```java
package efectos;

import imagenes.ColorRGB;

/**
 * Esta clase permite dibujar un marco a la imagen, con el color y grosor indicados.
 * @author David Jimenez Diaz-Pintado
 *
 */
public class EfectoMarco extends Efecto{

    private ColorRGB colorMarco;
    private int anchoMarco;

    /**
     * Construye un objeto que permitirá dibujar un marco a la imagen.
     *
     * @param nombre Nombre del efecto.
     * @param imagen Imagen a procesar.
     * @param ancho  Anchura del marco en píxeles. Como máximo debe ser la mitad del menor valor entre la altura y la anchura de la imagen. Si se supera este valor, el ancho del marco será colocado a 0.
     * @param color  Color del marco.
     */
    public EfectoMarco(String nombre, ImagenRGB imagen, int ancho , ColorRGB color) {

        super(nombre, imagen);
        colorMarco =  color;
        anchoMarco = ancho;

        ImagenProcesable = new ImagenRGB(imagen);
    }


    /**
     * Establece la anchura del marco.
     * @param ancho  Ancho que se le va a dar al marco. Máximo: la mitad del menor valor entre el alto y el ancho de la imagen.
     */

    public void setAnchura(int ancho) {

        int maxAnchoMarco = 1/2*(ImagenProcesable.getAlto()/ImagenProcesable.getAncho());

        if(ancho > maxAnchoMarco) {
            this.anchoMarco = 0;
        }else {
            this.anchoMarco = ancho;
        }
    }

    /**
     * Devuelve la anchura del marco.
     * @return Ancho para construir el marco.
     */
    public int getAnchura() {

        return anchoMarco;
    }

    /**
     * Establece el color del marco.
```

```
61      * @param color  Color para realizar el marco.
62      */
63     public void setColor(ColorRGB color) {
64 
65         if(color != null) {
66             colorMarco = color;
67         }
68     }
69 
70 
71     /**
72      * Devuelve el color del marco.
73      * @return Color para realizar el marco.
74      */
75 
76     public ColorRGB getColor() {
77         return colorMarco;
78     }
79 
80 
81     /**
82      * Aplica el efecto deseado: Introduce el marco parametrizado en la imagen. El resultado sobreescribe los píxeles necesarios de la imagen "para procesar" guardada en la superclase.
83      * Este método puede ser llamado varias veces. Por ejemplo: podríamos llamarlo una vez para realizar un marco azul de ancho 20 píxeles, cambiar el color a verde y el ancho a 10 píxeles, y
84      * volver a llamar al método "aplicar". De esta forma obtendríamos un marco formado por 2 colores (primero 10 píxeles en verde y después 10 píxeles en azul)
85      */
86 
87     public void aplicar() {
88 
89         int altoImagen = ImagenProcesable.getAlto();
90         int anchoImagen = ImagenProcesable.getAncho();
91 
92 
93         // Pintar borde superior//
94 
95         for(int y = 0; y < getAnchura(); y++) {
96             for(int x = 0; x < anchoImagen; x++) {
97 
98                 ImagenProcesable.setPixel(y, x, colorMarco);
99             }
100        }
101 
102        // Pintar borde inferior//
103 
104        for(int y = altoImagen - getAnchura(); y < altoImagen ; y++) {
105            for(int x = 0; x < anchoImagen; x++) {
106 
107                ImagenProcesable.setPixel(y, x, colorMarco);
108            }
109        }
110 
111        // Pintar borde izquierdo//
112 
113        for(int x = 0; x < getAnchura(); x++) {
114            for(int y = getAnchura(); y < altoImagen - getAnchura(); y++)  {
115 
116                ImagenProcesable.setPixel(y,x, colorMarco);
117            }
```

```
118         }
119
120         // Pintar borde derecho//
121
122         for(int x = anchoImagen -getAnchura(); x < anchoImagen; x++) {
123             for(int y = getAnchura(); y < altoImagen - getAnchura(); y++)  {
124
125                 ImagenProcesable.setPixel(y, x,colorMarco);
126             }
127         }
128     }
129 }
130
131
132
133
134
135
136
137
138
139
```