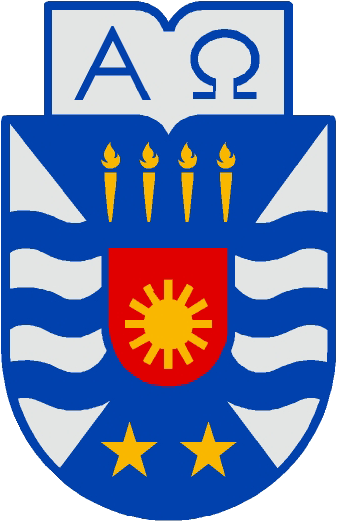
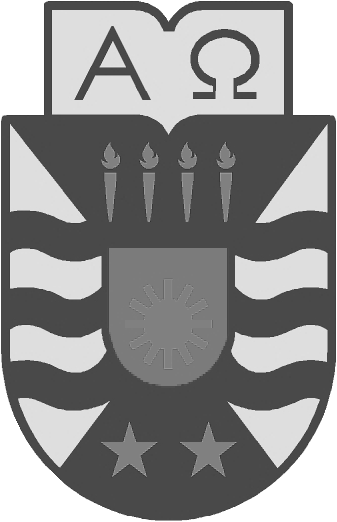
|  |
| --- |
| **Universidad del Bío-Bío Facultad de Ciencias Empresariales**  **Departamento de Ciencias de la Computación y Tecnologías de la Información** |



**Trabajo 2**

**SVG**

|  |
| --- |
| **Asignatura:** Fundamentos de Ciencias de la Computación.  **Docente:** Luis Gajardo.  **Integrantes:** Benjamín Días  Gastón Lara  Juan Muñoz  Camila Pinto  **Fecha de entrega:** 12/07/2017*.* |

# ÍNDICE

[ÍNDICE 2](#_Toc487629417)

[INTRODUCCIÓN 3](#_Toc487629418)

[EXPLICACIÓN GRAMÁTICA 4](#_Toc487629419)

[CLASES JAVA ADICIONALES 6](#_Toc487629420)

[ **Visor** 6](#_Toc487629421)

[ **Visitador** 6](#_Toc487629422)

[ **Figura** 7](#_Toc487629423)

[ **Circle** 8](#_Toc487629424)

[ **Ellipse** 9](#_Toc487629425)

[ **Line** 9](#_Toc487629426)

[ **Polygon** 10](#_Toc487629427)

[ **Polyline** 10](#_Toc487629428)

[ **Rectangle** 11](#_Toc487629429)

[OTRAS COSAS DE INTERÉS 12](#_Toc487629430)

[RESULTADOS 13](#_Toc487629431)

[CONCLUSIÓN 14](#_Toc487629432)

[BIBLIOGRAFÍA 15](#_Toc487629433)

# INTRODUCCIÓN

**Objetivos:** Tenemos que hacer una gramática que lea formas svg y bla bla, para hacer un java que aplique parsing y bla bla.

# EXPLICACIÓN GRAMÁTICA

La gramática fue escrita en el archivo svg.grammar, que está conformado por dos secciones: *Tokens* y *Productions*. Tokens contiene los elementos que servirán como terminales de la gramática, mientras que Productions indica las reglas que rigen la gramática.

Para elaborar la gramática, ha sido necesario probar códigos SVG en distintas formas, partiendo por crear una línea simple (Ilustración 1), con sus respectivas propiedades y atributos de estilo.

Se considera que todas las figuras tienen en común los atributos de *style*, pero cada una contiene un conjunto único de propiedades para ser construida (a excepción de Polygon y Polyline, que ambas tienen la propiedad *points*).



Ilustración : Línea simple, de color azul y ancho 2, escrita en SVG.

La gramática también incluye casos en que, en lugar de utilizar una etiqueta style, se utilizan los atributos de color y ancho de línea como si fueran propiedades de la figura (Ilustración 2).

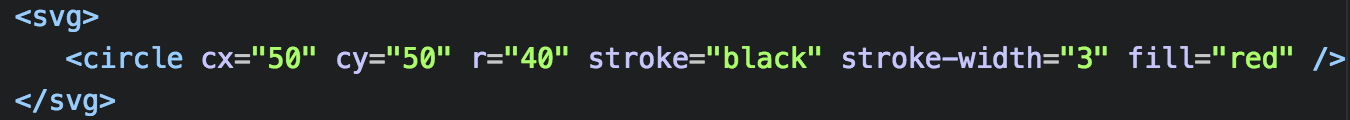


Ilustración : Círculo en SVG. En lugar de utilizar la etiqueta style, se especifican los atributos de estilo como las demás propiedades.

Finalmente, se añadió la posibilidad de unir múltiples figuras entre las etiquetas *<svg> … </svg>* y que el parser acepte los atributos de tamaño (width y height) dentro de la etiqueta <svg … >. Para lograr que el parser reconozca múltiples figuras se utilizan producciones recursivas, asegurando que se pueda insertar un número ilimitado de formas.

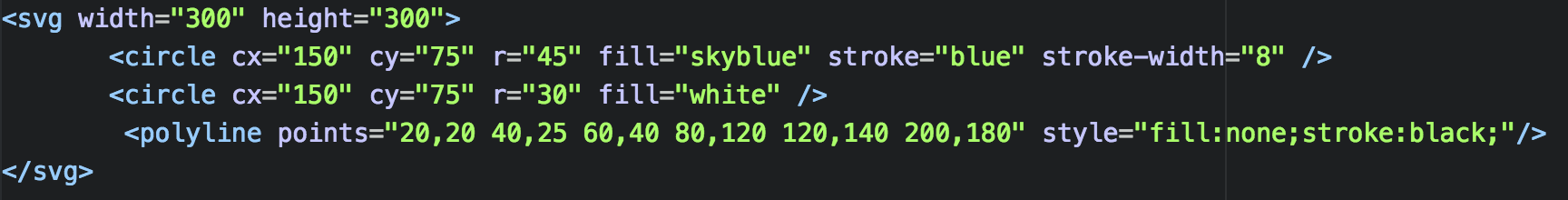


Ilustración :Dibujo SVG compuesto por múltiples figuras, combinando círculos y una polilínea.

**PRODUCTIONS**

A continuación se definen las producciones más relevantes:

**start:** Estado inicial de la gramática. Se insertan las etiquetas <svg> y </svg>, con la posibilidad de agregar dimensiones en <svg…>.

**dimensiones:** Las dimensiones del espacio donde se generará la figura.

**figura:** Se ve si es una figura o varias figuras concatenadas.

**pr1:** La producción de cada figura con sus atributos y estilo. El estilo puede ser definido como propiedades de la figura (Ilustración 2), por lo que la etiqueta *style* se hace opcional.

**f:** Nombre de la figura junto a sus propiedades.

**att:** Atributos de estilo expresados como propiedades. Se pueden generar recursivamente.

**att0:** Los atributos de las figuras.

**l0, c2, r2**, **e0, p3:** Propiedades específicas de cada tipo de figura.

**coorx**, **coory**, **coorx2,** **coory2:** Coordenadas de los puntos p1 y p2 de una línea.

**coorcx, coorcy,** **rad:** Radio y coordenadas del centro de un círculo.

**coorx0, coory0,** **wid**, **hei:** Posición, ancho y alto del rectángulo.

**coorcx, coorcy, radx, rady:** Posición del centro y radio en *x* e *y* del elipse.

**num:** Número entero entre comillas.

**p0:** El conjunto de puntos de un polígono o una polilínea. Cada punto se genera recursivamente y se pueden generar de forma ilimitada.

**p1:** Coordenadas de un punto específico.

**s:** Etiqueta *style = ...*

**sp:** El conjunto de atributos de estilo de una figura.

**sp0:** Los atributos individuales de estilo de una figura.

**TOKENS**

**n** = ['0' .. '9']+

**blank** = ( ' ' | 13 | 10)+

**svg1** = <svg

**svg2** = </svg>

**minus** = <

**mayus** = >

**co** = "

**ig** = =

**sl** = /

**line** = line

**circle** = circle

**rect** = rect

**ellipse** = ellipse

**polygon** = polygon

**polyline** = polyline

**x1** = x1

**y1** = y1

**x2** = x2

**y2** = y2

**cx** = cx

**cy** = cy

**r0** = r

**x0** = x

**y0** = y

# CLASES JAVA ADICIONALES

## **Visor**

## **Visitador**

La clase *agregarFigura()* almacena las figuras que se ingresan.

Verificando sus respectivos parámetros (coordenadas, radios, etc.).

## **Figura**

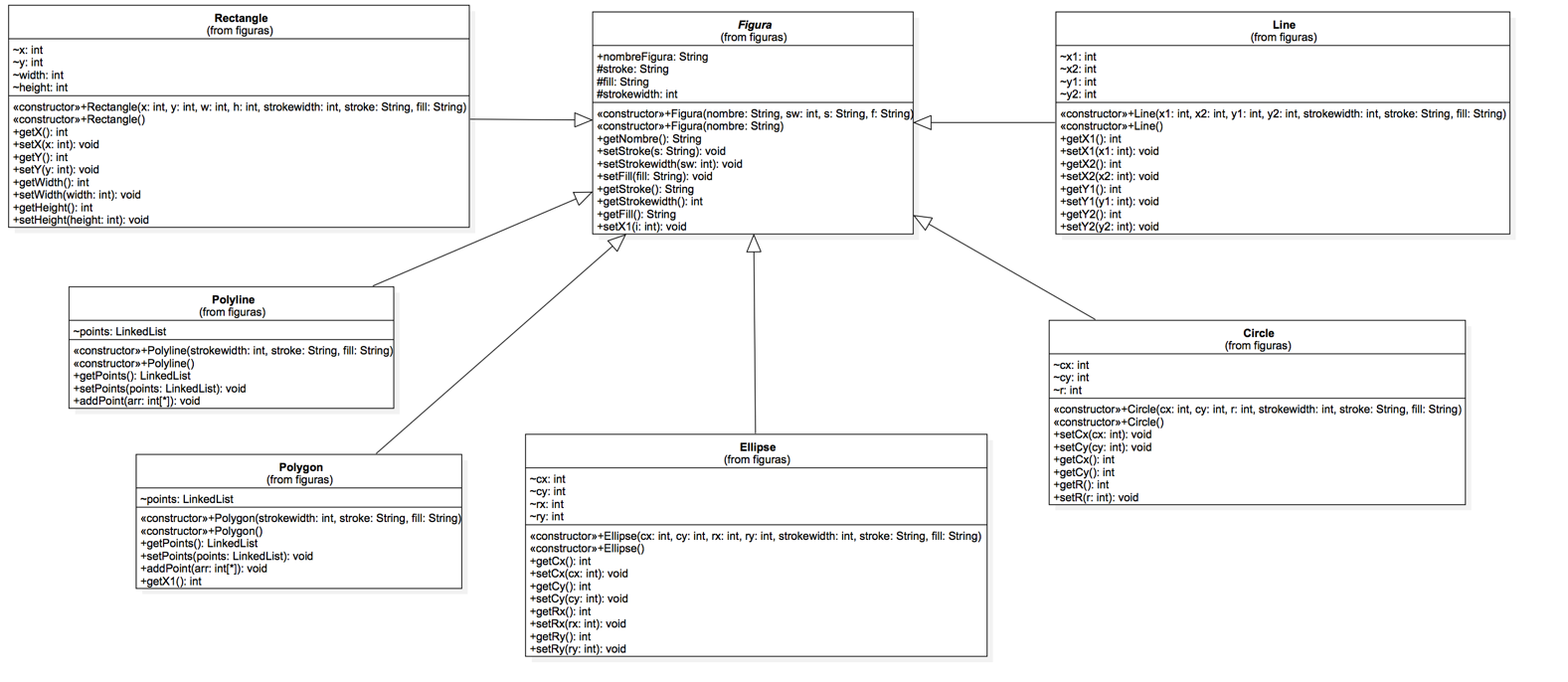


Ilustración : Diagrama UML representando la clase Figura y su herencia. Nótese que, por simplificar, no se han escrito todos los métodos de la clase Figura.

La clase *Figura* es una clase abstracta que heredará métodos y atributos (Ilustración 1) a las clases línea, círculo, rectángulo, elipse, polígono y polilínea. Los atributos de Figura corresponden a las propiedades *Style* de SVG (fill, stroke y stroke-width) y al nombre del tipo de figura.

Las subclases de Figura contienen constructores que llenan los atributos de la superclase con el nombre de la clase correspondiente y los valores por defecto de las figuras SVG: stroke = none; fill = black; stroke-width = 1.

## **Circle**

Los atributos *cx, cy,* son las coordenadas del eje *x*, y el eje *y* respectivamente. Y *r* es el radio de la figura.

*Nota:* Los atributos *cx, cy* de la clase *Circle* se comparten con la clase *Ellipse.*

## **Ellipse**

La clase *Ellipse* es heredera de la clase *Figura.* Ésta clase contiene el constructor, sus respectivos *getter and setter.* El método *super* del constructor especifíca el nombre de la figura correspondiente y sus atributos *stroke width, stroke y fill.*

Los atributos *cx, cy,* son las coordenadas del eje *x,* y el eje *y* respectivamente. Las coordenadas *rx,* y *ry,* son las coordenadas del radio de la figura.

*Nota:* Los atributos *cx, cy* de la clase *Ellipse* se comparten con la clase *Circle.*

## **Line**

La clase *Line* es heredera de la clase *Figura.* Ésta clase contiene el constructor, sus respectivos *getter and setter.* El método *super* del constructor especifíca el nombre de la figura correspondiente y sus atributos *stroke width, stroke y fill.*

Los atributos *x1, x2* son las coordenadas del eje *x* desde el punto de inicio hasta el punto final respectivamente*,* y los atributos *y1, y2* son las coordenadas del eje *y* desde el punto de inicio hasta el punto final respectivamente.

* **Polygon**

La clase *Polygon* es heredera de la clase *Figura.* Ésta clase contiene el constructor, sus respectivos *getter and setter.* El método *super* del constructor especifíca el nombre de la figura correspondiente y sus atributos *stroke width, stroke y fill.*

En esta clase se utilizó un LinkedList para almacenar los puntos

// Expláyate Benja, y explica por qué ocupaste un LinkedList para almacenar los puntos del polígono.

## **Polyline**

La clase *Polyline* es heredera de la clase *Figura.* Ésta clase contiene el constructor, sus respectivos *getter and setter.* El método *super* del constructor especifíca el nombre de la figura correspondiente y sus atributos *stroke width, stroke y fill.*

En esta clase se utilizó un LinkedList para almacenar los puntos

// Expláyate Benja, y explica por qué ocupaste un LinkedList para almacenar los puntos del poliline.

## **Rectangle**

La clase *Rectangle* es heredera de la clase *Figura.* Ésta clase contiene el constructor, sus respectivos *getter and setter.* El método *super* del constructor especifíca el nombre de la figura correspondiente y sus atributos *stroke width, stroke y fill.*

Los atributos *x, y* son las coordenadas en el eje *x,* y el eje *y* respectivamente, el atributo *w (width)* es el ancho de la figura, y el atributo *h (height)* es el alto de la figura.

# OTRAS COSAS DE INTERÉS

1. Este trabajo necesito el uso de varios conocimientos de ramos anteriormente cursados.
2. El poder aprender a utilizar la extensión de java Graphics 2D y todas sus funciones.

*//Esto no lo alcancé a modificar, de hecho solo lo leí a la rápida, pero ya veremos cómo podemos dejar esto bien hecho y bien redactado, así que si llegas aquí y yo sigo almorzando, me esperas para ayudarte;*

# RESULTADOS

// acá también esperame

# CONCLUSIÓN

**//y aquí quizás también (ojos)**

# BIBLIOGRAFÍA

<https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute>

<https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html>

<https://docs.oracle.com/javase/7/docs/api/java/awt/BasicStroke.html>

<https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html>

<http://sablecc.sourceforge.net/thesis/thesis.html#PAGE26>