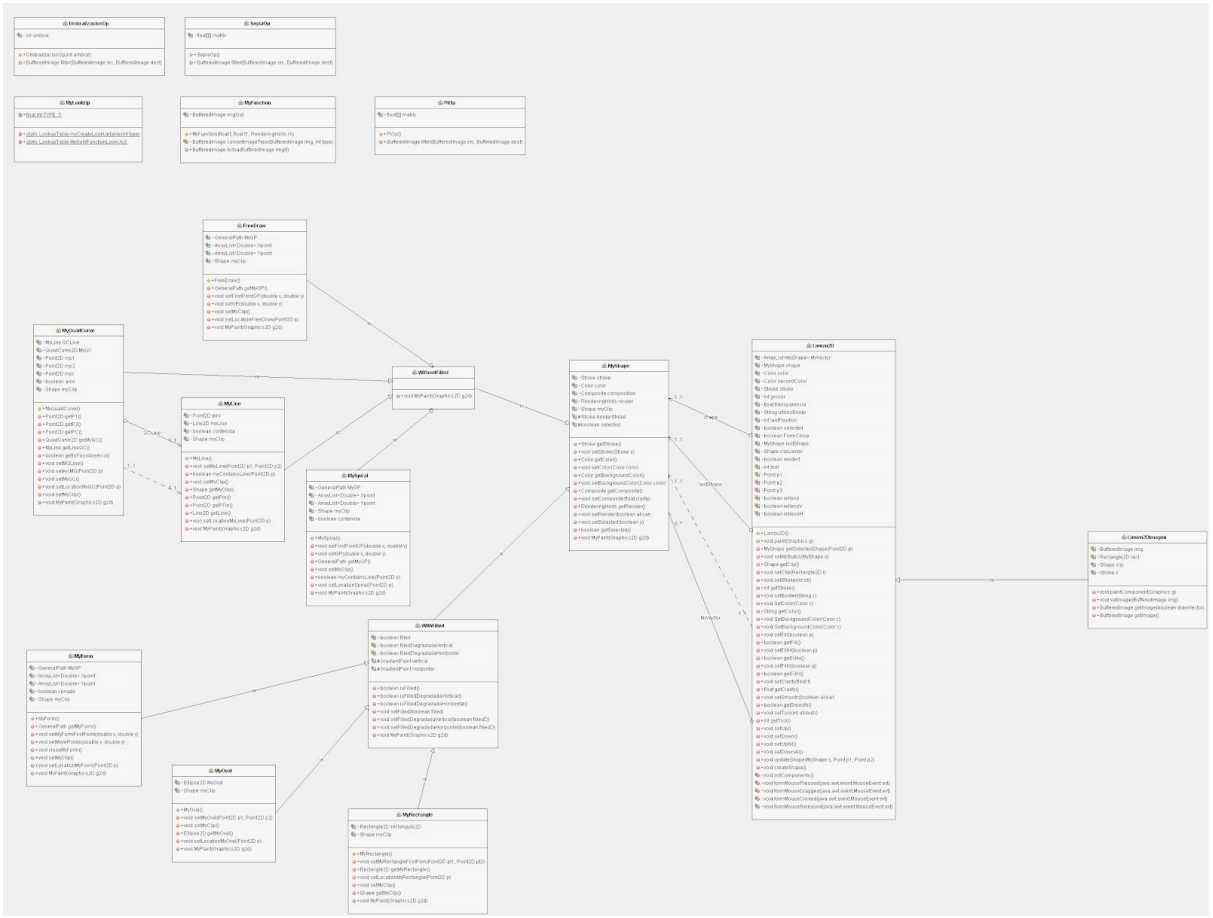
3. Diagrama de clases

Se insertarán aquí los diagramas de clases tanto del proyecto como de la propia biblioteca para facilitar la lectura de estos.(Se adjuntan por separado para su mayor comodidad).

·**Diagrama de Clases del Proyecto**



·Diagrama de Clases de la Biblioteca.SM.JJ


UmbralizacionOp
SepiaOp
MyLookUp
MyFunction
PiOp
FreeDraw
MyQuadCurve
MyLine
MyOval
MyRectangle
MyForm
WithoutFilled
WithFilled
MyShape
Lienzo2D
Lienzo3Dimagen

4. Bibliografía

Entre otros foros y enlaces interesantes los realmente influyentes a la hora de realizar y documentar la práctica son:

· https://docs.oracle.com/javase/7/docs/api/javax/swing/JColorChooser.html
(Documentación propia de oracle, gran recomendación, facilita trabajo con la elección de colores, permitiendo elegir entre una gran cantidad de colores sobre una paleta propia)

· “Java a tope: Java2D. Cómo tratar con Java figuras, imágenes y texto en dos dimensiones”.
Libro de Sergio Gálvez Rojas, Manuel Alcaide García, Miguel Ángel Mora Mata
https://books.google.es/books?id=M6reV4TGylQC&pg=PA107&lpg=PA107&dq=como+actúa+rescaleOP&source=bl&ots=jfqMpd1tyT&sig=fk_YBznev_EKue7_WRpwmaYQh5k&hl=es&sa=X&ved=0ahUKEwju8-eDg4ncAhVGGsAKHSKMAdsQ6AEIMTAB#v=snippet&q=rescale&f=false
(Facilidades a la hora de entender el funcionamiento de ciertos operadores de imágenes)

·”Documentación de PDF´s por parte del profesor: Jesús Chamorro Martínez”
(Para todos los campos, en especial imágenes,vídeo y sonido).