Ejercicio 34

Desde la versión de java 8.0 el atributo final a la hora de declarar una variable está deprecated, esto antes se utilizaba para poder acceder desde una variable anonima para poder facilitar el acceso a las variables desde las clases anonimas sin tener que crear getters o variables staticas.

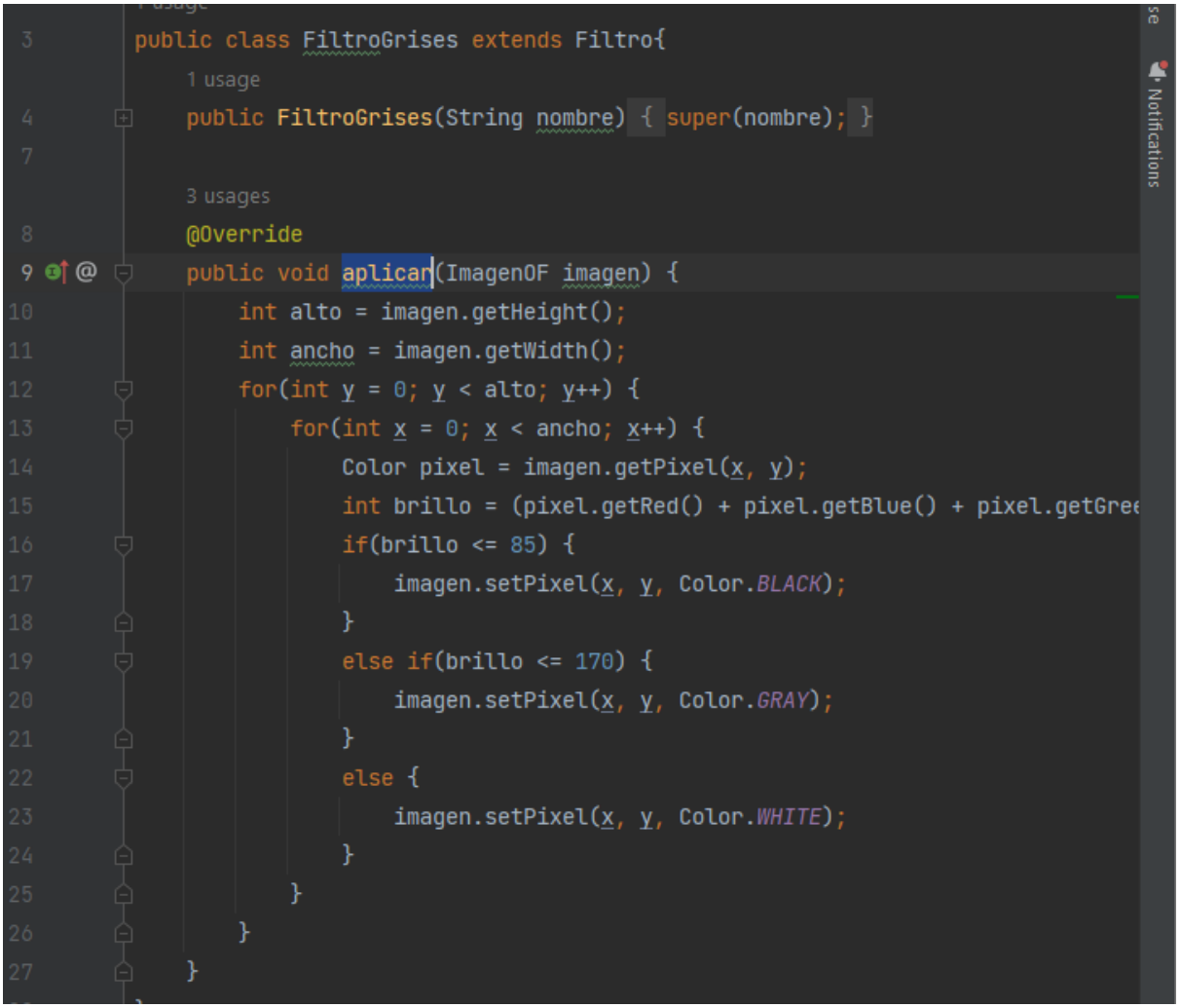
Ejercicio 35

Necesito crear una nueva clase que extienda de filtro y rehacer la función “aplicar” con un @override , además deberemos de añadirla a la lista de filtros en la clase VisorDeImagen.

_Ej. FOTO:

```
private List<Filtro> crearFiltros() {
    List<Filtro> listaDeFiltros = new ArrayList<Filtro>();
    listaDeFiltros.add(new FiltroOscuro( nombre: "Oscuro"));
    listaDeFiltros.add(new FiltroOscuro( nombre: "Claro"));
    listaDeFiltros.add(new FiltroGrises( nombre: "Umbral"));
    listaDeFiltros.add(new FiltroEspejo( nombre: "Espejo"));
    listaDeFiltros.add(new FiltroInvertir( nombre: "Invertir"));
    listaDeFiltros.add(new FiltroAlisar( nombre: "Alisar"));
    listaDeFiltros.add(new FiltroSolarizar( nombre: "Solarizar"));
    listaDeFiltros.add(new FiltroBordes( nombre: "Bordes"));
    listaDeFiltros.add(new FiltroOjoPez( nombre: "Ojo de Pez"));

    return listaDeFiltros;
}
```

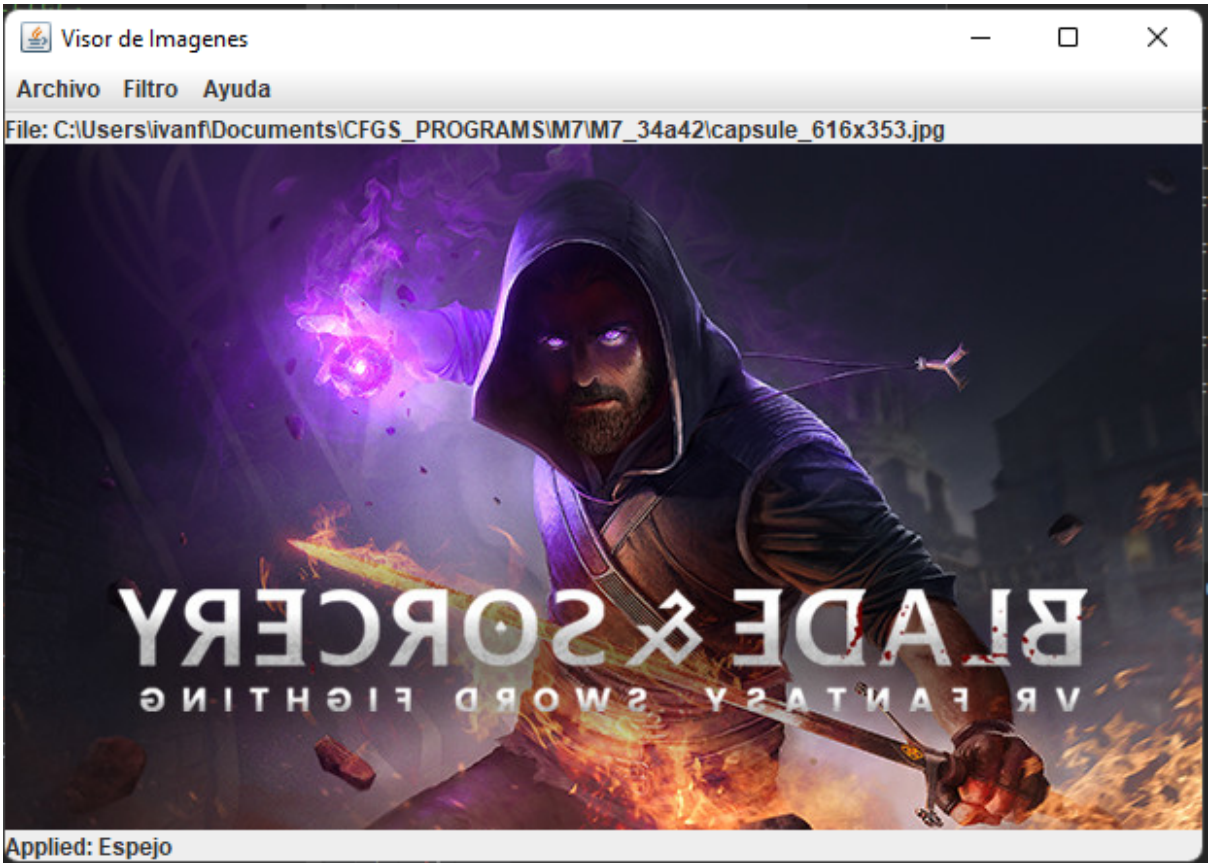
Ejercicio 36



```java
public class FiltroGrises extends Filtro{
    1 usage
    public FiltroGrises(String nombre) { super(nombre); }

    3 usages
    @Override
    public void aplicar(ImagenOF imagen) {
        int alto = imagen.getHeight();
        int ancho = imagen.getWidth();
        for(int y = 0; y < alto; y++) {
            for(int x = 0; x < ancho; x++) {
                Color pixel = imagen.getPixel(x, y);
                int brillo = (pixel.getRed() + pixel.getBlue() + pixel.getGree
                if(brillo <= 85) {
                    imagen.setPixel(x, y, Color.BLACK);
                }
                else if(brillo <= 170) {
                    imagen.setPixel(x, y, Color.GRAY);
                }
                else {
                    imagen.setPixel(x, y, Color.WHITE);
                }
            }
        }
    }
```

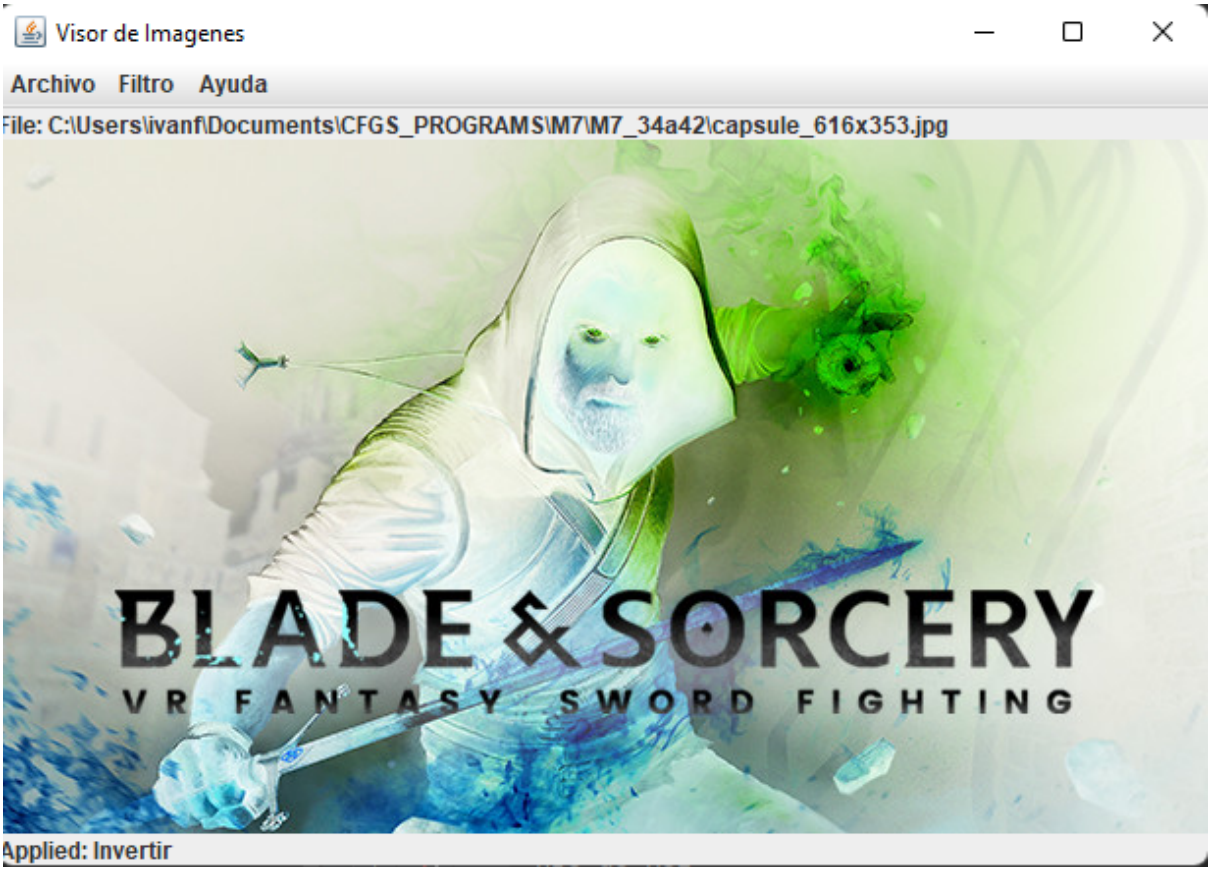
Ejercicio 37

```java
import java.awt.*;

1 usage
public class FiltroEspejo extends Filtro{

    1 usage
    public FiltroEspejo(String nombre) { super(nombre); }

    4 usages
    public void aplicar(ImagenOF image)
    {
        int height = image.getHeight();
        int width = image.getWidth();
        for(int y = 0; y < height; y++) {
            for(int x = 0; x < width / 2; x++) {
                Color left = image.getPixel(x, y);
                image.setPixel(x, y, image.getPixel( x: width - 1 - x, y));
                image.setPixel( x: width - 1 - x, y, left);
            }
        }
    }
}
```



Ejercicio 38

```java
import java.awt.*;

public class FiltroInvertir extends Filtro{

    public FiltroInvertir(String nombre) { super(nombre); }

    4 usages
    public void aplicar(ImagenOF image)
    {
        int height = image.getHeight();
        int width = image.getWidth();
        for(int y = 0; y < height; y++) {
            for(int x = 0; x < width; x++) {
                Color pix = image.getPixel(x, y);
                image.setPixel(x, y, new Color( r: 255 - pix.getRed(),
                        g: 255 - pix.getGreen(),
                        b: 255 - pix.getBlue()));
            }
        }
    }
}
```

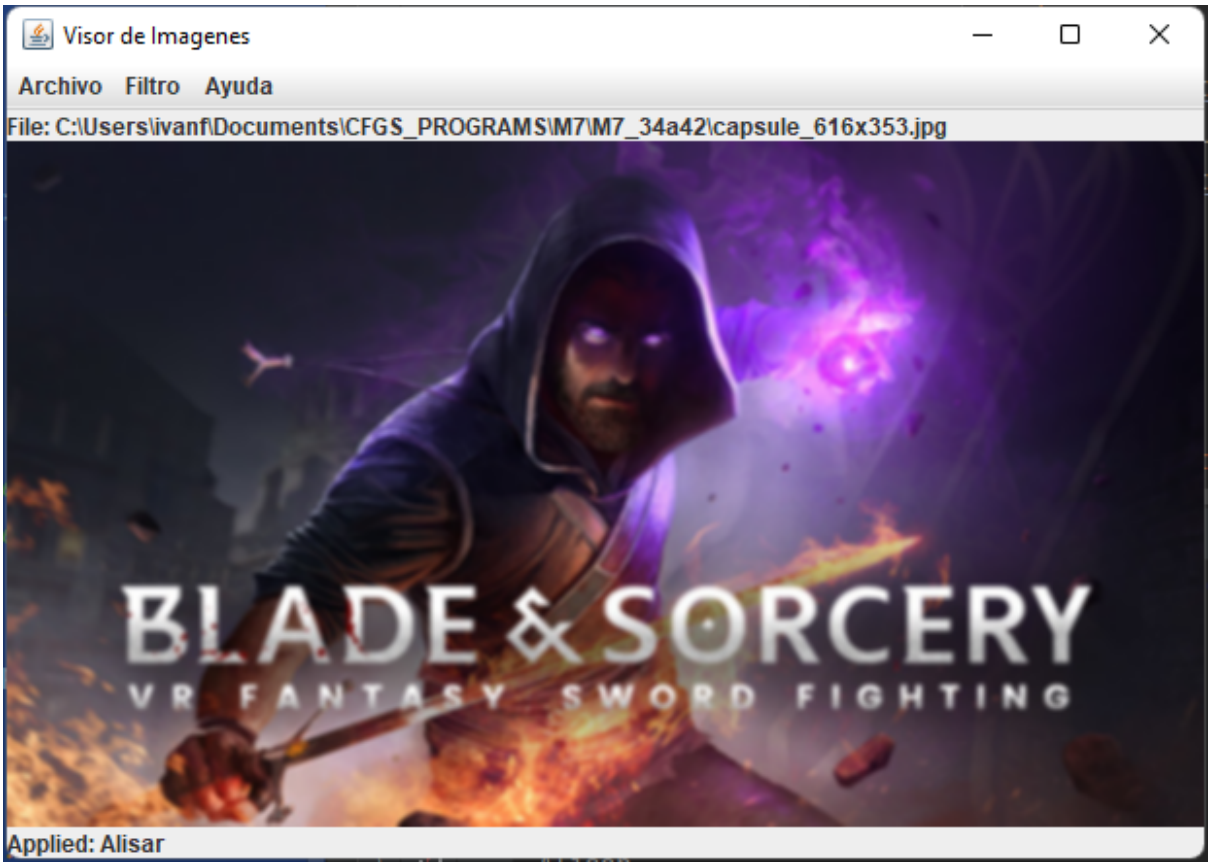


Ejercicio 39

```java
import java.awt.*;
import java.util.ArrayList;
import java.util.List;

1 usage
public class FiltroAlisar extends Filtro {

    1 usage
    public FiltroAlisar(String nombre) { super(nombre); }

    4 usages
    private ImagenOF original;
    3 usages
    private int width;
    3 usages
    private int height;

    4 usages
    public void aplicar(ImagenOF image)
    {
        original = new ImagenOF(image);
        width = original.getWidth();
        height = original.getHeight();

        for(int y = 0; y < height; y++) {
            for(int x = 0; x < width; x++) {
                image.setPixel(x, y, smooth(x, y));
            }
        }
    }

    1 usage
```

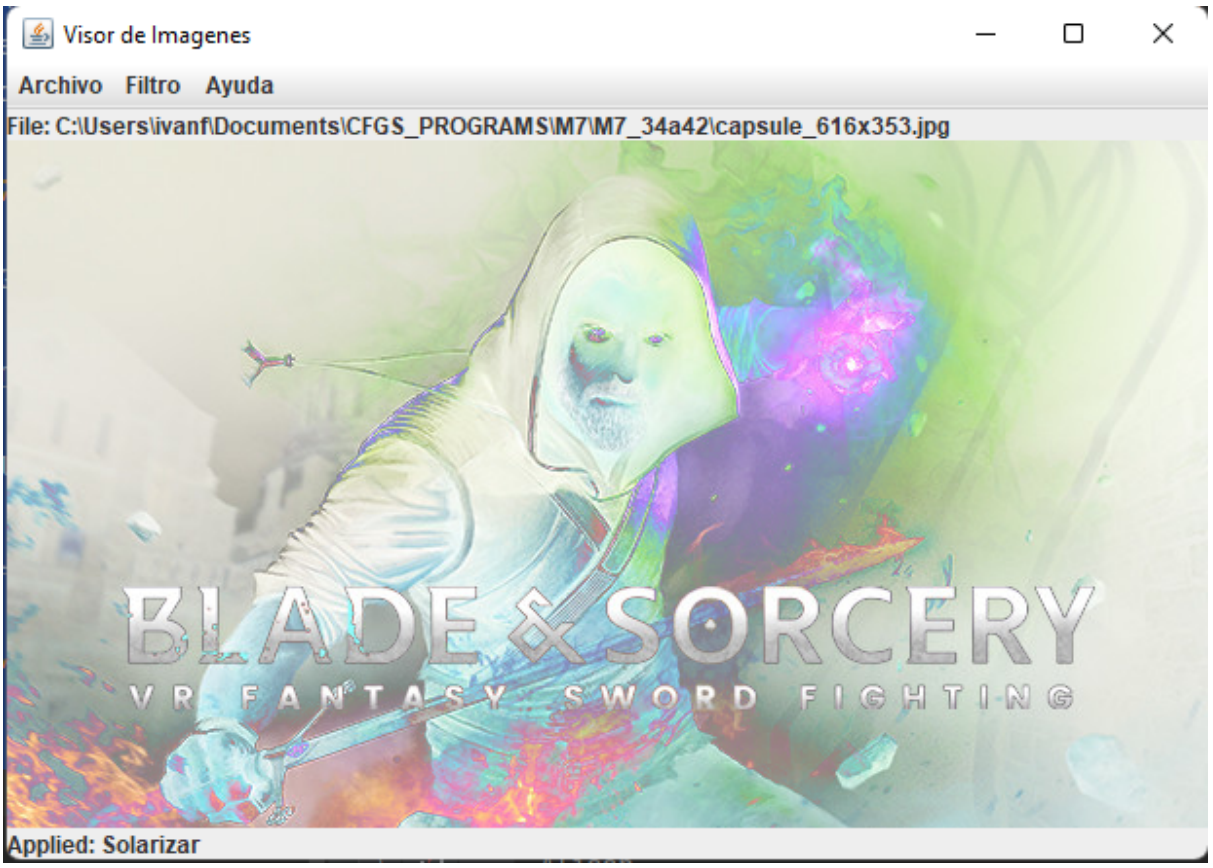
```java
private Color smooth(int xpos, int ypos)
{
    java.util.List<Color> pixels = new ArrayList<~>( initialCapacity: 9);

    for(int y = ypos - 1; y <= ypos + 1; y++) {
        for(int x = xpos - 1; x <= xpos + 1; x++) {
            if( x >= 0 && x < width && y >= 0 && y < height )
                pixels.add(original.getPixel(x, y));
        }
    }

    return new Color(avgRed(pixels), avgGreen(pixels), avgBlue(pixels));
}

1 usage
private int avgRed(java.util.List<Color> pixels)
{
    int total = 0;
    for(Color color : pixels) {
        total += color.getRed();
    }
    return total / pixels.size();
}

1 usage
private int avgGreen(java.util.List<Color> pixels)
{
    int total = 0;
    for(Color color : pixels) {
        total += color.getGreen();
    }
    return total / pixels.size();
}
```

```java
1 usage
private int avgBlue(List<Color> pixels)
{
    int total = 0;
    for(Color color : pixels) {
        total += color.getBlue();
    }
    return total / pixels.size();
}
```



Ejercicio 40

```java
import java.awt.*;

1 usage
public class FiltroSolarizar extends Filtro{

    1 usage
    public FiltroSolarizar(String nombre) { super(nombre); }

    public void aplicar(ImagenOF image)
    {
        int height = image.getHeight();
        int width = image.getWidth();
        for(int y = 0; y < height; y++) {
            for(int x = 0; x < width; x++) {
                Color pix = image.getPixel(x, y);
                int red = pix.getRed();
                if(red <= 127) {
                    red = 255 - red;
                }
                int green = pix.getGreen();
                if(green <= 127) {
                    green = 255 - green;
                }
                int blue = pix.getBlue();
                if(blue <= 127) {
                    blue = 255 - blue;
                }
                image.setPixel(x, y, new Color(red, green, blue));
            }
        }
    }

}
```

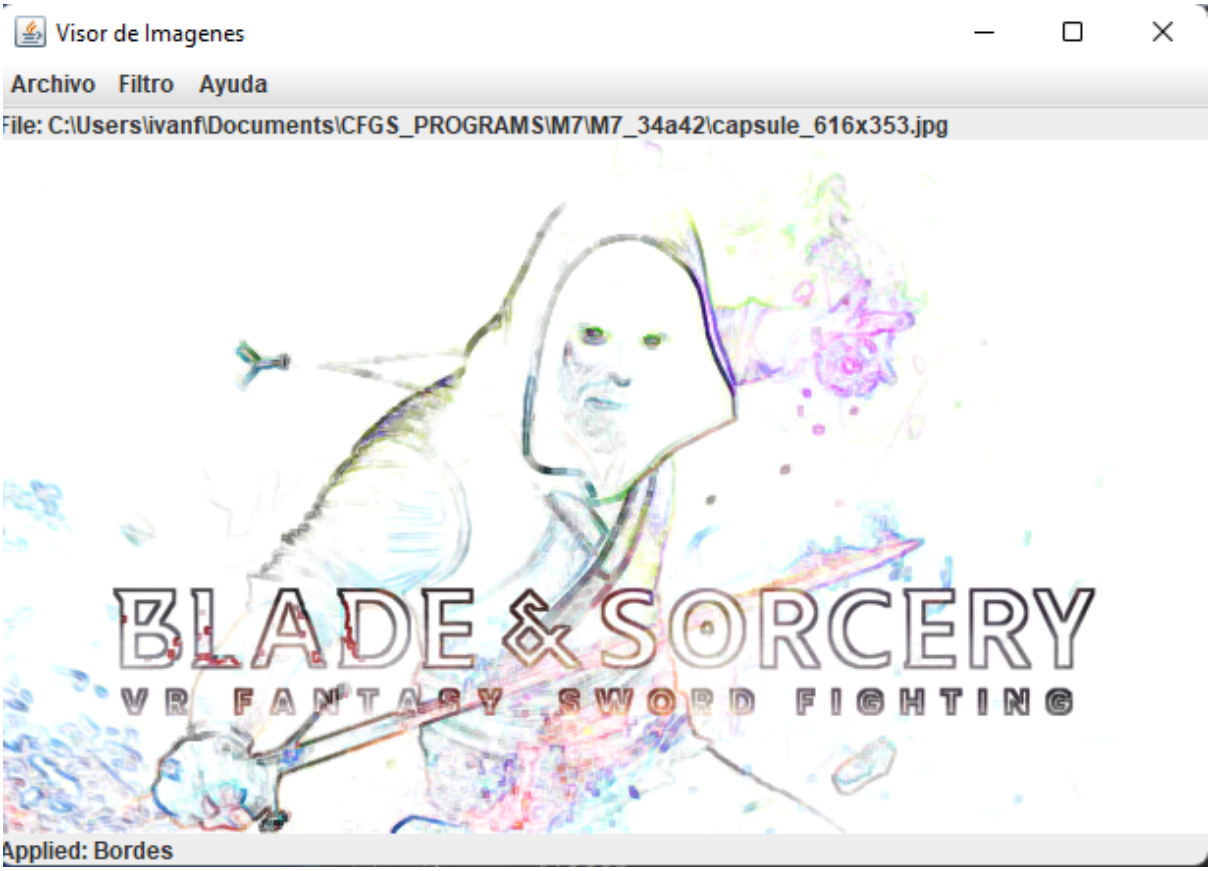


Ejercicio 41

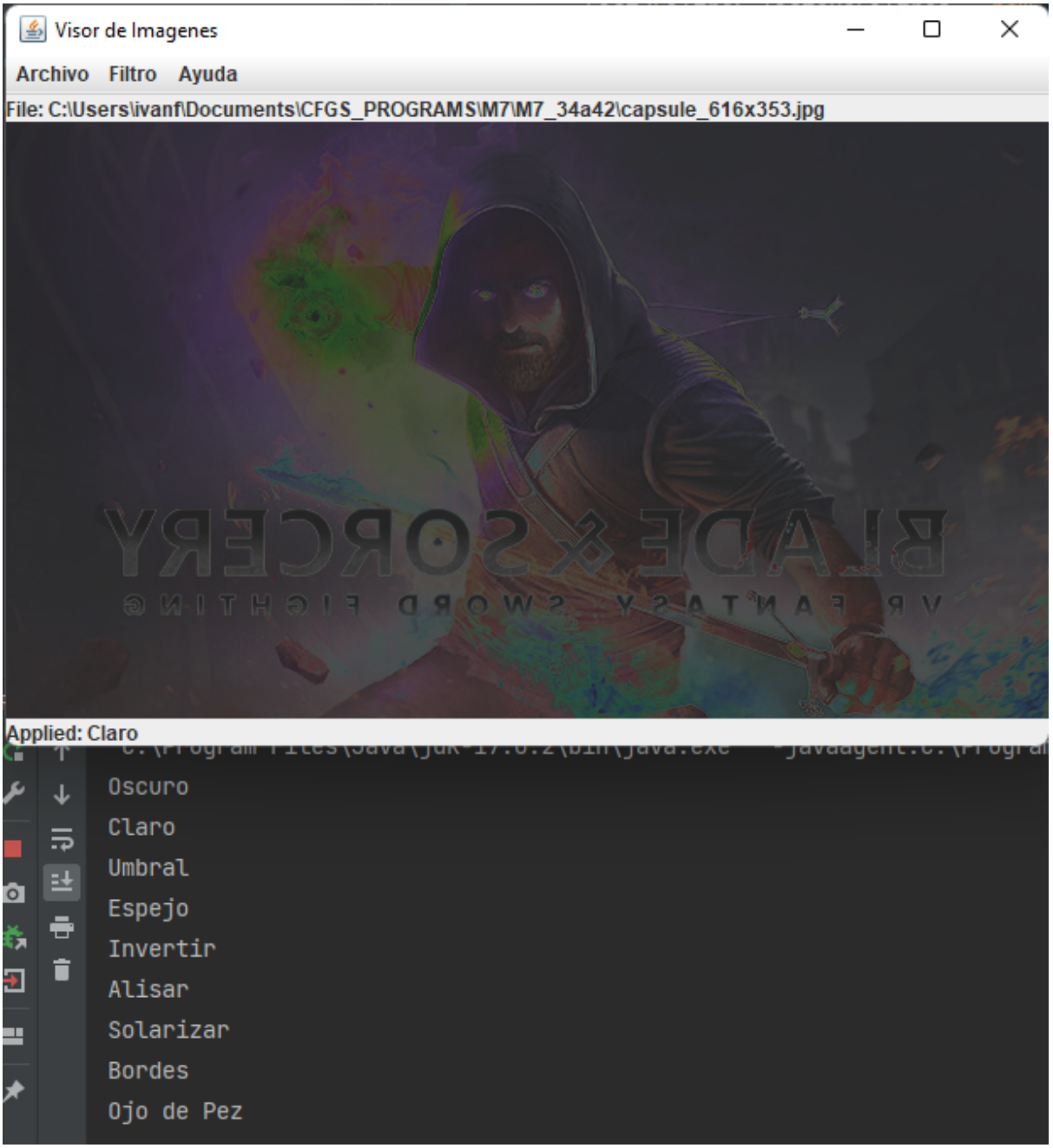
```java
import java.awt.*;
import java.util.ArrayList;
import java.util.List;

public class FiltroBordes extends Filtro{
    3 usages
    private static final int TOLERANCE = 20;

    4 usages
    private ImagenOF original;
    3 usages
    private int width;
    3 usages
    private int height;

    public FiltroBordes(String nombre) { super(nombre); }

    public void aplicar(ImagenOF image)
    {
        original = new ImagenOF(image);
        width = original.getWidth();
        height = original.getHeight();

        for(int y = 0; y < height; y++) {
            for(int x = 0; x < width; x++) {
                image.setPixel(x, y, edge(x, y));
            }
        }
    }
```

```java
private Color edge(int xpos, int ypos)
{
    java.util.List<Color> pixels = new ArrayList<~>( initialCapacity: 9);

    for(int y = ypos-1; y <= ypos+1; y++) {
        for(int x = xpos-1; x <= xpos+1; x++) {
            if( x >= 0 && x < width && y >= 0 && y < height ) {
                pixels.add(original.getPixel(x, y));
            }
        }
    }

    return new Color( r: 255 - diffRed(pixels),  g: 255 - diffGreen(pixels)
}

1 usage
private int diffRed(java.util.List<Color> pixels)
{
    int max = 0;
    int min = 255;
    for(Color color : pixels) {
        int val = color.getRed();
        if(val > max) {
            max = val;
        }
        if(val < min) {
            min = val;
        }
    }
    int difference = max - min - TOLERANCE;
    if(difference < 0) {
        difference = 0;
    }
    return difference;
}
```

```
{                                                    ⚠3 ⚠1 ✓6 ^ v
    int max = 0;
    int min = 255;
    for(Color color : pixels) {
        int val = color.getGreen();
        if(val > max) {
            max = val;
        }
        if(val < min) {
            min = val;
        }
    }
    int difference = max - min - TOLERANCE;
    if(difference < 0) {
        difference = 0;
    }
    return difference;
}

1 usage
private int diffBlue(List<Color> pixels)
{
    int max = 0;
    int min = 255;
    for(Color color : pixels) {
        int val = color.getBlue();
        if(val > max) {
            max = val;
        }
        if(val < min) {
            min = val;
        }
    }
    int difference = max - min - TOLERANCE;
    if(difference < 0) {
        difference = 0;
    }
    return difference;
```



Ejercicio 42


Visor de Imagenes
Archivo Filtro Ayuda
File: C:\Users\ivanf\Documents\CFGS_PROGRAMS\M7\M7_34a42\capsule_616x353.jpg
Applied: Claro
Oscuro
Claro
Umbral
Espejo
Invertir
Alisar
Solarizar
Bordes
Ojo de Pez