**PROGRAMACIÓN ORIENTADA A OBJETOS**

**Introducción. Clases y objetos.**

**2020-1**

**Laboratorio 1/6**

# OBJETIVOS

Desarrollar competencias básicas para:

1. Apropiar un paquete de clases revisando: diagrama de clases, documentación y código.
2. Crear y manipular un objeto. Extender y crear una clase.
3. Entender el comportamiento básico de memoria en la programación OO.
4. Investigar clases y métodos en el API de java[[1]](#footnote-0).
5. Utilizar el entorno de desarrollo de BlueJ
6. Vivenciar las prácticas XP :  *Planning*The project is divided into [iterations](http://www.extremeprogramming.org/rules/iterative.html). 

*Coding* All production code is [pair programmed](http://www.extremeprogramming.org/rules/pair.html).

# ENTREGA

 Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.

 Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios correspondientes.

 En el foro de entrega de avance deben indicar los logros y los problemas pendientes por resolver.

# 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?

**R:** Ofrece las clases Canvas, Circle, Rectangle y Triangle, en un Canvas puede haber al menos uno de cada una de las demás clases.

1. La **documentación**[[2]](#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? ¿qué atributos tiene la clase Circle? ¿cuáles métodos ofrece la clase Circle para que la figura cambie (incluya sólo el nombre)?

**R:**  -El paquete shapes tiene tiene las clases Canvas, Circle, Triangle.  
-Los atributos de Circle son: diameter, xPosition, yPosition, color, isVisible y PI, este último siendo el único público.

-Los métodos de Circle son: makeVisible, makeInvisible, moveRight,moveLeft, moveUp, moveDown, moveHorizontal, moveVertical, slowMoveHorizontal, slowMoveVertical, 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?. Con respecto a los métodos: ¿cuántos métodos tiene en total? ¿cuáles son privados?. ¿Quienes usan los componentes privados?

**R: -** 5 Atributos privados (diameter, xPosition, yPosition, color, isVisible) y 1 público (PI)

* 14 métodos en total, los métodos draw() y erase() son los métodos privados, makeInvisible, moveHorizontal, moveVertical, changeSize usan erase(), y makeVisible, moveHorizontal, moveVertical, slowMoveHorizontal, slowMoveVertical, changeSize y changeColor hacen uso de draw()

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

**R:** En la documentación no se ve la totalidad de los métodos, debe ser así por la privacidad de los mismos.

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

**R:** Indica que es un atributo que siempre tendrá ese valor para cualquier objeto circle() independientemente del valor de sus otros atributos.

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

**R:** Servir como una introducción de forma sencilla y dinámica a los objetos y clases en Java.

**Manipulando objetos. Usando opciones.**

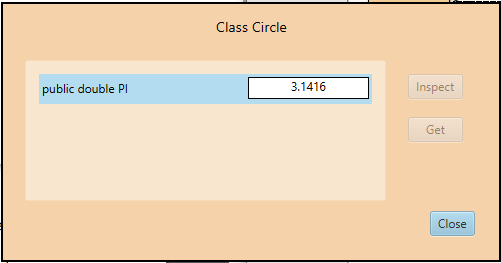
## [En lab01.doc]

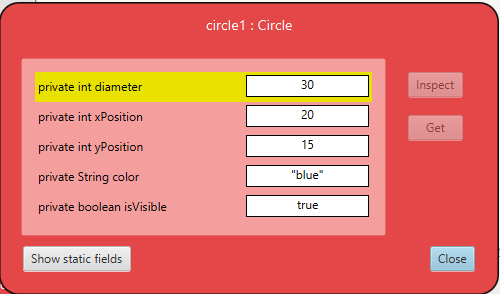
1. Creen un objeto de cada una de las clases que lo permitan[[3]](#footnote-2). ¿cuántas clases hay? ¿cuántos objetos crearon? ¿por qué?

RTA// Existen 3 clases triangule, circle y rectangle, y se crearon por lo tanto 3 objetos, un triángulo verde, un círculo rojo y un rectángulo azul porque los objetos creados con cada una de las clases que existían dándoles características se consideran objetos.

1. Inspeccionen el **estado** del objeto :Circle[[4]](#footnote-3), ¿cuáles son los valores de inicio de todos sus atributos? Capturen las pantallas

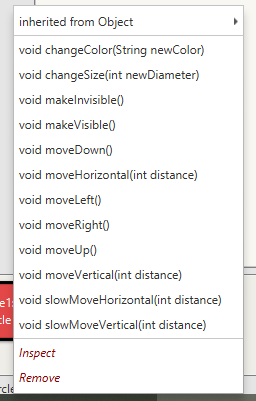
RTA//





1. Inspeccionen el **comportamiento** que ofrece el objeto :Circle5. Capturen la pantalla. ¿por qué no aparecen todos los que están en el código?

RTA// Porque existen métodos como erase y draw que son privados por lo tanto solo se utilizan dentro de la clase y nos los muestra.



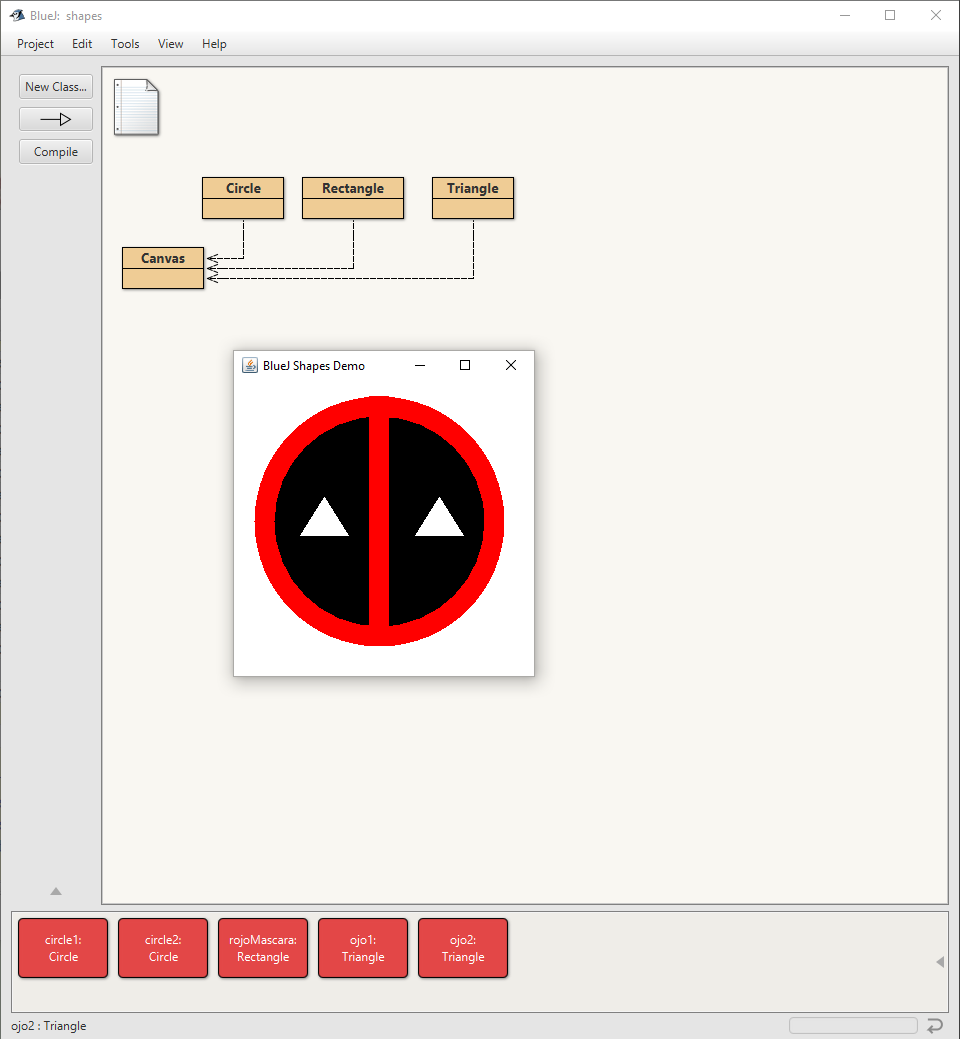
1. Construyan, con “shapes” sin escribir código, una propuesta de la imagen de su *comic* favorito. ¿Cuántas y cuáles clases se necesitan? ¿Cuántos objetos se usan en total? Capturen la pantalla.

RTA//se usaron las 4 clases existentes, se usaron 5 objetos para crear a deadpool.

imagen real

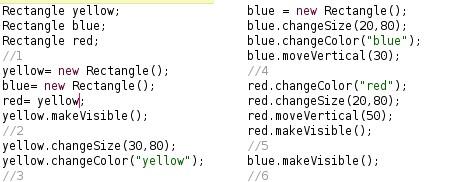


imagen creada



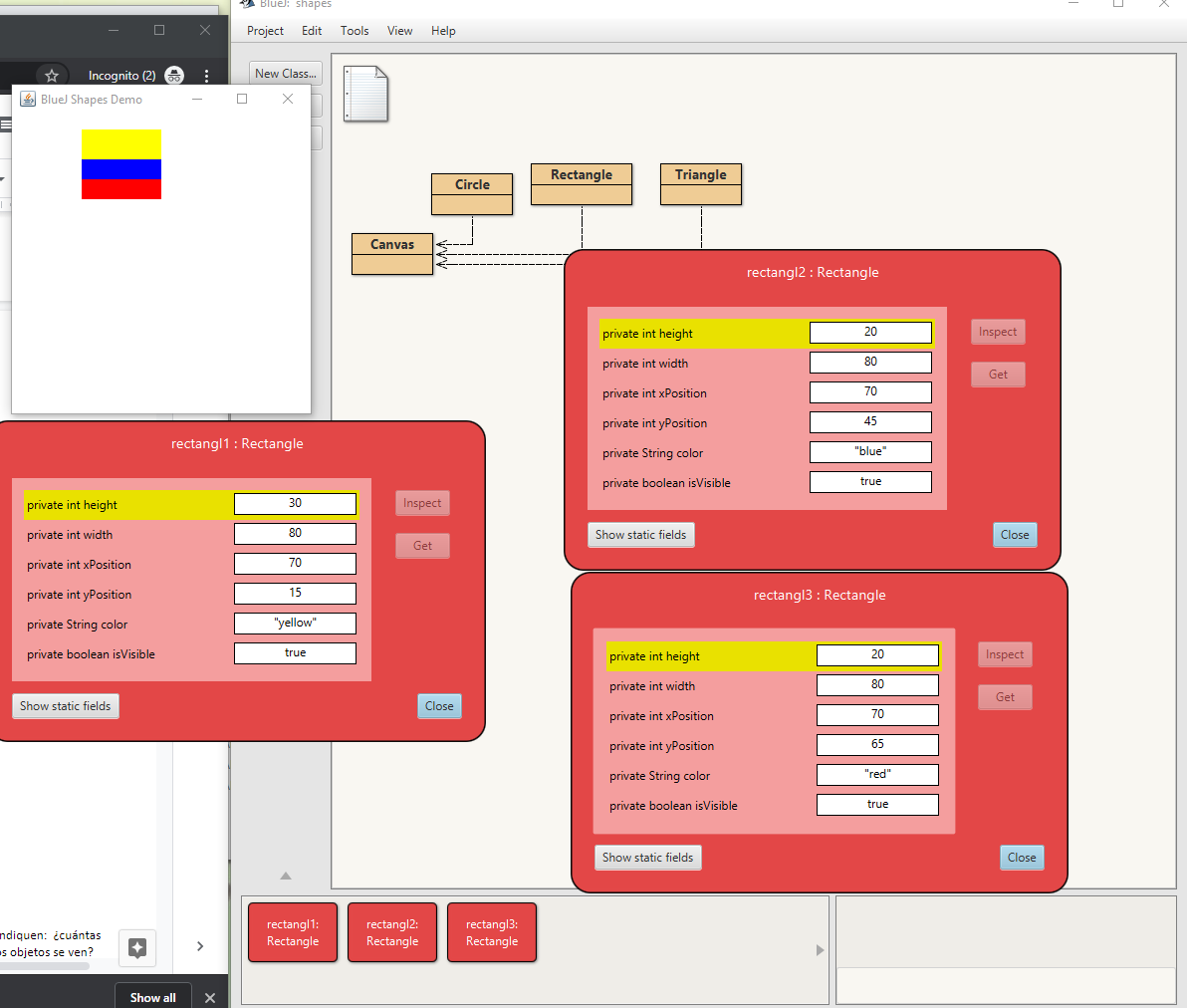
**Manipulando objetos. Analizando y escribiendo código.**

## [En lab01.doc]



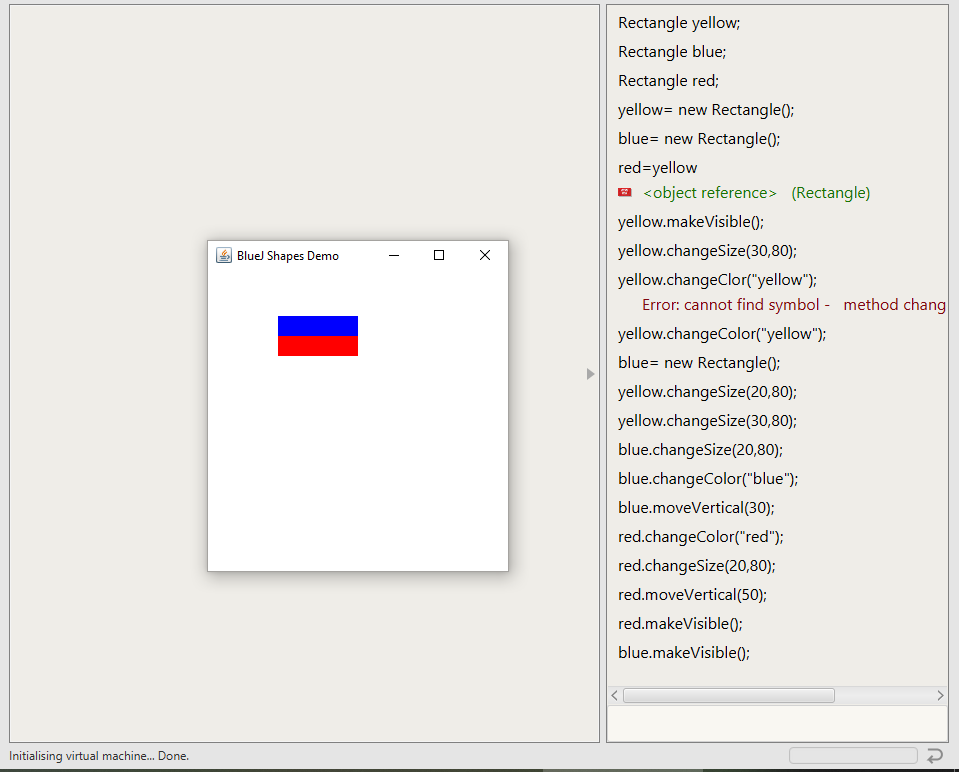
1. Lean el código anterior ¿cuál es la figura resultante? Píntenla.

RTA// se crea la bandera de colombia



1. Habiliten la ventana de código en línea[[5]](#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.

RTA// existen 3 Variables , existen 2 objetos ya que al asignar red a yellow se vuelven uno mismo estos dos rectángulos creados tienen color rojo y azul, se pueden ver 2 objetos.



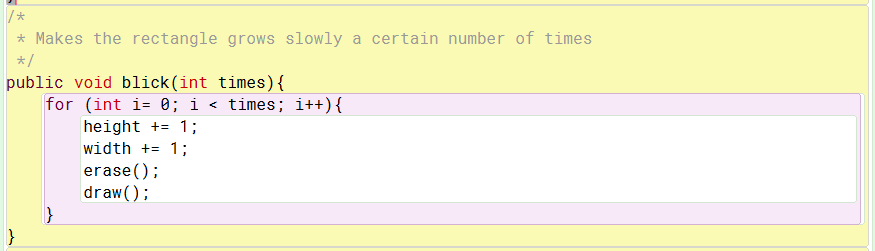
1. Es igual la figura pintada en 1. igual a la figura capturada en 2. , ¿por qué?

RTA// no es la misma imagen ya que en la primera parte se construyó desde el estado y se visualiza la bandera de Colombia, en cambio con el código se le asigna a un objeto otro igual y se crean 2 objetos solamente.

# Extendiendo clases

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

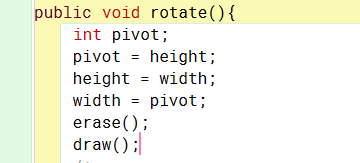
1. Desarrollen en Rectangle el método blick(times) (que hace que crezca lentamente el número dado de veces) . ¡Pruébenlo! grow up **HECHO**

****

1. Desarrollen en Rectangle el método perimeter(). ¡Pruébenlo! **HECHO**

****

1. Desarrollen en Rectangle el método rotate() (que hace que rote a la derecha transladándose) . ¡Pruébenlo!



1. Generen nuevamente la documentación y revise la información de estos nuevos métodos. Capture la pantalla.

|  |  |
| --- | --- |
| En este punto vamos a construir dos herramientas para los calculistas Mayas.  Los mayas utilizaban un sistema numérico de base 20; es decir, los “dígitos mayas” van de 0 a 19.  **XoX**  Esta clase nos permite repres  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. | XoX |

**Implementando una nueva clase. XoX.**

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

|  |  |  |
| --- | --- | --- |
|  |  | **Miniciclo 1** XooK getValue  **Miniciclo 2** makeVisible makeInvisible **Miniciclo 3** random setValue  **Miniciclo 4** moveHorizontal changeColor |

1. ¿Cuántos objetos conforman un Xook? ¿Cuáles son los diferentes tipos?

lo conforman 3 objetos, los triangulos,circulos y rectangulos

1. Clasifiquen los métodos en: constructores, analizadores y modificadores.

Constructores:Xook

Analizadores:getValue,random

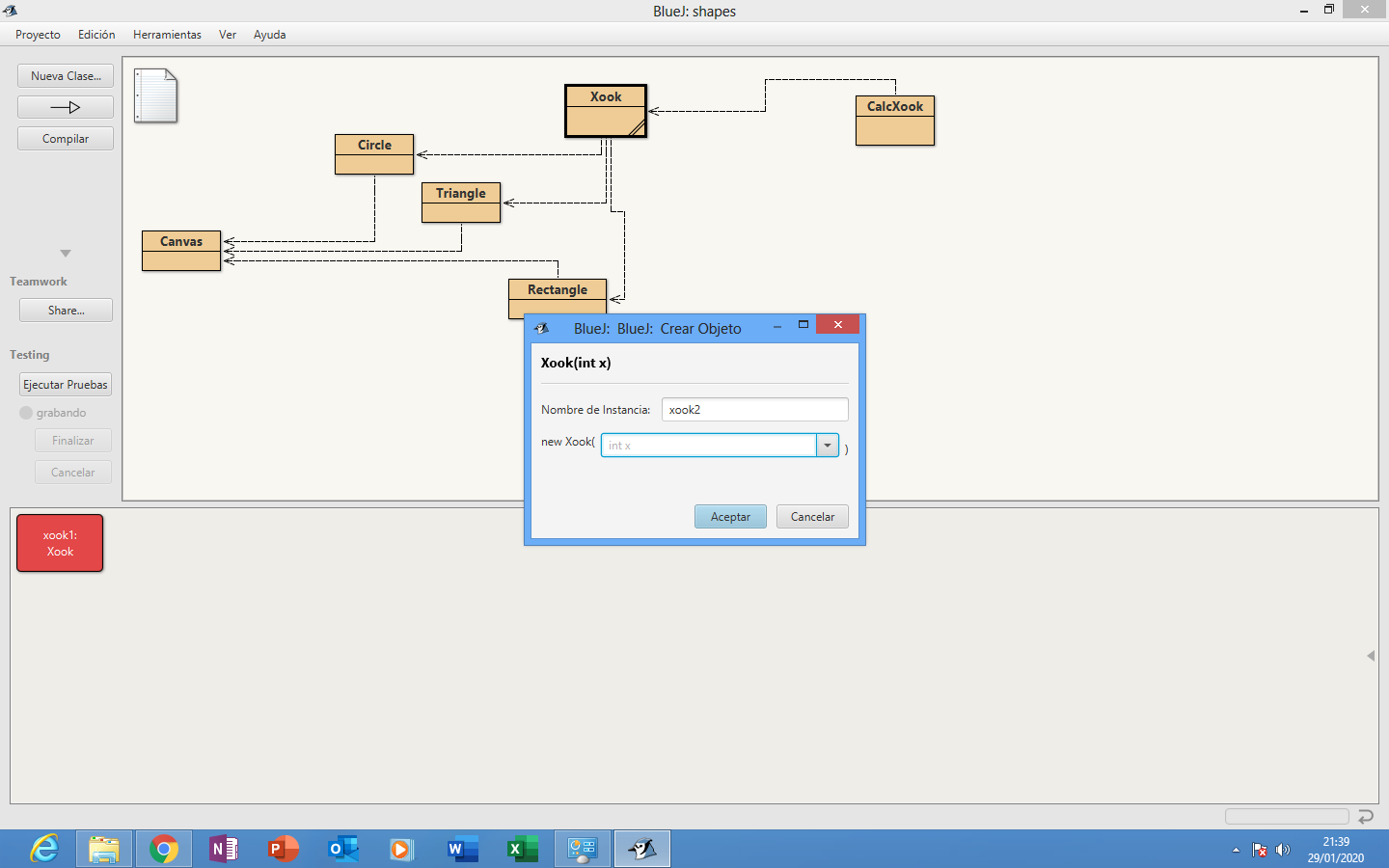
Modificadores:makeVisible,makeInvisible,moveHorizontal,changeColor

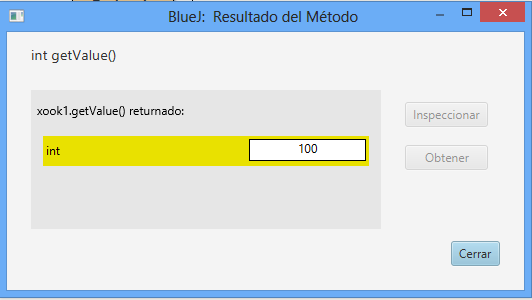
1. ¿Cuáles métodos requieren un prerrequisito? Explique su respuesta.

Xook,moveHorizontal,changeColor ya que a estos les ingresa un parámetro el cual deben modificar

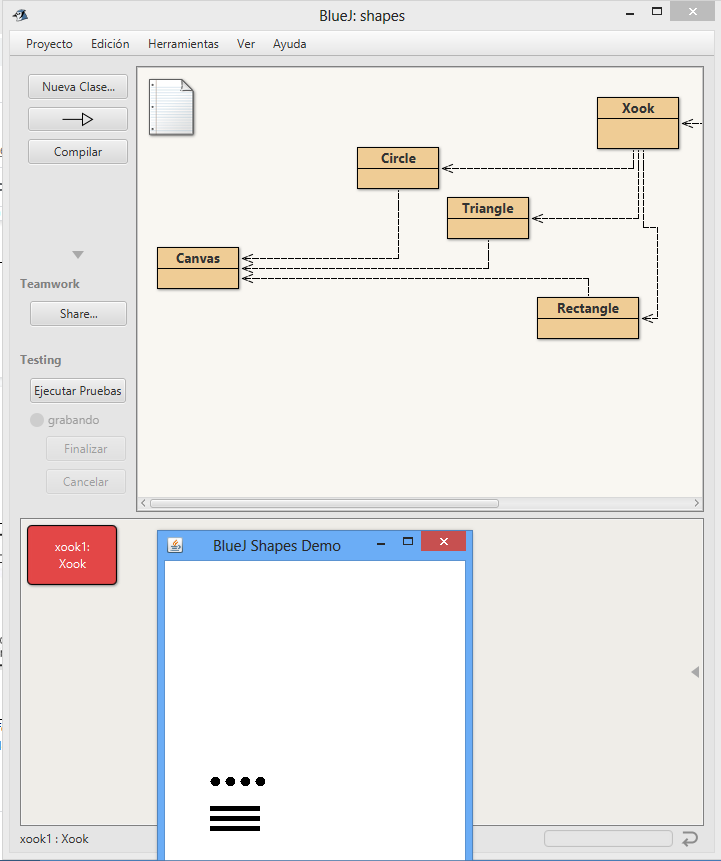
1. Desarrollen la clase Xook considerando los miniciclos. Al final de cada miniciclo realicen una prueba. Capturen las pantallas relevantes.

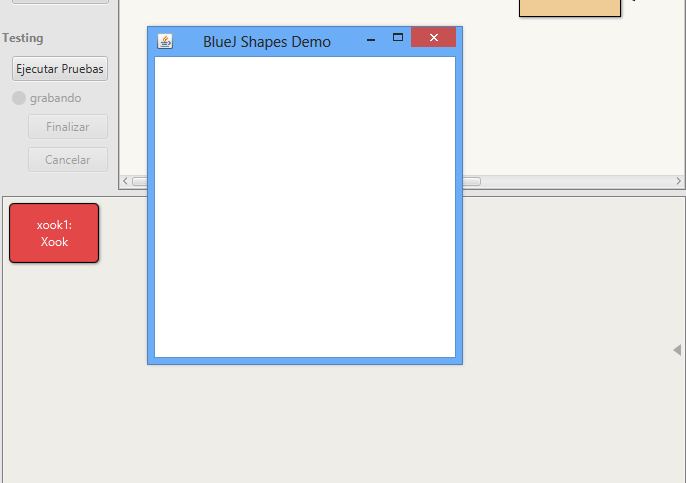
miniciclo1:



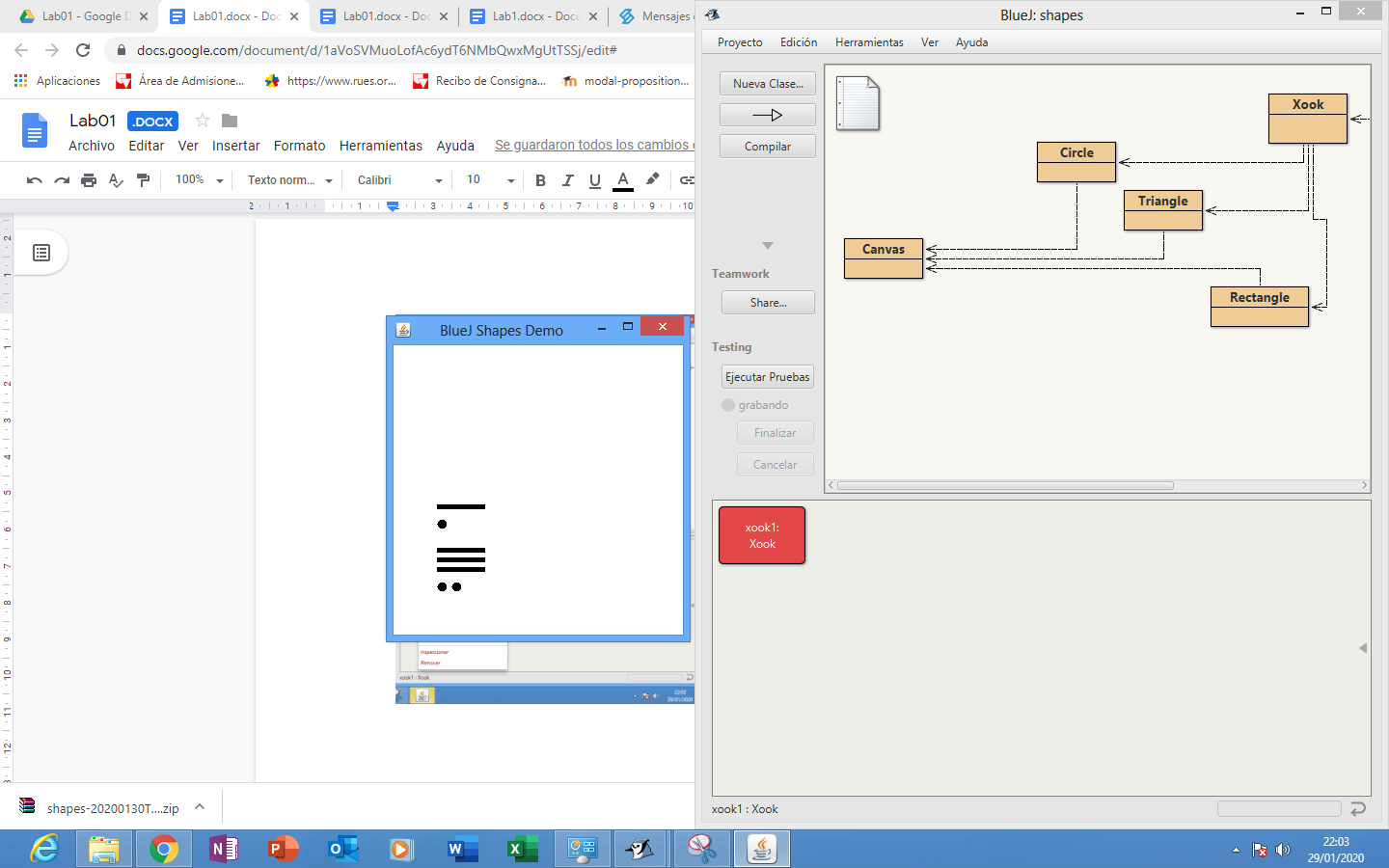
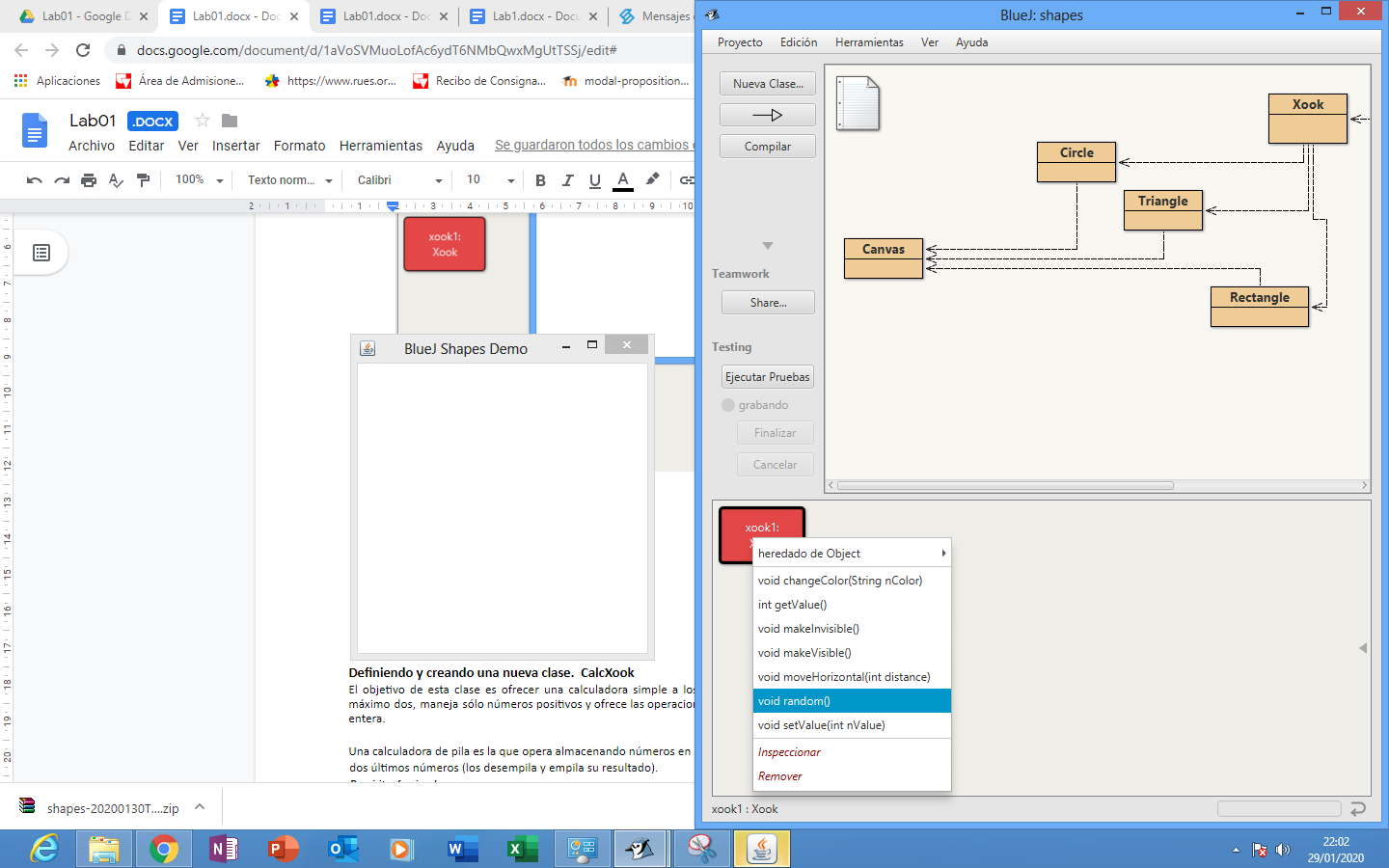


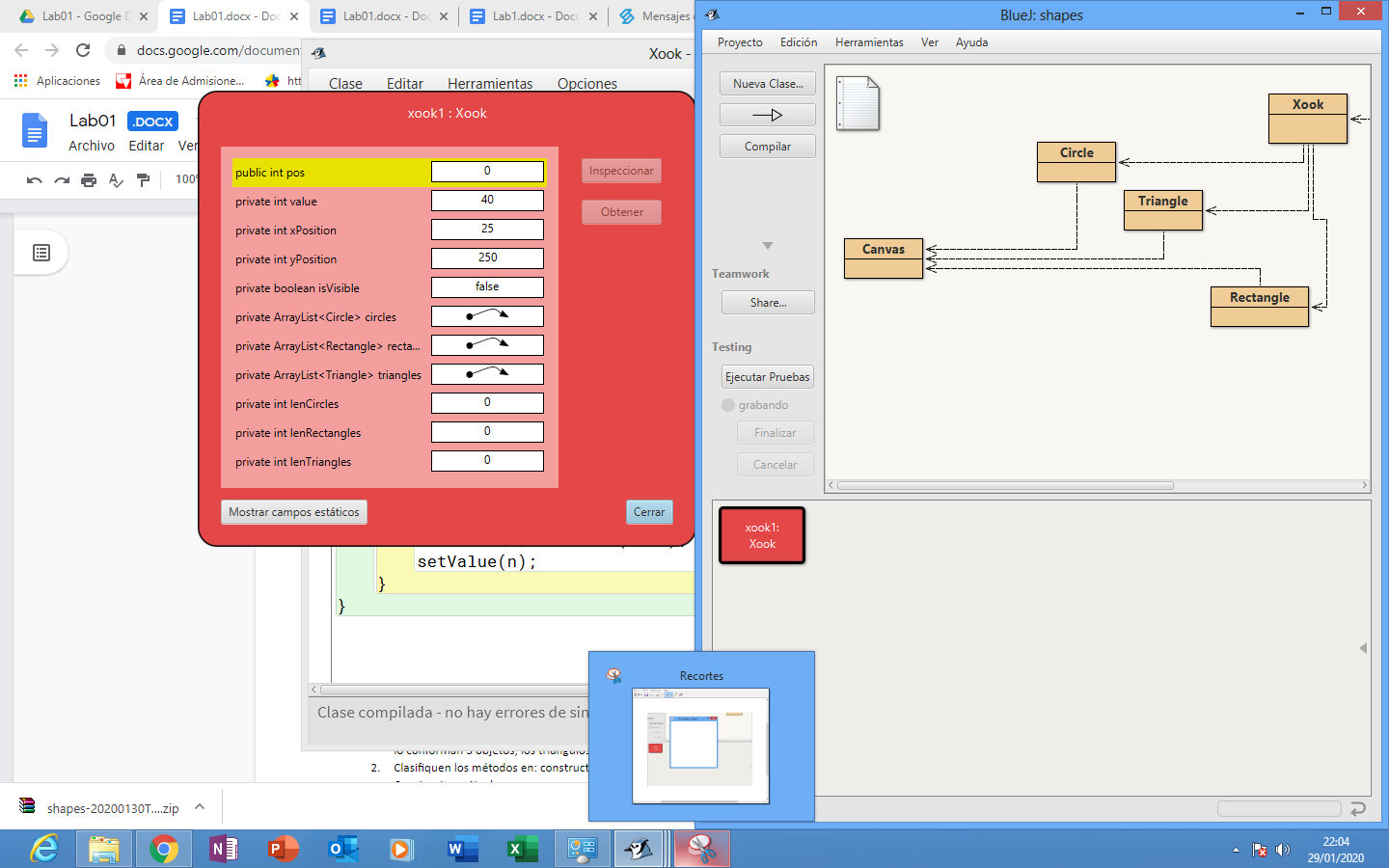
miniciclo2:



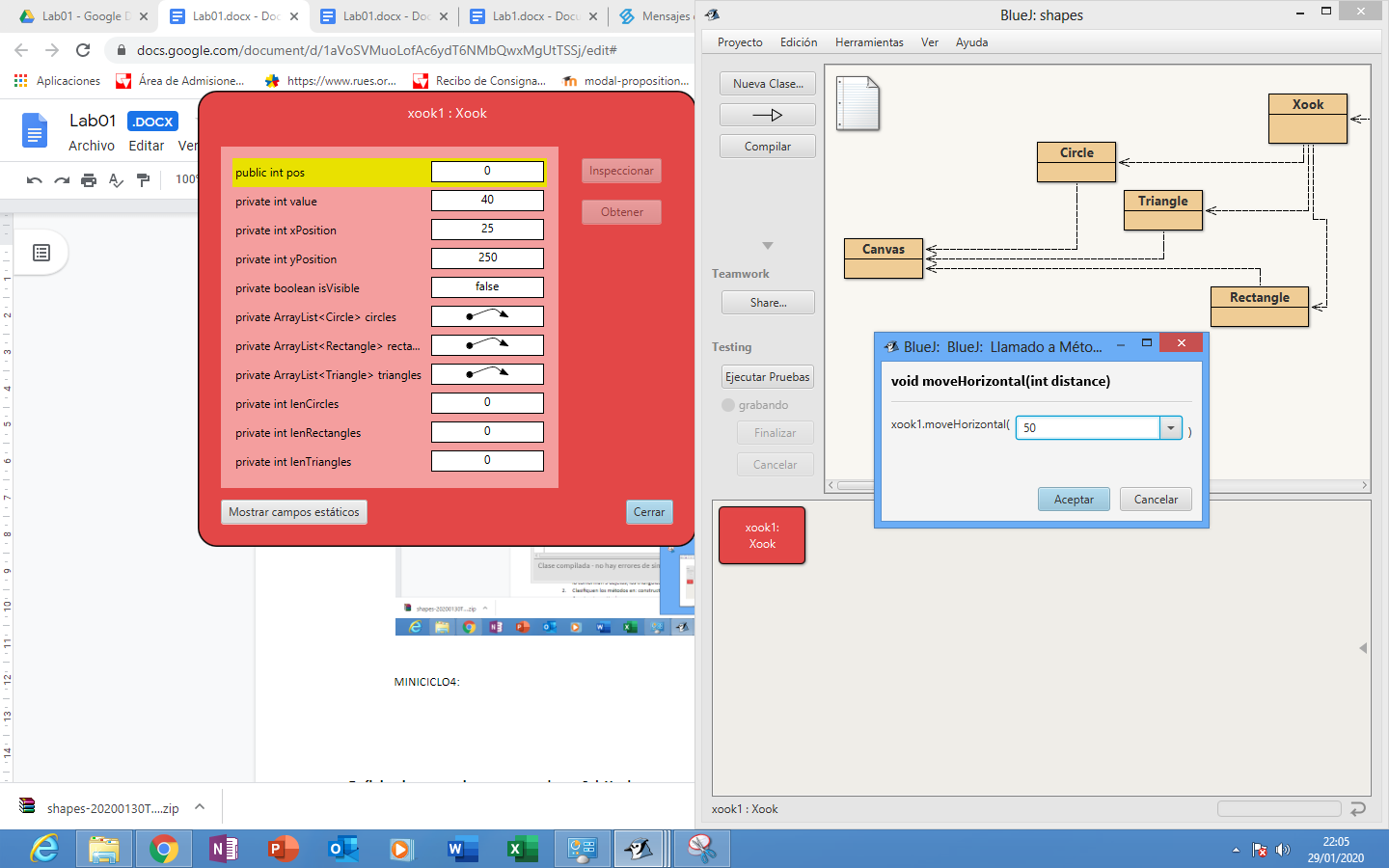


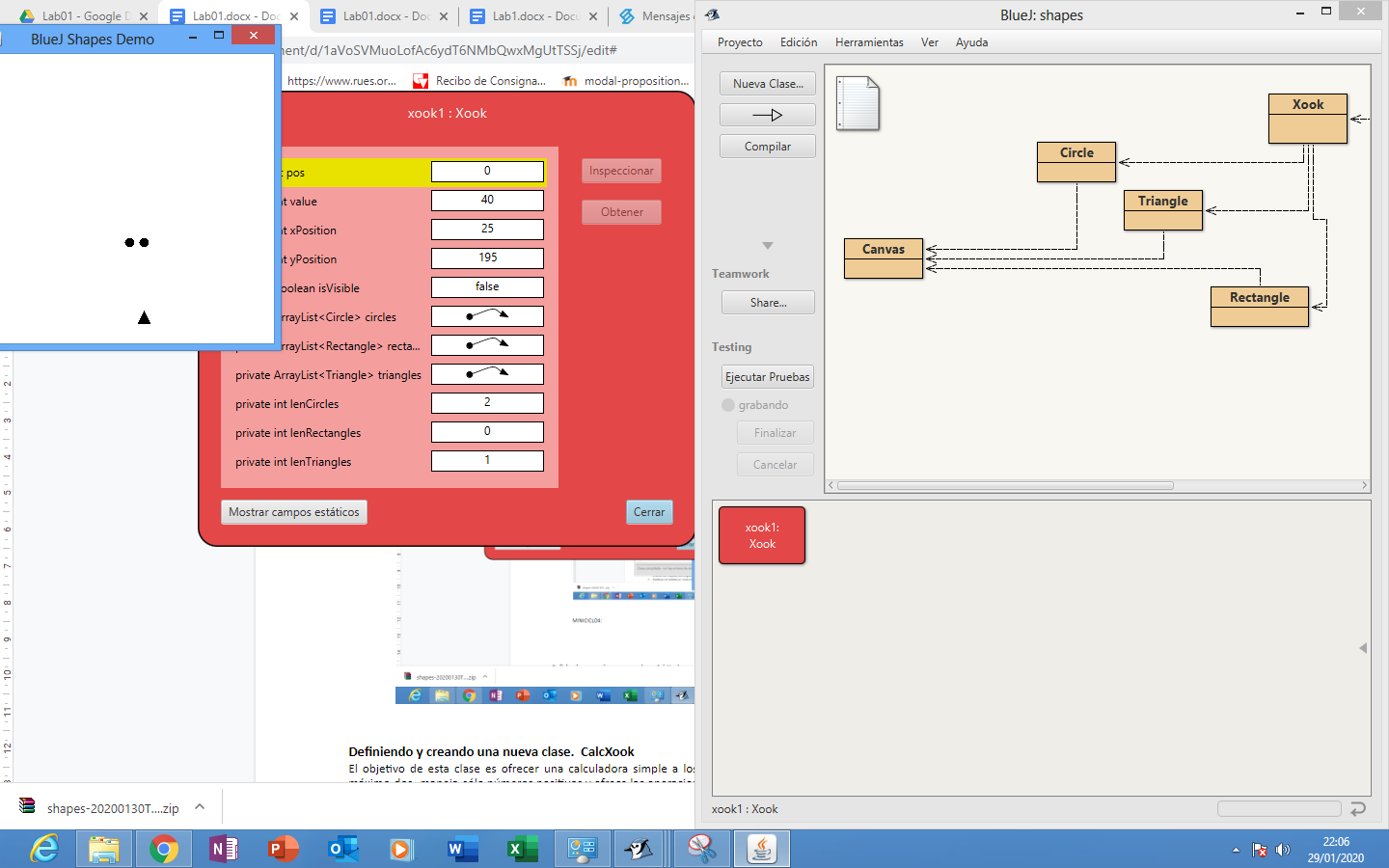
miniciclo3:

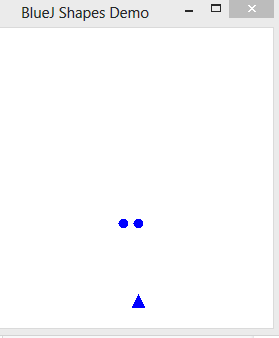




MINICICLO4:







# Definiendo y creando una nueva clase. CalcXook

El objetivo de esta clase es ofrecer una calculadora simple a los mayas. Esta calculadora es de pila de tamaño máximo dos, maneja sólo números positivos y ofrece las operaciones básicas: suma, resta, multiplicación y división entera.

Una calculadora de pila es la que opera almacenando números en la pila y para realizar las operaciones utiliza los dos últimos números (los desempila y empila su resultado).

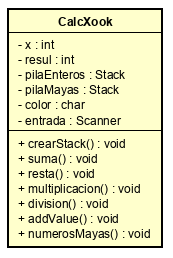
## Requisitos funcionales

* Permitir crear la calculadora, indicando el color.
* Permitir al usuario Introducir un número en decimal
* Realizar las operaciones básicas (+,-,\*)

## Requisitos de interfaz

* Las operaciones de la pila se ofrecen como métodos públicos de la clase CalcXook.
* La calculadora debe presentar los números de la pila en números mayas.
* 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 CalXook, es decir, definan los métodos que debe ofrecer.



1. Planifiquen la construcción considerando algunos miniciclos.

MINICICLOS:

1. CalcXook

crearStack

addValue

2.suma

resta

multiplicacion

3. numerosMayas

4. Crearerrores

1. Implementan la clase . Al final de cada miniciclo realicen una prueba de aceptación. Capturen las pantallas relevantes.
2. Indiquen las extensiones necesarias para reutilizar la clase CalXook. Explique.
3. Propongan un nuevo método para enriquecer el juego.

# Extendiendo una clase. CalcXook

**[En lab01.doc. CalcXoX.java]**

El objetivo es extender la calculadora maya para los matemáticos más avanzados.

## Nuevos requisitos

1. Permite trabajar con un número indefinido de operandos, no sólo dos.
2. Ofrece operaciones adicionales a las operaciones básicas.
3. Si está visible, muestra únicamente los diez números del tope de la pila.

# RETROSPECTIVA

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/ Hombre)

Fueron alrededor de 20 horas por hombre.

1. ¿Cuál es el estado actual del laboratorio? ¿Por qué?

El laboratorio se desarrollo exitosamente,faltaron detalles pero esto se debe gracias a pocol

1. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?

La práctica más útil fue la de programación a pares, ya que permite una complementación de ideas entre el equipo de trabajo.

1. ¿Cuál consideran fue el mayor logro? ¿Por qué?

Conocernos más como compañeros ya que vimos la forma en la que el otro trabaja.

1. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?

La implementación del “NumeroMayas” fue lo más difícil ya que no lo podiamos graficar bien, ya que el grafico tomaba otra ubicacion pero su pudo resolver utilizando set y get

1. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?

Supimos distribuir muy bien el trabajo y complementar nuestras ideas para el desarrollo del laboratorio, nos comprometemos a distribuir mejor el tiempo.

1. http://docs.oracle.com/javase/8/docs/api/ [↑](#footnote-ref-0)
2. Menu: Tools-Project Documentation [↑](#footnote-ref-1)
3. Clic derecho sobre la clase [↑](#footnote-ref-2)
4. Clic derecho sobre el objeto 5 Hacer clic derecho sobre el objeto. [↑](#footnote-ref-3)
5. Menú. View-Show Code Pad. [↑](#footnote-ref-4)