**PROGRAMACIÓN ORIENTADA A OBJETOS Construcción. Clases y objetos.**

**2025-1**

**Laboratorio 1/6**

# OBJETIVOS

Desarrollar competencias básicas para:

1. Apropiar un paquete 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-1).
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

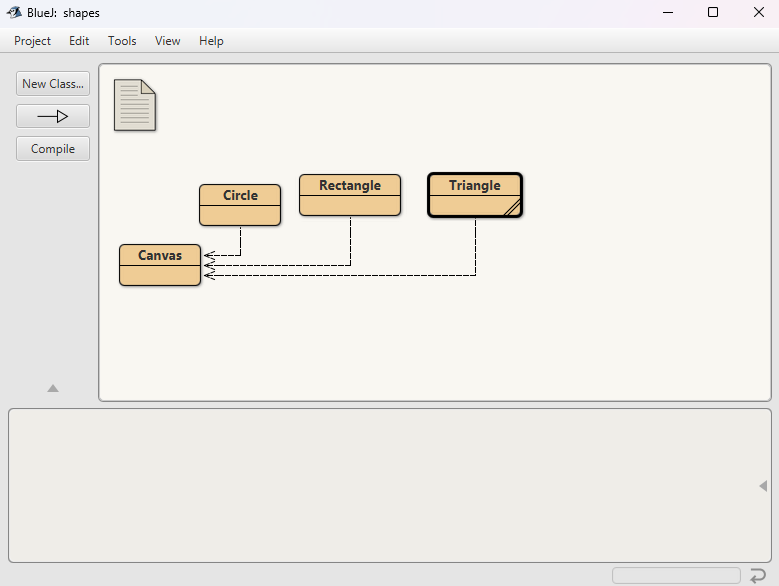
 Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.

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

**SHAPES**

# A. 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 BlueJ2. Capturen la pantalla.
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”

(a) ¿Qué clases ofrece?

* Ofrece las clases de “Circle”, “Rectangle”, “Triangle” y “Canvas”.

(b) ¿Qué relaciones existen entre ellas?

* La relación que tienen es que son figuras geométricas; además de esto las tres primeras se relacionan a “Canvas” lo que nos da a entender que esas figuras pueden verse en él.

1. La **documentación**[[2]](#footnote-2) presenta las clases del proyecto y, en este caso, la especificación de sus componentes públicos. De acuerdo con la documentación generada:

(a) ¿Qué clases tiene el paquete shapes?

* El paquete shapes tiene las clases, canvas, circle, rectangle, triangle.

(b) ¿Qué atributos tiene la clase Triangle?

* No lo dice en la documentación, pero se asumen como atributos: height, width, xPosition, yPosition, color y isVisible

(c) ¿Cuántos métodos ofrece la clase Triangle?

* Tiene 13 clases de ella misma y 11 heredados.

(d) ¿Qué atributos determinan el tamaño de un Triangle?

* newHigh, newWidth.

(e) ¿Cuáles métodos ofrece la clase Triangle para que la figura cambie su tamaño (incluya sólo el nombre)?

* changeSize

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

(a) ¿Cuántos atributos realmente tiene?

* Tiene 6: VERTICES, height, width, xPosition, yPosition, color, isVisible.

(b) ¿Quiénes pueden usar los atributos públicos?. Con respecto a los métodos:

* El atributo VERTICES es el único atributo público en la clase y puede ser usado por cualquier clase o instancia que tenga acceso a la clase Triangle. Al ser static, es accesible sin necesidad de crear una instancia de Triangle y puede ser referenciado directamente como Triangle.VERTICES.

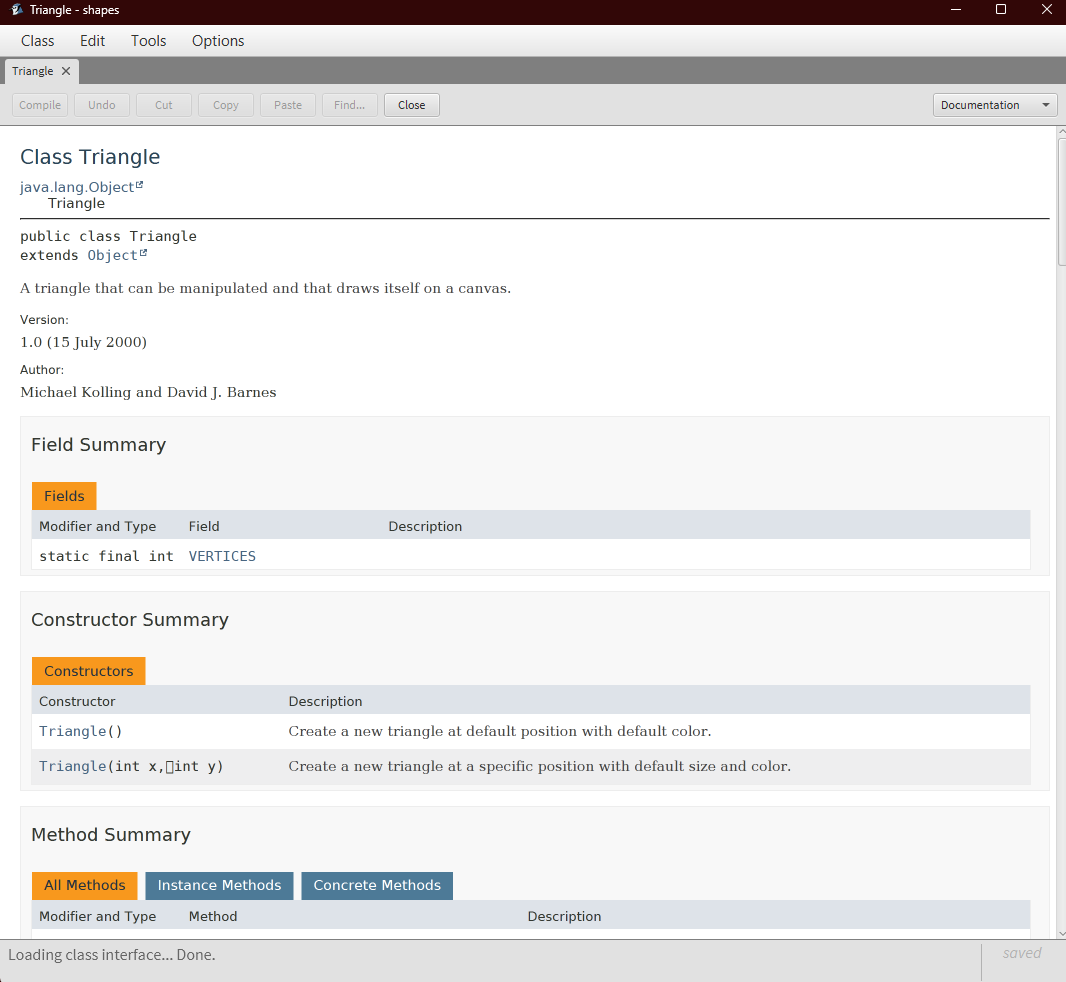
(c) ¿Cuántos métodos tiene en total?

* Tiene 14 métodos en total, 12 públicos y 2 privados.

(d) ¿Quiénes usan los métodos privados?

* draw(), erase()

(e) Desde el editor, consulte la documentación. Capture la pantalla.



1. Comparando la **documentación** con el **código**

(a) ¿Qué no se ve en la documentación?

* No se mencionan atributos, métodos privados, constantes y atributos estáticos y detalles sobre la implementación del lienzo.

(b) ¿por qué debe ser así?

* Debe ser así por: abstracción, encapsulamiento y reusabilidad y mantenimiento.

1. En el código de la clase Triangle, revise el atributo VERTICES
2. ¿Qué significa que sea public?

* Significa que puede ser accedido fuera de la clase TRIANGLE.

(b) ¿Qué significa que sea static?

* Es un atributo compartido por todas las instancias de la clase TRIANGLE.

(c) ¿Qué significa que fuera final? ¿Debe serlo?

* Si el atributo VERTICES fuera final, significaría que su valor no podría ser cambiado una vez asignado. Es decir, sería una constante. En este caso, sí debería ser final, ya que el número de vértices de un triángulo nunca cambia.

(d) Actualícenlo.

* public static final int VERTICES = 3;

1. En el código de la clase Triangle revisen el detalle del tipo del atributo height

(a) ¿Qué se está indicándo al decir que es int?.

* Se indica que high almacena valores tipo entero, En Java, el tipo int es un tipo de datos primitivo que puede almacenar números enteros en el rango de -2,147,483,648 a 2,147,483,647.

(b) Si fuera byte, ¿cuál sería el área del Triangle más grande posible?

* El tipo de datos byte en Java es un tipo de datos primitivo que puede almacenar valores enteros en un rango de -128 a 127. Dado que el atributo height es uno de los componentes que afectan el cálculo del área de un triángulo, sí height fuera un byte, el valor más grande posible para height sería 127.

(c) y ¿si fuera long?

* El tipo de datos long en Java puede almacenar números enteros en un rango mucho más grande que int (de -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807).

(d) ¿qué restricción adicional deberían tener este atributo?

* Al considerar el uso de height, deberíamos asegurarnos de que el valor de la altura sea siempre mayor que 0, ya que un triángulo no puede tener una altura negativa ni cero. Esto podría introducir una restricción lógica adicional sobre el valor mínimo de height.

La restricción adicional sería: height > 0.

(e) Refactoricen el código considerando

* if(newHeight >= 0 && newWidth>= 0){

height = newHeight;

width = newWidth;

}

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

* Hacer que el proceso de cambio de lenguaje digamos de Python a Java sea sencillo y compacto, sin necesidad de tanta teoría, sin embargo, también está la parte de mostrar cómo funcionan ciertas características de Java.

**B. Manipulando objetos. Usando un objeto.**

## [En lab01.doc]

1. Creen un objeto de cada una de las clases que lo permitan[[3]](#footnote-3).

(a) ¿Cuántas clases hay?

* 4 clases (triangulo, rectángulo, circulo, canvas)

(b) ¿Cuántos objetos crearon?

* 4, uno de cada uno

(c) ¿Quién se crea de forma diferente? ¿Por qué?

* El canvas porque tiene otros parámetros adicionales a los de las figuras geométricas. Además, que la forma de crear el canvas no es tan trivial como la creación de las figuras.

1. Inspeccionen los creadores de cada una de las clases.

(a) ¿Cuál es la principal diferencia entre ellos?

* La principal diferencia entre ellos es que el creador del canvas es mucho más explicativo a la hora de hacer comentarios, mientras que los que crearon las clases para las figuras no dicen tantos comentarios.

(b) ¿Qué se busca con la clase que tiene el creador diferente?

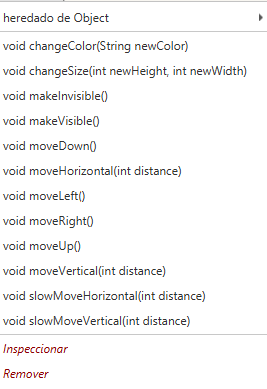
* Lo que se busca es hacer la creación del canvas, sin embargo, al tener tantos comentarios, probablemente es para que se pueda reusar el código del canvas para proyectos a futuro.

1. Inspeccionen el **estado** del objeto :Triangle[[4]](#footnote-4),
2. ¿Cuáles son los valores de inicio de todos sus atributos?

* **height:** 30
* **width:** 40
* **xPosition:** 140
* **yPosition:** 15
* **color:** "green"
* **isVisible:** false

1. Interfaz de usuario gráfica, Aplicación, Word

   Descripción generada automáticamenteCapturen la pantalla.
2. Inspeccionen el **comportamiento** que ofrece el objeto :Triangle[[5]](#footnote-5).
3. Capturen la pantalla.



1. ¿Por qué no aparecen todos los que están en el código?

* No aparecen dado que BlueJ solo muestra la parte del código que forma parte de la interfaz pública y el estado del objeto, dejando ocultos los detalles internos que no se espera que sean manipulados directamente.

1. Construyan, con “shapes” sin escribir código, una propuesta de la imagen del logo de de su chatbot IA favorito.
2. ¿Cuántas y cuáles clases se necesitan?

* Se necesitará un total de 5 clases, 4 círculos y 1 rectángulo.

1. ¿Cuántos objetos se usan en total?

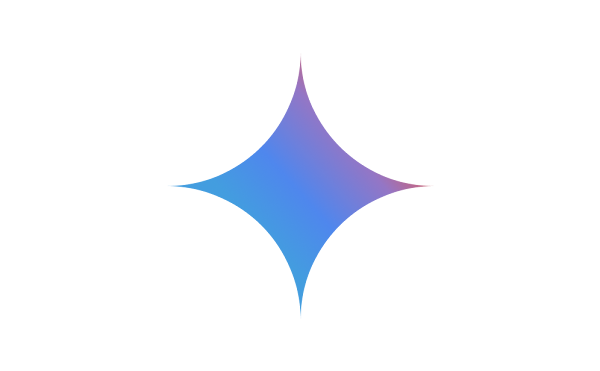
* En total se usaron 5 objetos.

(c) Capturen la pantalla.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

1. Incluyan el logo original.



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

## [En lab01.doc]

|  |  |
| --- | --- |
|  |  |

1. Lean el código anterior. (a) ¿cuál creen que es la figura resultante? (b) Píntenla.

* a) Creemos que da una especie de patrón de cuadrados, podría ser tal vez el símbolo antiguo de Windows.
* b) Gráfico, Gráfico en cascada

  Descripción generada automáticamente

1. Para cada punto señalado indiquen:

(a) ¿cuántas variables existen?

* a) Existen 4 variables: red, green, blue y yellow.

(b) ¿cuántos objetos existen? (no cuentan ni los objetos String ni el objeto Canvas)

* Existen 4 objetos.

(c) ¿qué color tiene cada uno de ellos?

* Los colores que tienen cada uno son:
  + **red**: Después de la asignación blue = red, su color se cambia a **azul**, ya que blue es solo una referencia a red.
  + **Green**: Su color sigue siendo **verde**.
  + **blue**: Tiene el color **azul**, debido a que blue apunta a red, lo que cambia el color de red a azul.
  + **yellow**: Su color es **amarillo**.

(d) ¿cuántos objetos se ven?

* d) Se ven un total de 3 objetos:
  + Green, yellow y “red”(Se ve azul).

1. Habiliten la ventana de código en línea, escriban el código. (a) Capturen la pantalla.

Gráfico, Gráfico en cascada

Descripción generada automáticamente

Al final, se ven 3 objetos visibles porque cuando asignamos blue = red, lo que estamos haciendo es hacer que blue apunte al mismo objeto que red. Entonces, al cambiar el color de blue a azul, estamos cambiando el color del objeto original, que es el mismo que red. Aunque red sigue existiendo, no es visible porque ahora blue tiene ese color. Por esa razón solo vemos 3 objetos en pantalla.

1. Compare la figura pintada en 1. con la figura capturada en 3.

(a) ¿son iguales?

* No, no son iguales. En la figura 1, utilizamos un método más sencillo, donde se crean diferentes rectángulos con opciones predefinidas, lo que permite trabajar directamente con la interfaz de cada objeto sin necesidad de asignar valores manualmente. En cambio, en la figura 2, el enfoque es más metódico, ya que hacemos asignaciones directas en la consola. Al decir red = blue, estamos indicando que red ahora será igual a blue, lo que significa que blue tomará todos los parámetros de red.

(b) ¿por qué?

* En la figura 1, los rectángulos se crean de manera independiente, utilizando métodos ya definidos, lo que permite modificar sus propiedades sin afectar a otros objetos. Cada rectángulo tiene su propio estado y se manipula por separado.

En la figura 2, al asignar red = blue, no estamos creando un nuevo objeto; en realidad, estamos haciendo que blue apunte al mismo objeto que red. Por lo tanto, cualquier cambio en blue afecta directamente a red, ya que ambos hacen referencia al mismo rectángulo en memoria. Esto explica por qué los colores y posiciones de los objetos en la figura 2 pueden no ser los mismos que en la figura 1, ya que al manipular blue también estamos afectando a red.

**D. Extendiendo una clase. Triangle.**

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

1. Interfaz de usuario gráfica

   Descripción generada automáticamenteDesarrollen en Triangle el método area(). ¡Pruébenlo! Capturen una pantalla.

* public int area(){

return (width \* height) / 2;

}

1. Desarrollen en Triangle el método equilateral() (transforma el triángulo en un triángulo equilatero de área equivalente ) .

¡Pruébenlo! Capturen dos pantallas.

public void equilateral(){

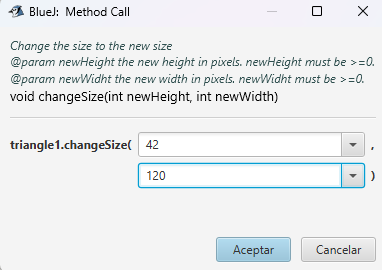
int originalArea = this.area();

double side = Math.sqrt((4 \* originalArea) / Math.sqrt(3));

int newHeight = (int) (side \* Math.sqrt(3) / 2);

int newWidth = (int) side;

this.changeSize(newHeight, newWidth); }

Forma

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteForma, Polígono

Descripción generada automáticamente

1. Desarrollen en Triangle el método shrink(times:int, height: int) (disminuye su tamaño **times** veces. Hasta llegar a una altura de **height**.) ¡Pruébenlo! Capturen tres pantallas.

A screenshot of a computer code

Description automatically generated

Forma

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamente con confianza bajaForma

Descripción generada automáticamente

1. Desarrollen en Triangle un nuevo creador que permita crear un triángulo en una posición específica. ¡Pruébenlo! Capturen una pantalla.

Forma

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteTexto

Descripción generada automáticamente con confianza media

1. Propongan un nuevo método para esta clase. Desarrollen y prueban el método.

* Propusimos el método de rotación del triángulo, usando la transformación de rotación en dos dimensiones, que se basa en la matriz de rotación denominada como:

Dado que lo queremos es rotar el ángulo θ alrededor del origen (0,0), entonces es un buen método para generar la rotación de nuestra figura que en este caso es un triángulo.

* Código en BlueJ:

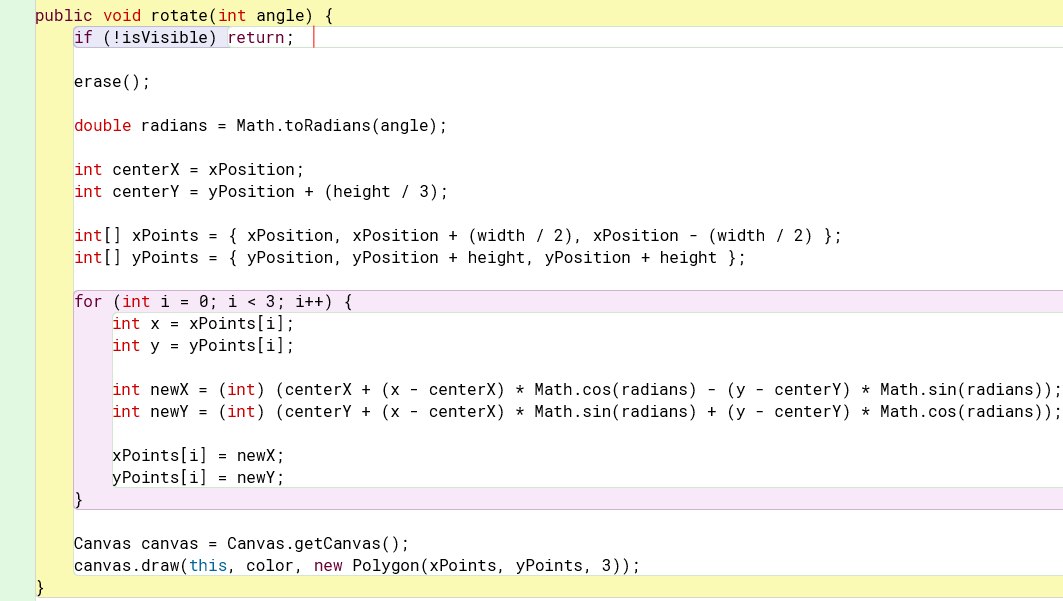


Imagen que contiene Forma

Descripción generada automáticamente

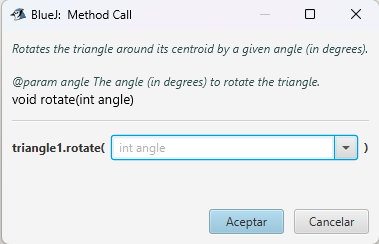
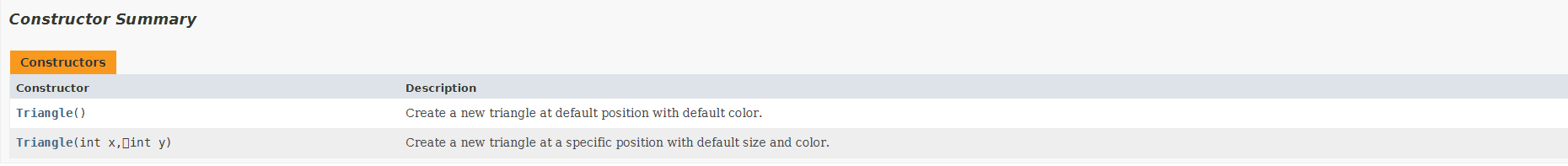


Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

1. Interfaz de usuario gráfica, Texto, Aplicación

   Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

   Descripción generada automáticamenteGeneren nuevamente la documentación y revise la información de estos nuevos métodos. Capturen la pantalla.

# E. Usando un paquete. shapes

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

En este punto vamos a crear huecos en cuadrados para guardar semillas. El diseño gráfico lo definen ustedes. Estos son algunos ejemplos.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Los huecos deben ofrecer los siguiente métodos

|  |  |
| --- | --- |
|  | **Mini-ciclo: 1**  \_(big):Pit putSeeds(seeds) removeSeeds(seeds)  seeds()  **Mini-ciclo: 2** makeVisible() makeInvisible()  **Mini-ciclo: 3** changeColors(pit,seeds)  moveTo(x,y) |

1. Inicie la construcción únicamente con los atributos. Justifique su selección. Adicione pantallazo con los atributos.

A screenshot of a computer

Description automatically generated

Los atributos utilizados también son los de rectángulo ya que hueco es un atributo de la clase Pit de tipo Rectangle.

A screenshot of a math problem

Description automatically generated

1. Desarrollen la clase considerando los 3 mini-ciclos. Al final de cada miniciclo realicen dos pruebas indicando su propósito. Capturen las pantallas relevantes.

**Mini-ciclo: 1**

A screenshot of a computer program

Description automatically generated

Pruebas

Creación de Pit

Se crea con cantidad de semillas = 0

A screenshot of a computer

Description automatically generated



Al agregar 5, la cantidad aumenta a 5

A screenshot of a computer

Description automatically generated



Al quitar 3, la cantidad queda en 2

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Mini-ciclo: 2

A screenshot of a computer

Description automatically generated

Pruebas

makeVisible

A screenshot of a computer

Description automatically generated

makeInvisible

A screenshot of a computer

Description automatically generated

Mini-ciclo: 3

A yellow screen with black and red text

Description automatically generated

Pruebas

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA green square on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

# F. De python a java

## [En lab01.doc]

En este punto vamos a evaluar dos recursos de apoyo para la transición de Python a Java en las encuestas preparadas con ese objetivo.

1. El video.
2. Los prompts.

# KALAH

A close up of a text

Description automatically generated

**G. Definiendo y creando una nueva clase. Kalah.**

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

El objetivo de este trabajo es programar una mini-aplicación para Kalah**.**

## Requisitos funcionales

* Crear el estado inicial
* Realizar los movimientos
* Reiniciar el juego
* Consultar el estado del juego. (Un mensaje con el número de semillas en cada almacén)
* Informar cuando alguién gana el juego. (Un mensaje de felicitación)

## Requisitos de interfaz

* Las casas se identifican por el jugador (‘N’orth, ‘S’outh) y el número de la casa (1 a 6).
* En caso que no sea posible realizar una de las acciones, debe generar un mensaje de error.
* Para los mensajes use JoptionPane. Diseñen la clase,es decir, definan los métodos que debe ofrecer.

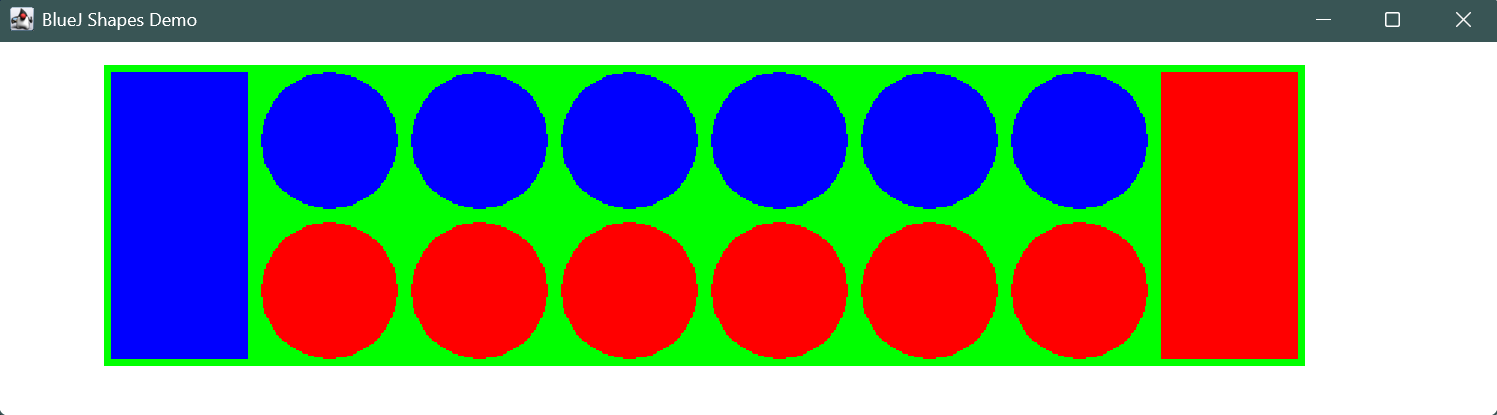
1. Diseñen la clase, es decir, definan los métodos que debe ofrecer.

2. Planifiquen la construcción considerando algunos mini-ciclos.

3. Implementen la clase . Al final de cada mini-ciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes.

**Mini-ciclo: 1**

* Creación del entorno gráfico



**Mini-ciclo: 2**

* Consultar estado del juego

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**Mini-ciclo: 3**

* Realizar movimientos

Tabla

El contenido generado por IA puede ser incorrecto.

* Reiniciar juego

Tabla

El contenido generado por IA puede ser incorrecto.Tabla

El contenido generado por IA puede ser incorrecto.

**Mini-ciclo: 4**

* **Informar cuando gana juego**

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

BONO. Nuevos requisitos funcionales. Kalah.

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

El objetivo de este trabajo es extender la mini-aplicación Kalah.

Nuevos requisitos funcionales

* Hacer un movimiento (la máquina decide). Explique la estrategia.
* Deshacer el último movimiento

1. Diseñen, es decir, definan los métodos que debe ofrecer.

2. Implementen los nuevos métodos. Al final de cada método realicen una prueba

indicando su propósito. Capturen las pantallas relevantes.

# RETROSPECTIVA

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

* 22 / Edgar Daniel Ruiz
* 22 / Juan Esteban Sánchez García

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

* El estado actual es terminado, menos el bono.

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

* La práctica de trabajo en parejas :D, consideramos que fue bastante útil para poder terminar el laboratorio.

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

* Poder corregir los errores a tiempo.

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

* El caso de que no supiéramos muchas cosas y nos tocara hacerlo a la fuerza.

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

* Trabajar juntos :p

1. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.

* No usamos referencias para este laboratorio.

1. http://docs.oracle.com/javase/8/docs/api/ 2 Use el comando de menú Project-Open. [↑](#footnote-ref-1)
2. Menu: Tools-Project Documentation [↑](#footnote-ref-2)
3. Clic derecho sobre la clase [↑](#footnote-ref-3)
4. Clic derecho sobre el objeto [↑](#footnote-ref-4)
5. Hacer clic derecho sobre el objeto. [↑](#footnote-ref-5)