**Brayan Santiago Buitrago Murillo**

**Gonzalez Cuentas César Eduardo**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Introducción. Clases y objetos.**

**2019-2**

**Laboratorio 1/6**

# SHAPES Conociendo el proyecto shapes

## [En lab01.doc]

1. **El proyecto “shapes” es una versión modificada de un recurso ofrecido por BlueJ. Para trabajar con él, bajen shapes.zip y ábranlo en BlueJ**
2. **El diagrama de clases permite visualizar las clases de un artefacto software y las relaciones entre ellas. Considerando el diagrama de clases de “shapes” ¿qué clases ofrece? ¿qué relaciones existen entre ellas?** 
   1. Ofrece las siguientes clases:

* Circle
* Rectangle
* Triangle
* Canvas

Y las relaciones que existen son de uso de parte de circle, rectangle y triangle hacia canvas para pintarse y vizualizarse.

1. **La documentación[[1]](#footnote-1) presenta las clases del proyecto y, en este caso, la especificación de sus componentes públicos. De acuerdo con la documentación generada:**

**¿qué clases tiene el paquete shapes?**

* Circle
* Rectangle
* Triangle
* Canvas

**¿qué atributos tiene la clase Circle?**

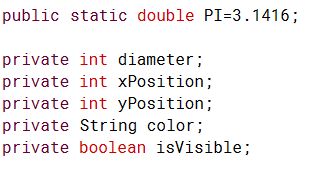
* Tiene un público llamado pi.

**¿cuáles métodos ofrece la clase Circle para que la figura cambie (incluya sólo el nombre)?**

* changeSize()
* changeColor()

1. **En el código de cada clase está el detalle de la implementación. Revisen el código de la clase Circle. Con respecto a los atributos:**

**¿cuántos atributos realmente tiene?**



**¿cuáles son privados y cuáles públicos?.**

El único privado es pi.

Y los privados:

* diameter
* xPosition
* yPosition
* color
* isVisible

**Con respecto a los métodos: ¿cuántos métodos tiene en total? ¿cuáles son privados?. ¿Quienes usan los componentes privados?**

Tiene en total 15 métodos incluyendo el constructor.

Los métodos privados son: draw() y erase()

Los atributos privados los utilizan los métodos que describen un comportamiento al objeto.

1. **¿Qué no se ve en la documentación? ¿por qué debe ser así?**

En la documentación no se ven los atributos ni los métodos privados, ya que estos solo son usados o vistos por la misma clase

1. **En el código de la clase Circle revisen el detalle del atributo PI. ¿qué se está indicando?**

public static double PI=3.1416;

Indica que el atributo además de ser public se mantiene constante en todo momento

1. **¿Cuál dirían es el propósito del proyecto “shapes”?**

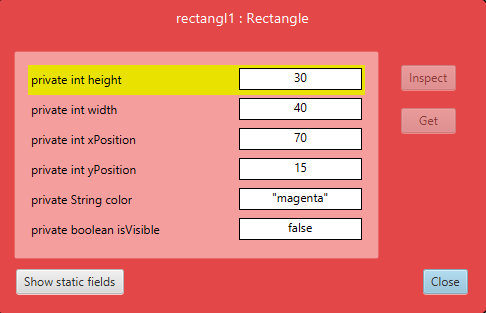
Aprender a manejar clases y objetos, en este caso modificando y moviendo figuras, observando sus respectivos comportamientos con los diferentes métodos.

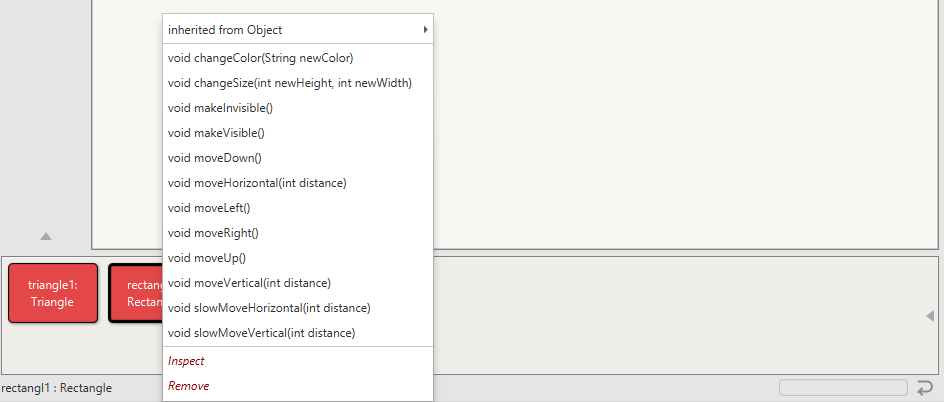
**Manipulando objetos. Usando opciones.**

## [En lab01.doc]

1. **Creen un objeto de cada una de las clases que lo permitan[[2]](#footnote-2). ¿cuántas clases hay? ¿cuántos objetos crearon? ¿por qué?**

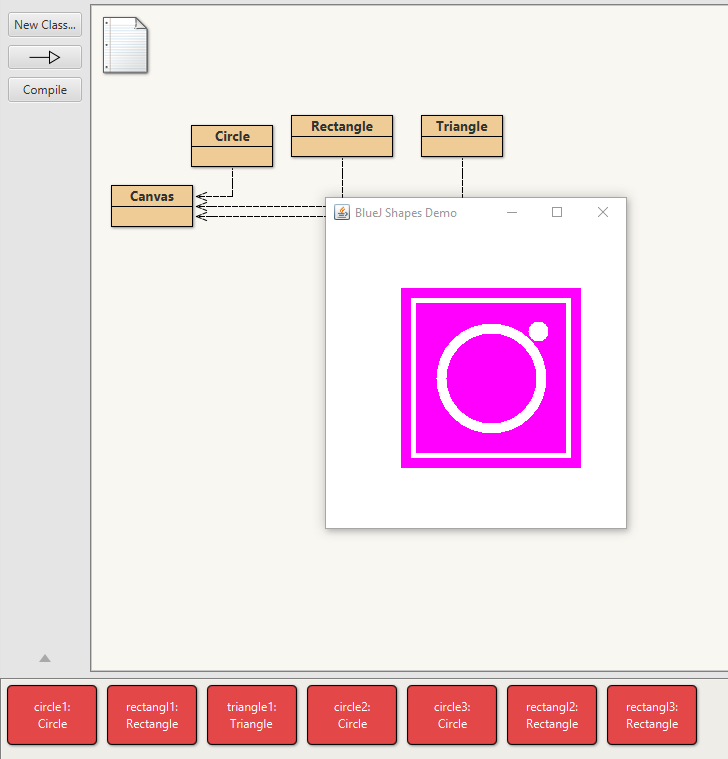
Hay 4 clases y creamos 3 objetos, porque las clases Circle, Rectangle y Triangle usan la clase Canvas por lo tanto al crear una figurase visualizara en el canvas.

1. **Inspeccionen el estado del objeto :Rectangle[[3]](#footnote-3), ¿cuáles son los valores de inicio de todos sus atributos? Capturen las pantallas**
2. **Inspeccionen el comportamiento que ofrece el objeto :Rectangle5. Capturen la pantalla. ¿por qué no aparecen todos los que están en el código?**

****

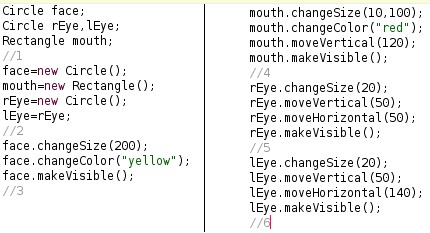
No aparecen todos los métodos, porque los privados solo los utiliza la misma clase

1. **Construyan, con “shapes” sin escribir código, una propuesta de la imagen de su *app* favorito. ¿Cuántas y cuáles clases se necesitan? ¿Cuántos objetos se usan en total? Capturen la pantalla.**

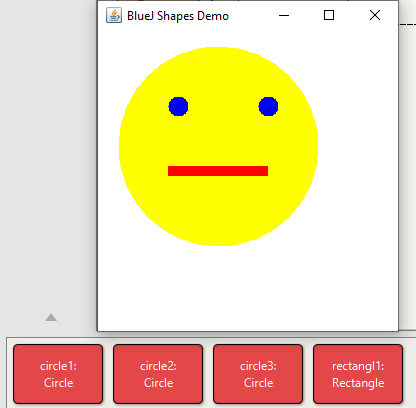


**Manipulando objetos. Analizando y escribiendo código.**

## [En lab01.doc]

****

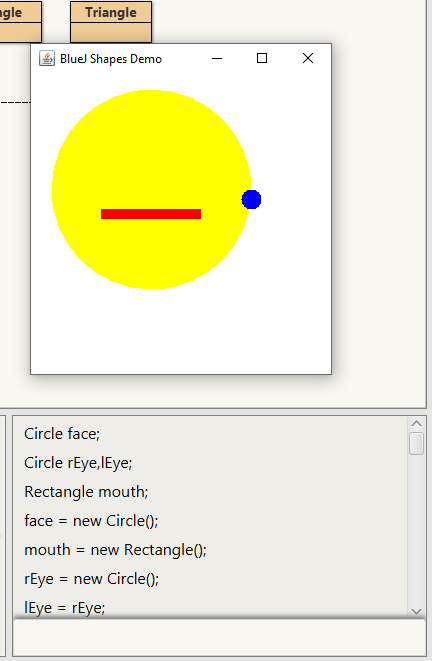
1. **Lean el código anterior ¿cuál es la figura resultante? Píntenla.**



1. **Habiliten la ventana de código en línea[[4]](#footnote-4), escriban el código y para cada punto señalado indiquen: ¿cuántas variable existen? ¿cuántos objetos existen? ¿qué color tiene cada uno de ellos? ¿cuántos objetos se ven? Expliquen. Capturen la pantalla.**

Hay 4 variables, pero existen 3 objetos. La cara que es un circulo tiene color amarillo, la boca que es un rectángulo tiene color rojo y el ojo que “supuestamente son dos” es azul y es otro circulo.

Se ven 3 objetos, porque en la línea de código al hacer rEye=lEye solo se le asigno otro nombre al mismo objeto más no se creó otro.



1. **Es igual la figura pintada en 1. igual a la figura capturada en 2. , ¿por qué?**

No es igual porque uno entiende por encima que hay 4 objetos, pero la máquina solo ve tres, porque en la línea de código al hacer rEye=lEye solo se le asigno otro nombre al mismo objeto más no se creó otro.

# Extendiendo clases

## [En lab01.doc y \*.java]

1. **Desarrollen en Circle el método growUp(times) (que hace que crezca lentamente el número dado de veces) . ¡Pruébenlo! grow up**
2. **Desarrollen en Circle el método area(). ¡Pruébenlo!**
3. **Desarrollen en Circle el método jump() (que hace que salte a la derecha) . ¡Pruébenlo!**
4. **Generen nuevamente la documentación y revise la información de estos nuevos métodos. Capture la pantalla.**

# SLOT MACHINE

|  |  |
| --- | --- |
| **En este punto vamos a construir dos versiones de un tragamonedas.**  **La primera versión es un tragamonedas simple: sólo tiene una fila. Para ganar se deben tener todos los elementos iguales.**  **La segunda versión es un tragamonedas cuadrado. Para ganar se deben tener iguales todos los elementos de una fila, una columna o una diagonal.**  **Las dimensiones de un tragamodena debe estar entre 2 y 5. El número de elementos diferentes debe ser uno más que su dimensión.**  **Pueden escoger el tipo de elementos de su tragamoneda.** | **Este es un ejemplo de un tragamonedas de frutas de dimensión 3. Las posibles frutas son naranja, ciruela, manzaja y limón.**  **Versión 1**    **Versión 2** |

**Implementando una nueva clase. SlotMachine.**

## [En lab01.doc. SlotMachine.java]

|  |  |  |
| --- | --- | --- |
|  |  | **Miniciclo 1 SlotMachine pull pull(times) isWinningState**  **Miniciclo 2 reset**  **percentageOfWinning**  **Miniciclo 3 makeVisible makeInvisible**  **move** |

1. **¿De que tipo de elementos es el tragamonedas? ¿Cuáles son los diferentes elementos?**
2. **¿Cuál es la probabilidad de ganar en esta máquina? Explique su respuesta.**
3. **Clasifiquen los métodos en: constructores, analizadores y modificadores.**
4. **Desarrollen la clase SlotMachine considerando los miniciclos. Al final de cada miniciclo realicen una prueba. Capturen las pantallas relevantes.**
5. **¿Cual es el porcentaje de estados ganadores después de hacer 1, 10, 100 y 1000 jugadas? Presente un análisis de los datos considerando la respuesta dada en 2.**

# Definiendo y creando una nueva clase. SquareSlotMachine

## [En lab01.doc. SquareSlotMachine.java]

**El objetivo es implementar diseñar e implementar un tragamonedas cuadrado.**

**Requisitos funcionales**

* **Permitir crear un tragamonedas, indicando el tamaño.**
* **Permitir reiniciar el tragamonedas**
* **Permitir jugar (todas las filas giran)**
* **Permitir jugar indicando la fila con el que se desea jugar (1 ...n de arriba a abajo)**
* **Permitir consultar si se ha ganado**
* **Permitir consultar el porcentaje de juegos ganadores desde el último reinicio Requisitos de interfaz**
* **Las operaciones se deben ofrecer como métodos públicos de la clase SquareSlotMachine**
* **El tragamodedas compuesto debe “sonar” cada vez que llega a un estado ganador  Se debe presentar un mensaje amable al usuario si hay algún problema.**

**Consulte y use el método showMessageDialog de la clase JOptionPane.**

1. **Diseñen la clase SquareSlotMachine, es decir, definan los métodos que debe ofrecer.**
2. **Planifiquen la construción considerando algunos miniciclos.**
3. **Implementen la clase . Al final de cada miniciclo realicen una prueba de aceptación. Capturen las pantallas relevantes.**
4. **Indiquen las extensiones necesarias para reutilizar la clase SlotMachine. Explique.**
5. **Propongan un nuevo método para enriquecer el juego.**

**Extendiendo una clase. SquareSlotMachine.**

## [En lab01.doc. SquareSlotMachine.java]

**El objetivo es crear dos nuevas funcionalidades para jugadas “arregladas” : una que encuentre el mínimo número de movientos para ganar y la otra que permita hacer uno de los movimientos planeados. Nuevos requisitos funcionales**

1. **Mínimos números de movientos**
2. **Mueva siguiendo la estrategia**

# RETROSPECTIVA

1. **¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/ Hombre)**
2. **¿Cuál es el estado actual del laboratorio? ¿Por qué?**
3. **Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?**
4. **¿Cuál consideran fue el mayor logro? ¿Por qué?**
5. **¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?**
6. **¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para** mejorar los resultados?

1. Menu: Tools-Project Documentation [↑](#footnote-ref-1)
2. Clic derecho sobre la clase [↑](#footnote-ref-2)
3. Clic derecho sobre el objeto 5 Hacer clic derecho sobre el objeto. [↑](#footnote-ref-3)
4. Menú. View-Show Code Pad. [↑](#footnote-ref-4)