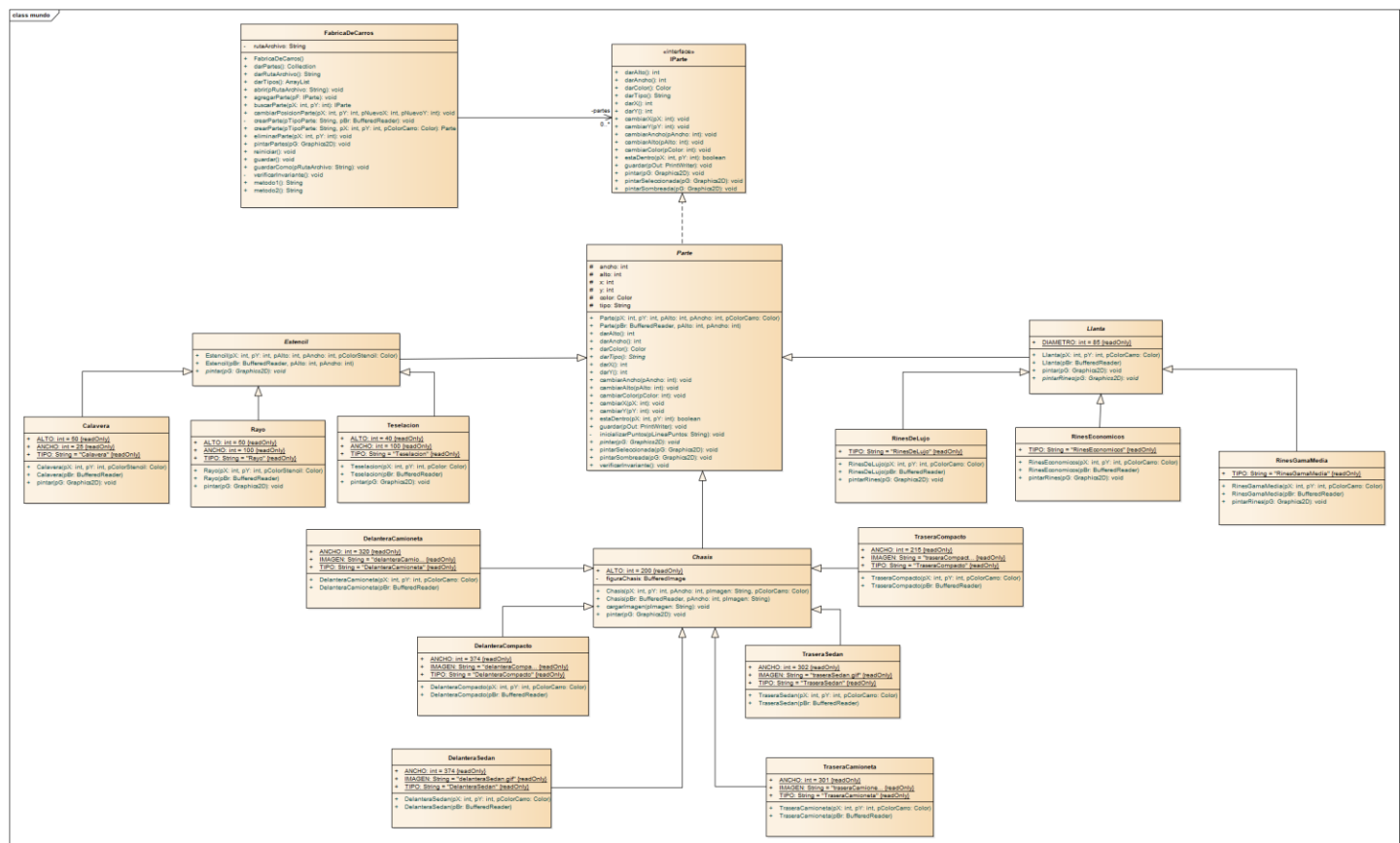


 Proyecto Cupi2	ISIS-1205 Algorítmica y Programación II <b>Consideraciones adicionales de diseño</b>
Ejercicio:	n10_fabricaDeCarros
Autor:	Equipo Cupi2 2019
Semestre:	2019-1

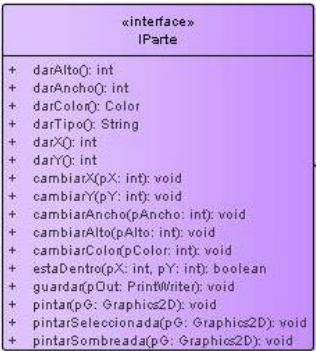
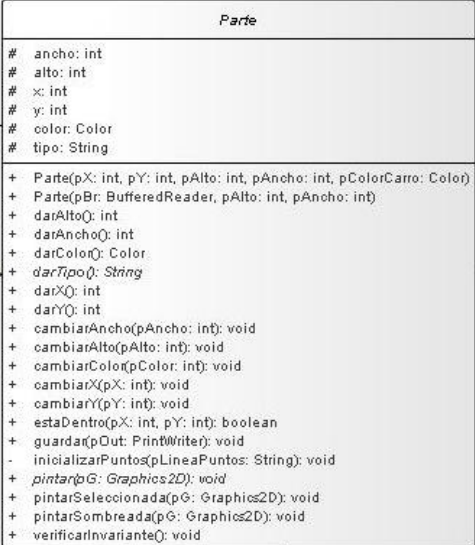
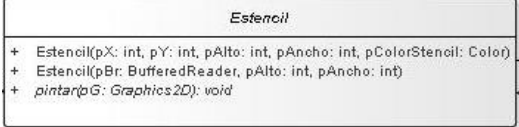
## Reutilización y desacoplamiento

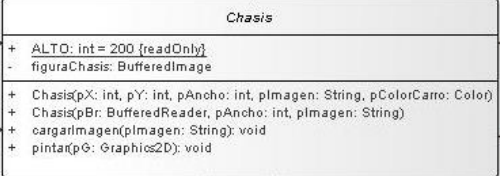
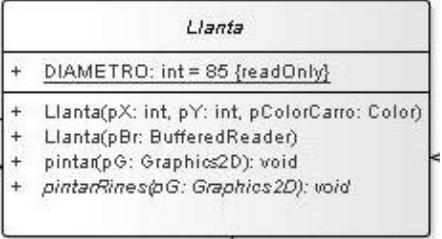
La aplicación debe estar construida de manera que sea fácilmente extensible. Esto quiere decir que debe permitir agregar fácilmente nuevos tipos de fábrica de carros conformadas de nuevas partes y nuevos requerimientos funcionales. Los elementos del modelo conceptual con los que se construye esta aplicación deben estar desacoplados entre sí y deben ser, en lo posible, reutilizables.

Considerando lo anterior, se propone el siguiente modelo del mundo:



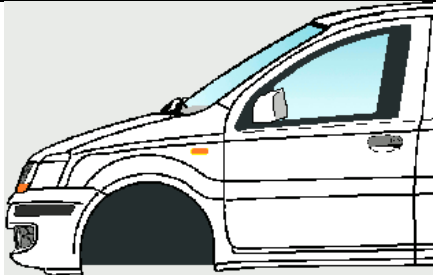
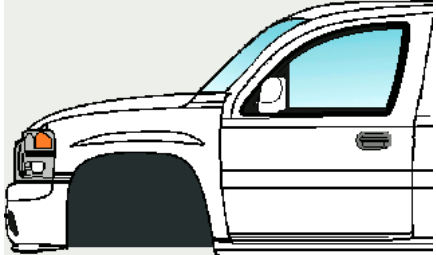
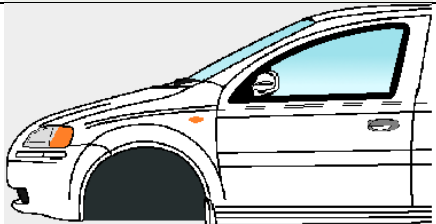
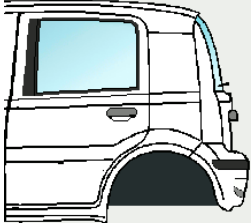
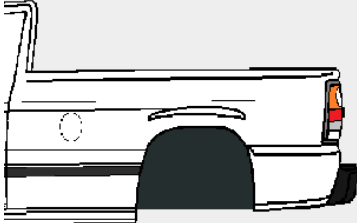
En la siguiente tabla se describen algunas de las clases incluidas para que la aplicación cumpla con las características de desacoplamiento y reutilización esperadas:

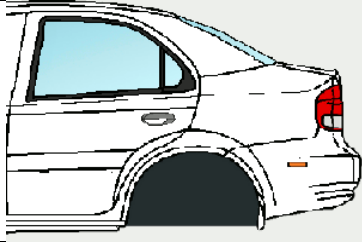
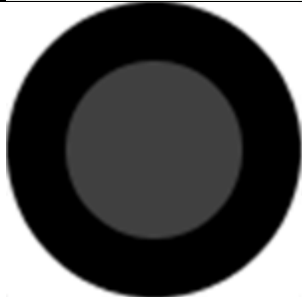
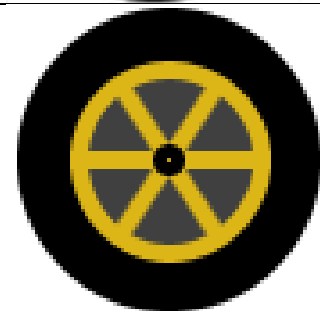
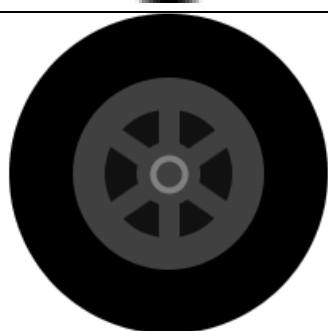

Clase	Descripción
 <pre> «interface» IParte  + darAlto(): int + darAncho(): int + darColor(): Color + darTipo(): String + darX(): int + darY(): int + cambiarX(pX: int): void + cambiarY(pY: int): void + cambiarAncho(pAncho: int): void + cambiarAlto(pAlto: int): void + cambiarColor(pColor: int): void + estaDentro(pX: int, pY: int): boolean + guardar(pOut: PrintWriter): void + pintar(pG: Graphics2D): void + pintarSeleccionada(pG: Graphics2D): void + pintarSombreada(pG: Graphics2D): void </pre>	<p>IParte (Interface)</p> <p>Interface que define las funcionalidades que debe ofrecer cualquier tipo de parte. Considera todos los aspectos relevantes gráficos de cada parte en la fábrica de carros.</p>
 <pre> Parte  # ancho: int # alto: int # x: int # y: int # color: Color # tipo: String  + Parte(pX: int, pY: int, pAlto: int, pAncho: int, pColorCarro: Color) + Parte(pBr: BufferedReader, pAlto: int, pAncho: int) + darAlto(): int + darAncho(): int + darColor(): Color + darTipo(): String + darX(): int + darY(): int + cambiarAncho(pAncho: int): void + cambiarAlto(pAlto: int): void + cambiarColor(pColor: int): void + cambiarX(pX: int): void + cambiarY(pY: int): void + estaDentro(pX: int, pY: int): boolean + guardar(pOut: PrintWriter): void + inicializarPuntos(pLineaPuntos: String): void + pintar(pG: Graphics2D): void + pintarSeleccionada(pG: Graphics2D): void + pintarSombreada(pG: Graphics2D): void + verificarInvariante(): void </pre>	<p>Parte (Abstract)</p> <p>Clase abstracta que representa una parte. Define las propiedades (atributos y métodos) que comparten todas las partes. Sirve como base para la implementación de las clases de todas las partes. Esta clase se compromete con el contrato funcional de la interface <i>IParte</i>.</p>
 <pre> Estencil  + Estencil(pX: int, pY: int, pAlto: int, pAncho: int, pColorStencil: Color) + Estencil(pBr: BufferedReader, pAlto: int, pAncho: int) + pintar(pG: Graphics2D): void </pre>	<p>Estencil (Abstract)</p> <p>Clase abstracta que representa un estencil. Define métodos que todo estencil ejecuta de</p>

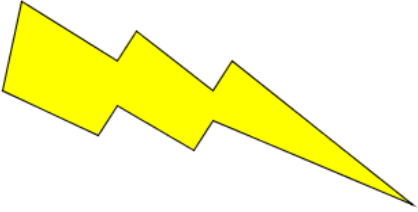


	manera similar. Esta clase hereda de la clase Parte.
 <pre> classDiagram     class Chasis {         +ALTO: int = 200 {readOnly}         -figuraChasis: BufferedImage         +Chasis(pX: int, pY: int, pAncho: int, plmagen: String, pColorCarro: Color)         +Chasis(pBr: BufferedReader, pAncho: int, plmagen: String)         +cargarImagen(plmagen: String): void         +pintar(pG: Graphics2D): void     }         </pre>	<p><b>Chasis (Abstract)</b></p> <p>Clase abstracta que representa un Chasis. Define métodos y constantes que todo Chasis ejecuta de manera similar. Esta clase hereda de la clase Parte.</p>
 <pre> classDiagram     class Llanta {         +DIAMETRO: int = 85 {readOnly}         +Llanta(pX: int, pY: int, pColorCarro: Color)         +Llanta(pBr: BufferedReader)         +pintar(pG: Graphics2D): void         +pintarRines(pG: Graphics2D): void     }         </pre>	<p><b>Llanta (Abstract)</b></p> <p>Clase abstracta que representa un Llanta. Define métodos y constantes que toda Llanta ejecuta de manera similar. Esta clase hereda de la clase Parte.</p>

## Partes

A continuación se presentan las partes con las que cuenta la fábrica de carros, y sus dimensiones.

Nombre	Figura	Consideraciones	Clase Padre
DelanteraCompacto		<b>Alto :</b> 200 <b>Ancho:</b> 374 <b>Archivo:</b> delanteraCompacto.gif	Chasis
DelanteraCamioneta		<b>Alto:</b> 200 <b>Ancho:</b> 320 <b>Archivo :</b> delanteraCamioneta.gif	Chasis
DelanteraSedan		<b>Alto:</b> 200 <b>Ancho:</b> 374 <b>Archivo:</b> delanteraSedan.gif	Chasis
TraseraCompacto		<b>Alto:</b> 200 <b>Ancho:</b> 215 <b>Archivo:</b> traseraCompacto.gif	Chasis
TraseraCamioneta		<b>Alto:</b> 200 <b>Ancho:</b> 301 <b>Archivo:</b> traseraCamioneta.gif	Chasis

TraseraSedan		<b>Alto:</b> 200 <b>Ancho:</b> 302 <b>Archivo:</b> traseraSedan.gif	Chasis
Llanta (clase abstracta)		<b>Alto:</b> 85 <b>Ancho:</b> 85 Dibujado con Java2D	Parte
RinesDeLujo		<b>Alto:</b> 85 <b>Ancho:</b> 85 Dibujado con Java2D	Llanta
RinesGamaMedia		<b>Alto:</b> 85 <b>Ancho:</b> 85 Dibujado con Java2D	Llanta
RinesEconomicos		<b>Alto:</b> 85 <b>Ancho:</b> 85 Dibujado con Java2D	Llanta

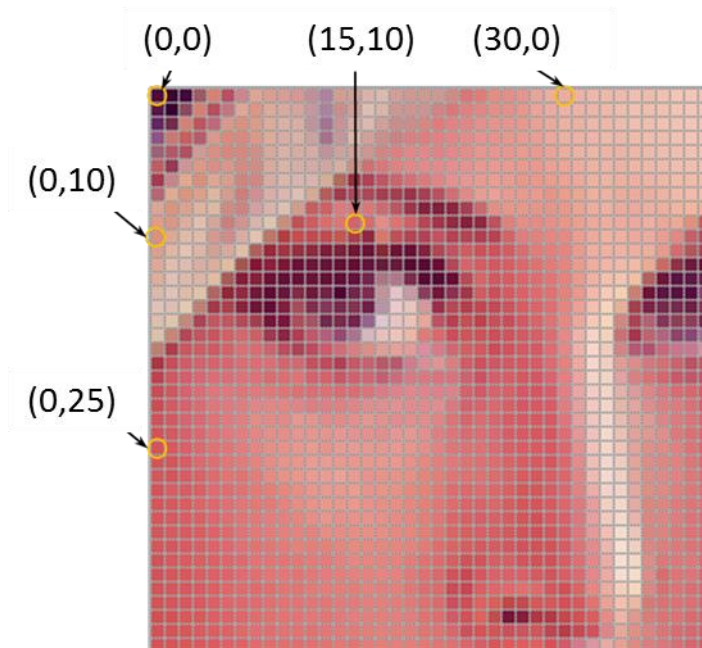
Rayo		<b>Alto:</b> 50 <b>Ancho:</b> 100 Dibujado con Java2D	Estencil
Teselacion		<b>Alto:</b> 40 <b>Ancho:</b> 100 Dibujado con Java2D	Estencil
Calavera		<b>Alto:</b> 50 <b>Ancho:</b> 25 Dibujado con Java2D	Estencil

A continuación se dan consejos útiles que guiarán al estudiante a dibujar correctamente cada parte de la fábrica.

## Sistema de coordenadas

A continuación se exponen varios detalles útiles para el desarrollo del proyecto:

- Las coordenadas de los símbolos gráficos del editor están dadas en píxeles.
- Java2D dibuja sobre una superficie de píxeles, cuya dimensión depende del panel en donde se esté dibujando.
- El origen de coordenadas se encuentra en la posición (0,0) que está situada en la esquina SUPERIOR IZQUIERDA de la superficie de dibujo (lienzo).
- La primera coordenada se refiere a la COLUMNA del píxel con el cual se está trabajando. Es creciente hacia la derecha.
- La segunda coordenada se refiere a la FILA del píxel con el cual se está trabajando. Es creciente HACIA ABAJO.
- Todo esto se ilustra en la siguiente figura:



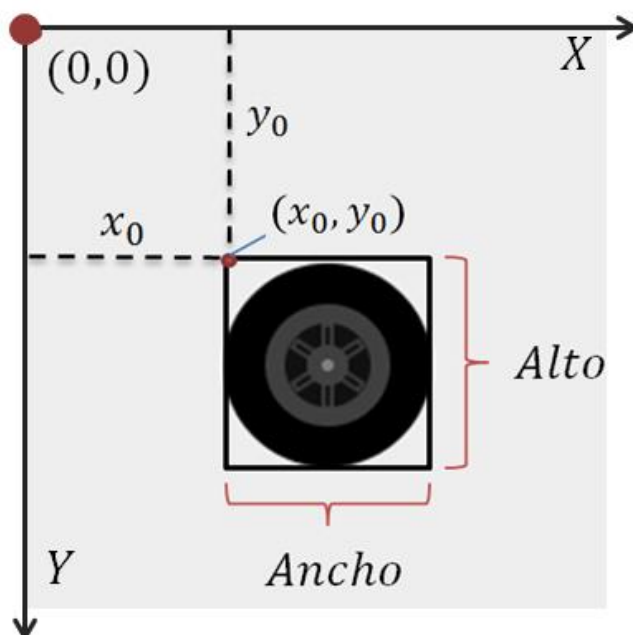
Se debe tener esto en cuenta, pues en el código las variables de las coordenadas se llaman  $(x, y)$ , que “inducen” a pensar erróneamente en un plano cartesiano, que tiene el origen en la esquina inferior izquierda, donde la primera coordenada es creciente hacia la derecha y la segunda coordenada es creciente hacia arriba.



## Sugerencias para pintar las partes de la fábrica de carros.

La fábrica de carros cuenta con varios tipos de partes que permiten dibujar en conjunto un carro. Como se explicó arriba, el panel de edición funciona como un sistema de coordenadas  $(x, y)$  que tiene su origen  $(0,0)$  en la esquina superior izquierda del panel, y aumenta sus unidades (en píxeles) hacia la derecha para 'x' y hacia abajo para 'y'. Este sistema sirve para definir e identificar la posición de las partes en el panel.

Las diferentes partes cuentan con una representación gráfica en el panel de edición. Esta representación está inscrita en un rectángulo (que no se dibuja) de dimensiones 'alto' y 'ancho'. La posición  $(x, y)$  de la parte hace referencia a la posición de la esquina superior izquierda del rectángulo en el panel de edición. La Figura 1 explica esto.



**Figura 1. Ejemplo de una parte en el panel de edición.**

La clase Graphics2D es utilizada para realizar el trazado de diferentes figuras básicas (rectángulos, óvalos, polígonos etc.) con o sin relleno<sup>1</sup>, así como cargar y dibujar imágenes diseñadas que en conjunto forman cada una de las partes. Además de esto, permite variar el color con el que se está dibujando con el método setColor(Color c) y el grosor de las líneas con el método setStroke<sup>2</sup>.

A continuación se dan indicaciones de cómo elaborar las figuras de las diferentes partes que ofrece la fábrica de ropa.

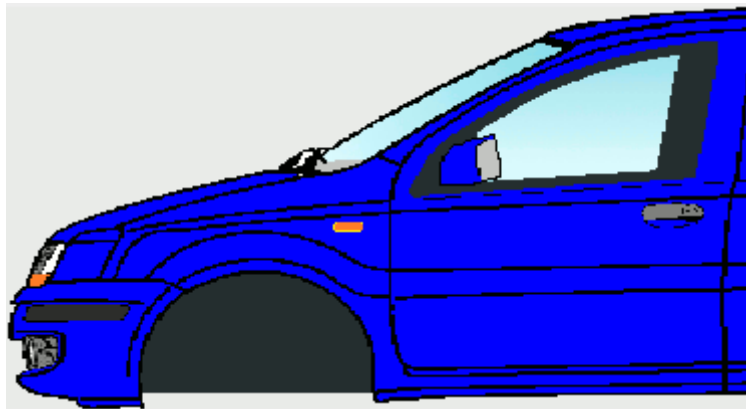
<sup>1</sup> Observe por ejemplo la diferencia entre el método drawOval() y fillOval().

<sup>2</sup> Refiérase a la documentación de la clase BasicStroke y al método de Graphics2D setStroke.



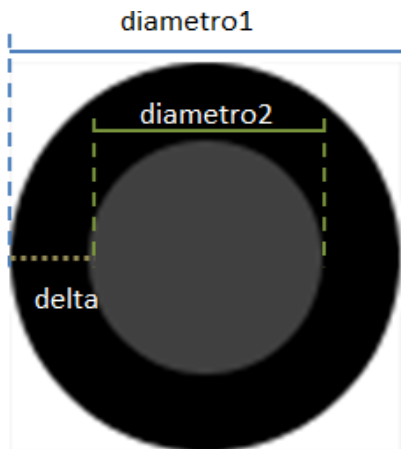
## Chasis (abstract).

La clase Chasis hace referencia a los tipos de delanteras y traseras del carro. Esta clase se encarga de cargar una figura ya diseñada y guardada dentro de la carpeta “./data/imágenes/”. Según el tipo de chasis esta figura varía. La figura se pinta con el color de fondo escogido por el usuario en el panel de partes. Un ejemplo de esto es el chasis DelanteraSedan.



## Llanta (abstract).

La clase abstracta Llanta, define el modelo básico de cualquier tipo de llanta. Tiene una constante llamada DIAMETRO que hace referencia al diámetro de la llanta. Este tiene el mismo valor de la variable ANCHO. A continuación se explica cómo dibujarla.

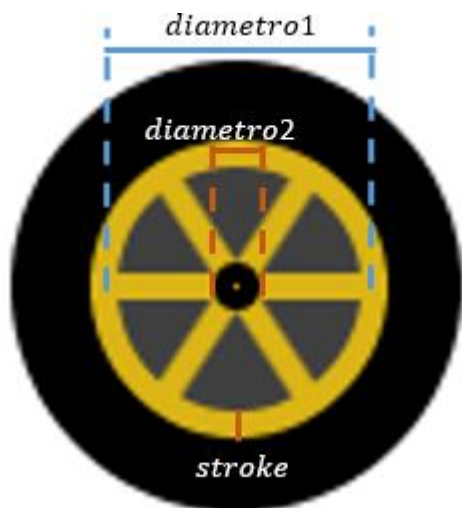


<i>delta</i>	$\frac{DIAMETRO}{5}$
<i>diametro1</i>	$DIAMETRO$
<i>diametro2</i>	$\frac{3 * DIAMETRO}{5}$

Ya definido el modelo de la clase Llanta, se explica cómo dibujar los diferentes tipos de llantas que se crean a partir de esta definición.

## ➤ *RinesDeLujo:*

Clase que representa los rines de lujo. Su marco es en color amarillo.



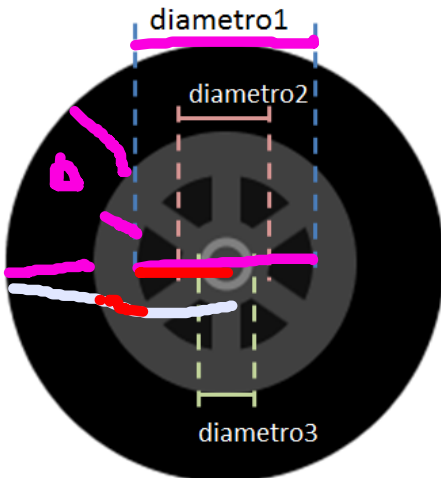
<i>diámetro1</i>	$\frac{3 * DIAMETRO}{5}$
<i>diámetro2</i>	4
<i>stroke</i>	$\frac{DIAMETRO}{15}$



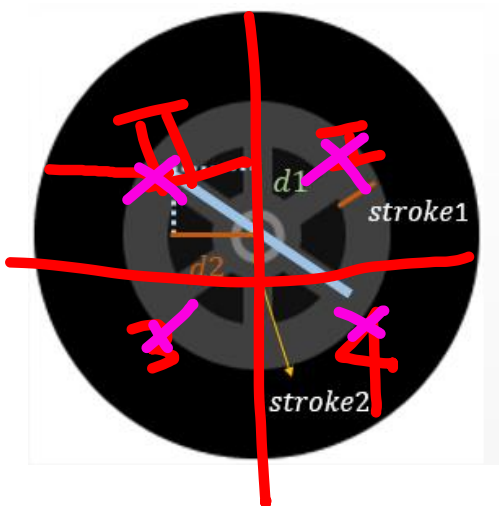
<i>d1</i>	$\frac{diámetro1 * 0.829}{2}$
<i>d2</i>	$\frac{diámetro1 * 0.559}{2}$

➤ **RinesGamaMedia:**

Clase que representa los rines de gama media.



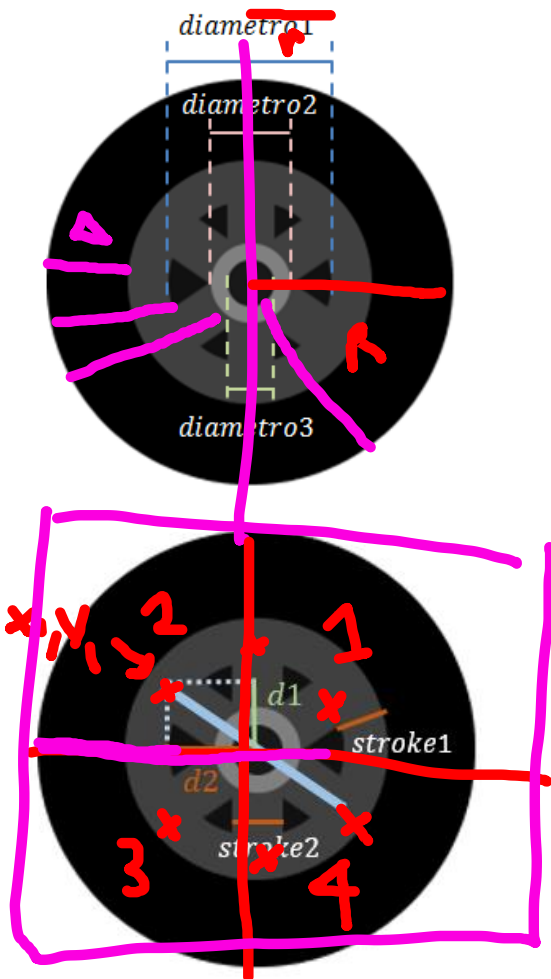
$d1$	$\frac{\text{diametro1} * 0.559}{2}$
$d2$	$\frac{\text{diametro1} * 0.829}{2}$
$\text{stroke1}$	$\frac{\text{DIAMETRO}}{15}$
$\text{stroke2}$	$\frac{\text{DIAMETRO}}{50}$



$\text{diametro1}$	$\frac{2 * \text{DIAMETRO}}{5}$
$\text{diametro2}$	$\frac{\text{DIAMETRO}}{5}$
$\text{diametro3}$	$\frac{\text{DIAMETRO}}{10}$

➤ **RinesEconomicos:**

Clase que representa los rines económicos.



<i>diametro1</i>	$\frac{2 * DIAMETRO}{5}$
<i>diametro2</i>	$\frac{2 * DIAMETRO}{25}$
<i>diametro3</i>	$\frac{DIAMETRO}{20}$

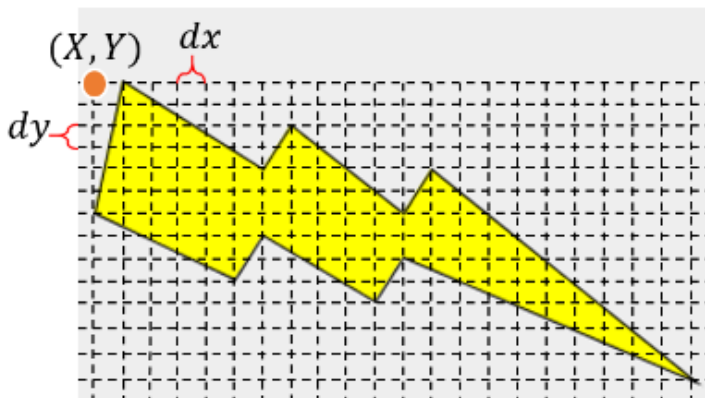
<i>d1</i>	$\frac{diametro1 * 0.559}{2}$
<i>d2</i>	$\frac{diametro1 * 0.829}{2}$
<i>stroke1</i>	$\frac{DIAMETRO}{15}$
<i>stroke2</i>	$\frac{DIAMETRO}{8.6}$

## Estencil (abstract).

Es la clase modelo para modelar los estencils que ofrece la fábrica de carros. Para la elaboración de estas partes le será útil consultar el método `fillPolygon()` de la clase `Graphics2D`.

### ➤ **Rayo:**

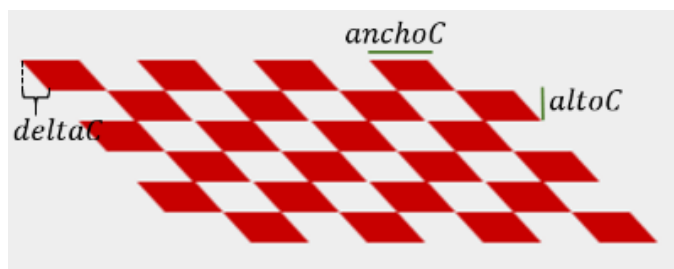
Clase que modela un estencil con forma de rayo. Se especifica una cuadrícula con su separación para que los puntos que forman la figura se puedan identificar fácilmente.



$dx$	$\frac{ANCHO}{21}$
$dy$	$\frac{ALTO}{13}$

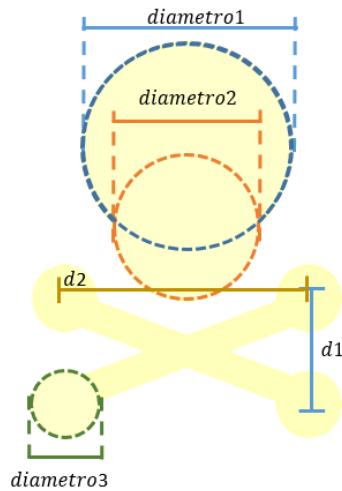
### ➤ **Teselacion:**

Clase que modela un estencil con patrón de teselación en rojo.

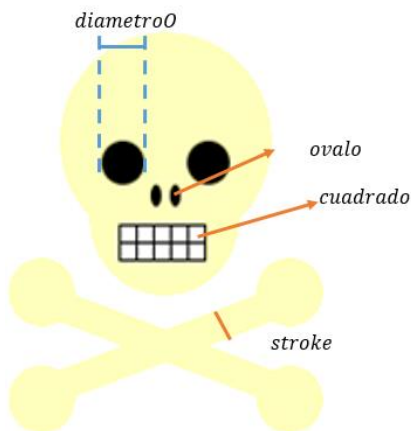


$anchoC$	$\frac{ANCHO}{8}$
$altoC$	$\frac{ALTO}{6}$
$deltaC$	$\frac{anchoC}{2}$

➤ **Calavera:**



<i>diametro1</i>	<i>ANCHO</i>
<i>diametro2</i>	$5 * \frac{ANCHO}{7}$
<i>diametro3</i>	$\frac{ANCHO}{3}$
<i>d1</i>	$11 * \frac{ANCHO}{45}$
<i>d2</i>	$7 * \frac{ANCHO}{6}$



<i>diametro0</i>	$\frac{ANCHO}{5}$
<i>ovalo → alto</i>	$\frac{ANCHO}{9}$
<i>ovalo → ancho</i>	$\frac{ANCHO}{18}$
<i>cuadrado → lado</i>	$\frac{ANCHO}{12}$
<i>stroke</i>	$\frac{ANCHO}{6}$