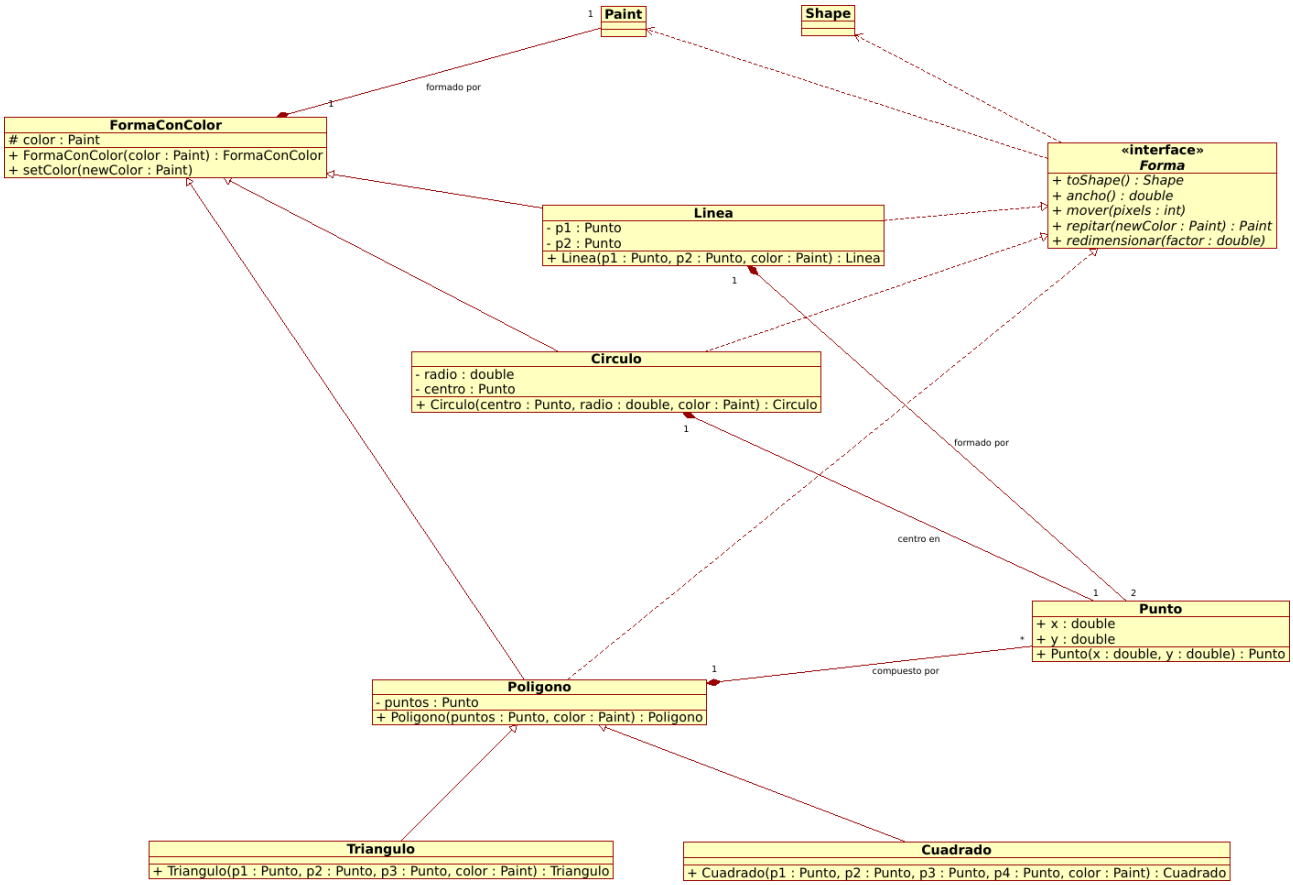
Diagrama de clases

Implementación

Enunciado

En esta práctica tienes que terminar de implementar un editor 2D escrito con Java y JavaFX. Esta parte por terminar la tienes modelizada en el siguiente UML:



El código a completar lo tienes en el repositorio siguiente: https://github.com/rgmf/eed-tema4-p3

Haz un *clone* del mismo, quita le remoto y súbelo a tu cuenta de GitHub para empezar a trabajar sobre esta base.

La **práctica se entregará vía GitHub** dentro del **plazo indicado en Aules**. El repositorio de GitHub en la que esté tu práctica se tiene que llamar **`eed-tema4-p3`** y será un **repositorio privado** en el que me pondrás a mí (rgmf) como colaborador para poder acceder a él.

Del diagrama UML tienes que saber que las clases Paint y Shape son de JavaFX y no las tienes que implementar, evidentemente. Los paquetes de estas clases y, los import, que tienes que usar son:

```
import javafx.scene.paint.Paint;
import javafx.scene.shape.Shape;
```

Explicación del modelo UML

En el UML ves modelado una editor 2D en el que tenemos tres tipos de figuras: líneas, círculos y polígonos en general. De los polígonos tenemos triángulos y cuadrados.

Además, como ves, todos los tipos de figuras heredan de FormaConColor e implementan la interfaz Forma. De esta manera, en nuestro programa "cliente" se hace uso del tipo genérico Forma para facilitar la programación: usamos, pues, polimorfismo por medio de la herencia de tipos/clases.

En los siguientes apartados te doy los detalles necesarios para implementar los métodos de todas las clases.

Compilar/ejecutar el proyecto con Maven

El código base es un proyecto de Java/Maven que usa el plugin JavaFX para poder ejecutar programas JavaFX.

Aunque desde un IDE o editor se puede ejecutar el programa con sencillez, te explico cómo hacerlo desde la terminal. Todos estos comandos hay que ejecutarlos desde la raíz del proyecto:

Para limpiar el proyecto:

`mvn clean install`

Para compilar y ejecutar el proyecto:

`mvn javafx:run`

Notas importantes

Todas las clases, interfaces... las tienes que crear dentro del a carpeta `formas` que verás en `src/main/java/com/proferoman/`

Clase FormaConColor

Es una clase muy simple que solo maneja el color. Para almacenar el color se usa la clase de `JavaFX` `Paint`. Tiene un atributo protegido llamado `color` que se inicializa en el constructor. Además, proporciona un método `setter` para cambiar el color.

Interfaz Forma

Es una interfaz que declara los métodos u operaciones que pueden realizar todas las formas. Todo lo que necesitas saber para esta interfaz viene dado en el UML.

Clase Punto

Dado que se van a manejar coordenadas o puntos (x, y) para posicionar las formas en la aplicación, aquí creamos una clase simple, con dos atributos públicos para almacenar las posiciones de los puntos que forman las figuras.

Todo lo que necesitas para este simple clase lo tienes en el UML.

Clases Linea, Circulo y Poligono

Estas clases se diferencian en el número de puntos y la manera en que se realizan las operaciones:

`toShape`

Linea	Circulo	Poligono

@Override
public Shape toShape() {
var line = new Line(p1.x, p1.y, p2.x, p2.y);
line.setStroke(color);
return line;
}
	@Override
	public Shape toShape() {
	return new Circle(centro.x, centro.y, radio, color);
	}
		@Override
		public Shape toShape() {
		var xy = new Double[puntos.size() * 2];
		int i = 0;
		for (Punto p : puntos) {
		xy[i++] = p.x;
		xy[i++] = p.y;
		}
		
		Polygon polygon = new Polygon();
		polygon.getPoints().addAll(xy);
		polygon.setFill(color);
		return polygon;
		}

`ancho`

- Linea: para calcular el ancho de una línea tienes que calcular la distancia que hay entre las coordenadas x de ambos puntos.
- Circulo: el cálculo del ancho de un círculo es muy sencillo: el doble del radio.
- Poligono: para calcular el ancho de un poligono en general tienes que obtener los puntos (x, y) más separados en sus coordenadas y luego calcular la distancia entre dichas coordenadas x.

`mover`

Esta operación tiene que mover en horizontal (eje x) la figura para lo cual hay que sumarle los pixeles a las coordenadas x de todos los puntos.

`repitar`

Esta operación solo modificar el atributo color de la figura.

`redimensionar`

Dado el factor a usar, solo hay que multiplicar las coordenadas (x, y) de todos los puntos. El caso del Circulo es diferente, tienes que multiplicar el factor por el radio.

Clases Triangulo y Cuadrado

Estas clases derivan de Poligono y solo tienes que implementar sus constructores para construir el Poligono en cada caso.

Así pues, tendrás que llamar al constructor de la clase base (Poligono) con el método super de Java.