**ººPROGRAMACIÓN ORIENTADA A OBJETOS**

Construcción. Clases y objetos.

2023-2

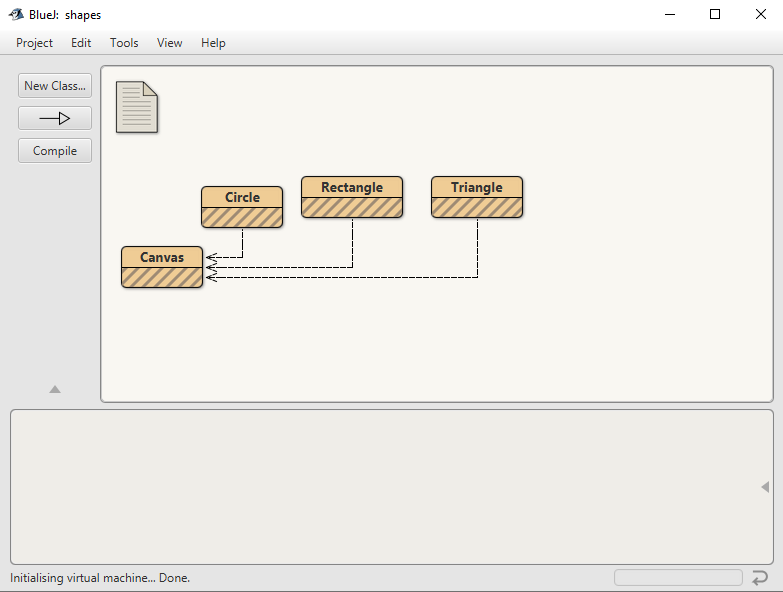
Laboratorio 1/6

**SHAPES**

1. **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. 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?**

Las clases que ofrece el diagrama de shape son Canvas, Circle, Rectangle, Triangle

**(b) ¿Qué relaciones existen entre ellas?**

La relación que existe es que Circle, Rectangle, Triangle usan métodos de la clase Canvas

**3.** La **documentación** 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?**

Canvas, Circle, Rectangle, Triangle



**(b) ¿Qué atributos tiene la clase Triangle?**

* height
* width
* xPosition
* yPosition
* color
* isVisible

**(c) ¿Cuántos métodos ofrece la clase Triangle?**

* 13, teniendo en cuenta los 12 métodos y el constructor

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

* changeSize()

**4.** 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?**

* 7

**(b) ¿Cuáles atributos determinan su forma?.**

* height
* width

Con respecto a los métodos:

**(c) ¿Cuántos métodos tiene en total?**

* 15

**(d) ¿Quiénes usan los métodos privados?**

* draw()
* erase()

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

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

No se ven los siguientes 6 seis atributos: height, width, xPosition, yPosition, color, isVisible.

No se ven los siguientes 2 métodos: draw(), erase()

**(b) ¿por qué debe ser así?**

No se observan en la documentación porque son privados

**6.** En el código de la clase Triangle, revise el atributo VERTICES

**(a) ¿Qué significa que sea public?**

Que sea público significa que pueden ser accedidos desde cualquier otra clase en el programa, permitiendo que se puedan llamar de manera directa y que se utilicen en otras partes del código.

**(b) ¿Qué significa que sea static?**

Que sea static significa que se crean elementos para la clase en sí y únicamente para la clase

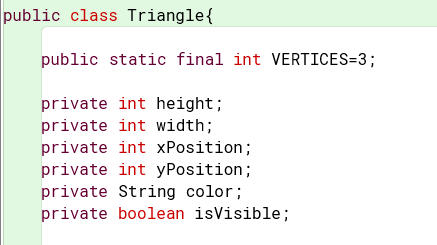
**(c) ¿Qué significaría que fuera final? ¿Debe serlo?**

Significa que no admitirá cambios después de su inicialización y asignación de valor. Si debería ser final ya que un triángulo siempre va a tener 3 vértices.

**(d) ¿De qué tipo de dato debería ser (byte, short, int, long)? ¿Por qué?**

Debería ser de tipo int. VERTICES representa el número de vértices en un triángulo, los triángulos tienen tres vértices. Ya que el valor de VERTICES siempre será 3 en este contexto, tiene sentido que sea de tipo int, que es un tipo de dato entero que puede almacenar valores numéricos sin decimales.

**(e) Actualízalo.**



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

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

Indica que el tipo de dato es entero

**(b) Sí height fuera byte, ¿cuál sería el triángulo más grande posible?**

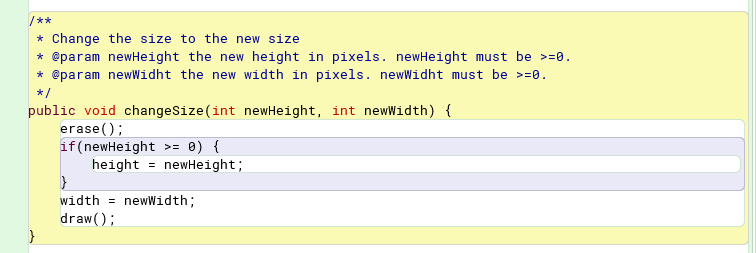
El triángulo más grande posible sería de 127

**(c) y ¿si fuera long?**

El triángulo más grande posible sería de (2^31)-1 = 2147483647 de altura

**d) ¿qué restricción adicional deberían tener este atributo? Refactorizar el código considerando (d).**

Una restricción adicional sería garantizar que la altura sólo acepte valores positivos



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

El propósito de shape es poder trabajar con figuras geométricas en pantalla

1. **Manipulando objetos. Usando un objeto.**

[En lab01.doc]

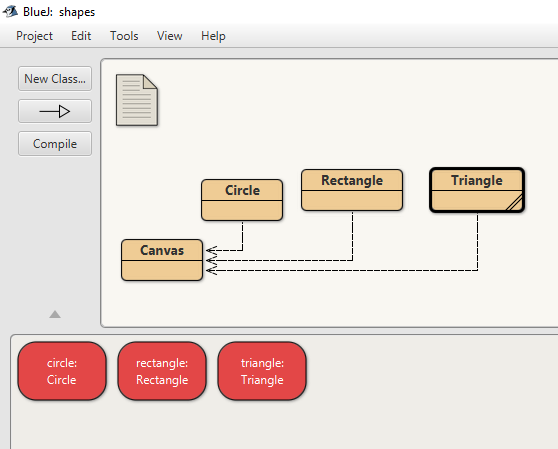
**1. Creen un objeto de cada una de las clases que lo permitan .**

**(a) ¿Cuántas clases hay?**

4, Canvas, Circle, Rectangle, Triangle

**(b) ¿Cuántos objetos crearon?**

3



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

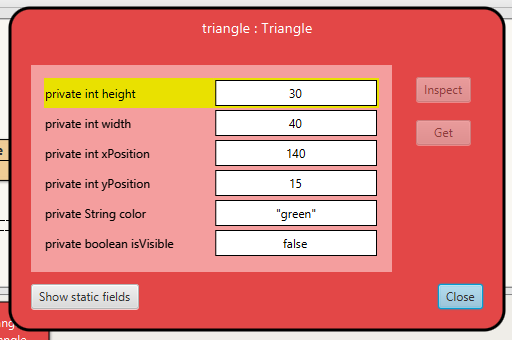
Canvas, porque hay clases que dependen de ella. Además, es donde los objetos se hacen visibles cuando se utiliza el método makeVisible.

**2. Inspeccionen el estado del objeto :Triangle,**

(a) ¿Cuáles son los valores de inicio de todos sus atributos?

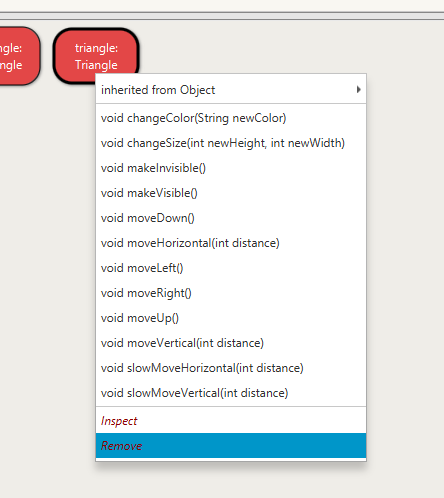
* Height: 30
* Width: 40
* xPosition: 140
* yPosition: 15
* color: “green”
* isVisible: false

(b) Capturen la pantalla.



**3. Inspeccionen el comportamiento que ofrece el objeto :Triangle.**

**(a) Capturen la pantalla.**



**(b) ¿Por qué no aparecen todos los que están en el código?**

No aparecen todos ya que solo se muestran los comportamientos que son públicos.

**4. Construyan, con “shapes” sin escribir código, una propuesta de la imagen del logo de su red social favorita.**

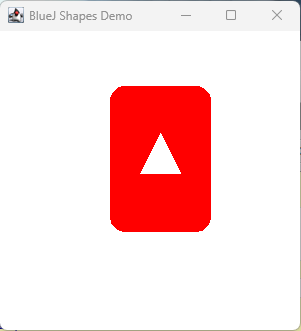
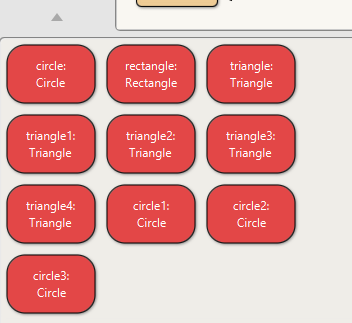
**(a) ¿Cuántas y cuáles clases se necesitan?**

3. canvas, rectangle, triangle

**(b) ¿Cuántos objetos se usan en total?**

10

**(c) Capturen la pantalla.**

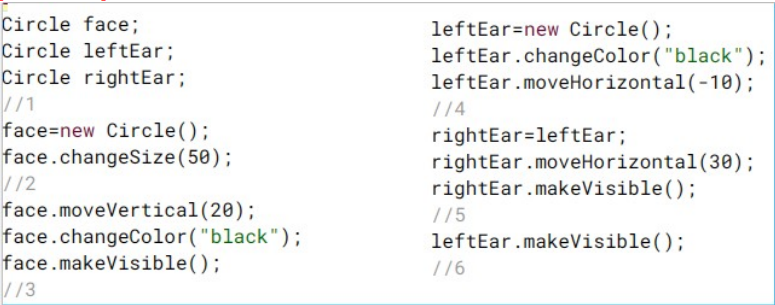
 

**(d) Incluyan el logo original.**



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

**[En lab01.doc]**

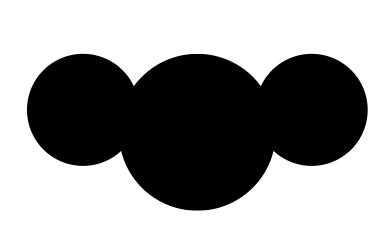
****

1. Lean el código anterior.

**(a) ¿cuál creen que es la figura resultante?**

un Mickey Mouse

**(b) Píntenla.**

****

2. Habiliten la ventana de código en línea, escriban el código. Para cada punto

señalado indiquen:

**(a) ¿cuántas variables existen?**

// 1 - 3 variables

// 2 - 3 variables

// 3 - 3 variables

// 4 - 3 variables

// 5 - 3 variables

// 6 - 3 variables

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

// 1 - 0 objetos

// 2 - 1 objeto

// 3 - 1 objeto

// 4 - 2 objetos

// 5 - 2 objetos

// 6 - 2 objetos

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

// 1 - sin color

// 2 - face = azul

// 3 - face = negro

// 4 - leftEar = azul y después negro

// 5 - face y rightEar negros

// 6 - face y rightEar negros

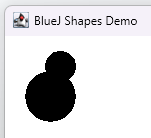
**(d) ¿cuántos objetos se ven?**

2

**Al final, (e) Expliquen sus respuestas.**

Cuando se asigna un objeto a otra variable, se está diciendo que la nueva variable ahora apunta al mismo objeto que la variable original. Esto significa que cualquier cambio que se haga en el objeto original también se refleja en la nueva variable.

**(f) Capturen la pantalla.**

****

**3. Compare la figura pintada en 1. con la figura capturada en 2.** ,

**(a) ¿son iguales?**

No

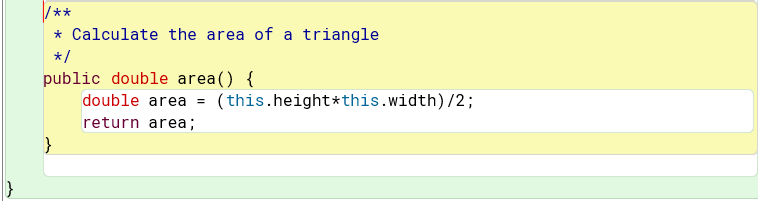
**(b) ¿por qué?**

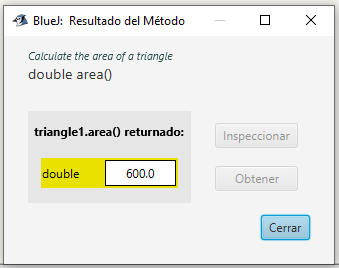
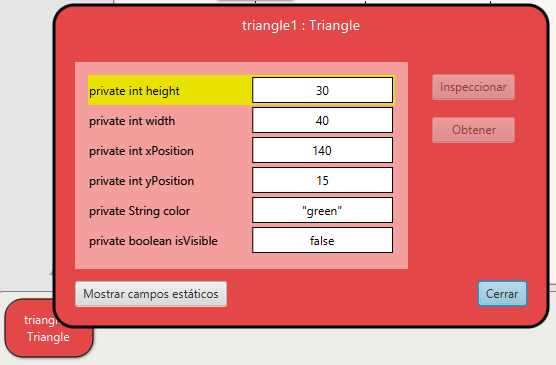
Cuando se asigna un objeto a otra variable, se está diciendo que la nueva variable ahora apunta al mismo objeto que la variable original. Esto significa que cualquier cambio que se haga en el objeto original también se refleja en la nueva variable.

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

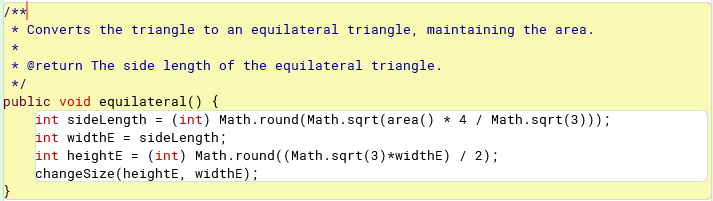
[En lab01.doc y \*.java]

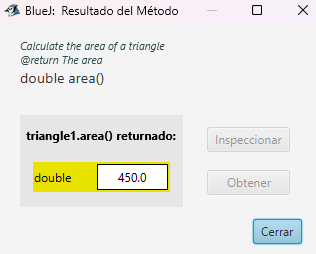
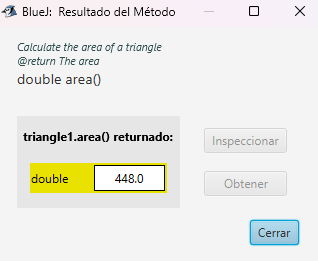
**1. Desarrollen en Triangle el método área(). ¡Pruébenlo! Capturen una pantalla.**





**2. Desarrollen en Triangle el método equilateral() (convierte el triángulo en un equilátero manteniendo el área) . 7 ¡Pruébenlo! Capturen dos pantallas.**



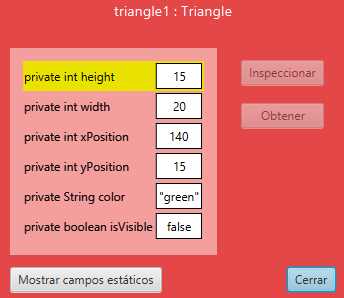
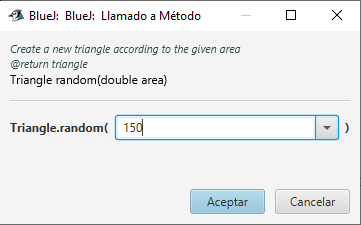
 equilateral() -> 

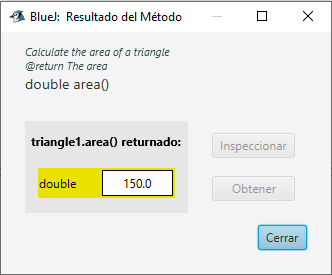
Se pierden algunos decimales en la conversión de tipo de datos

**3. Desarrollen en Triangle el método blink(times:int) (si times>0, cambia de color el número de veces indicado (Por ejemplo, original – nuevo – original serían dos veces. El color nuevo se escoge al azar entre los disponibles 8 ) . ¡Pruébenlo! Capturen dos pantallas.**

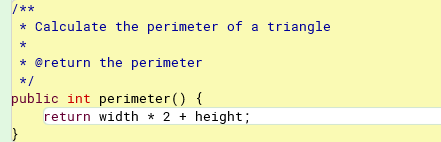
<https://youtube.com/shorts/ZWMqkyKNh7U?feature=share>

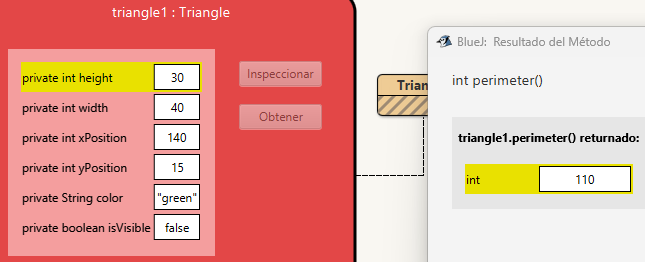
**4. Desarrollen en Triangle un nuevo creador que dada el área crea un triángulo al azar con dicha área.**



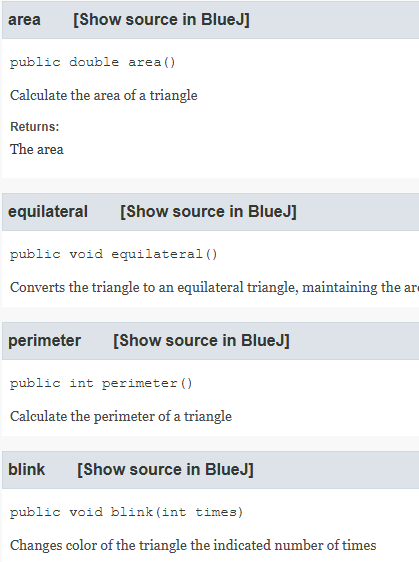


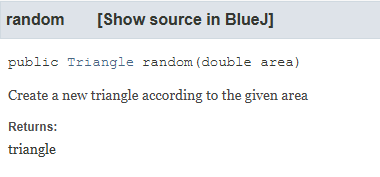
**5. Propongan un nuevo método para esta clase. Desarrolle y prueban el método**.





**6. Generen nuevamente la documentación y revise la información de estos nuevos métodos. Capturen la pantalla.**

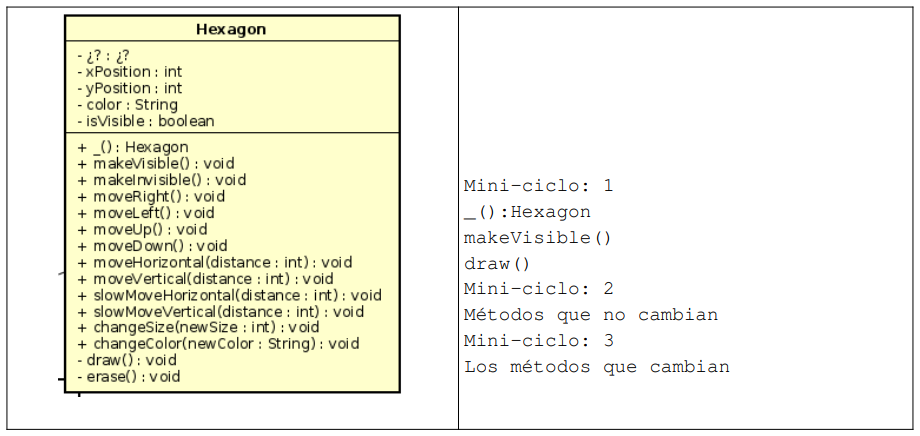




**E. Extendiendo un paquete. shapes + Hexágono**

[En lab01.doc y \*.java]

En este punto vamos a adicionar una nueva forma al paquete shapes: un hexágono regular. Los hexágono regulares tienen todos sus lados iguales.



**1. Para la construcción, tomen como base la clase Triangle inicial.**

**(a) ¿Qué métodos podrían quedar iguales?**

Los métodos que podrían quedar iguales son draw(), erase(), changeSize(), changeColor(), makeVisible(), makeInvisible(), moveHorizontal(), moveVertical(), moveRight(),moveLeft(), moveUp(), moveDown(), slowMoveHorizontal(), slowMoveVertical()

**(b) ¿Qué atributos podrían quedar iguales?**

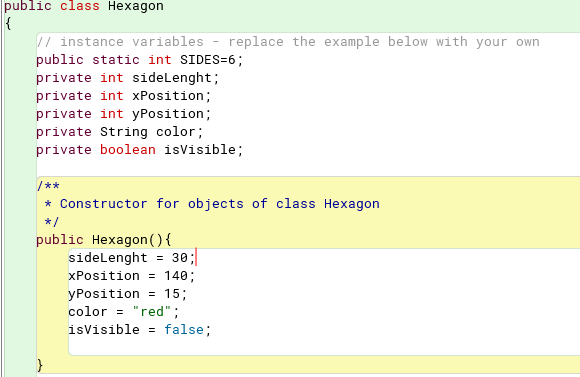
xPosition, yPosition, color, isVisible

**(c) Justifiquen su respuesta.**

Los métodos y atributos mencionados son generales y se aplican a todos los tipos de formas. No dependen de la forma específica de la forma, por lo que pueden ser utilizados por la clase Triangle y la clase Hexagono Regular.

**2. Inicie la construcción de la clase Hexagon únicamente con los atributos.**

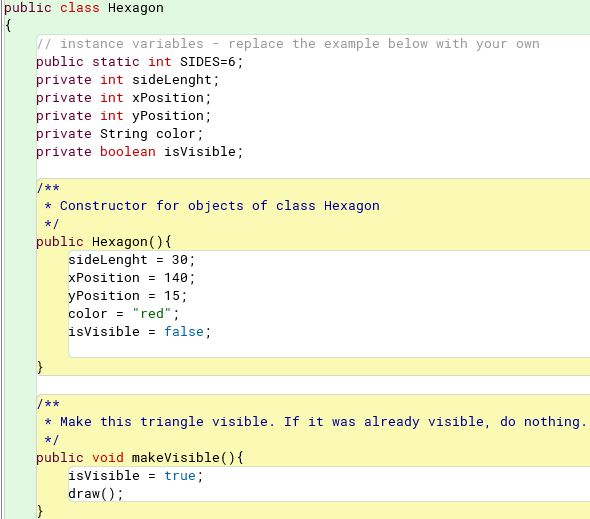
**(a) Adicione pantallazo con los atributos.**

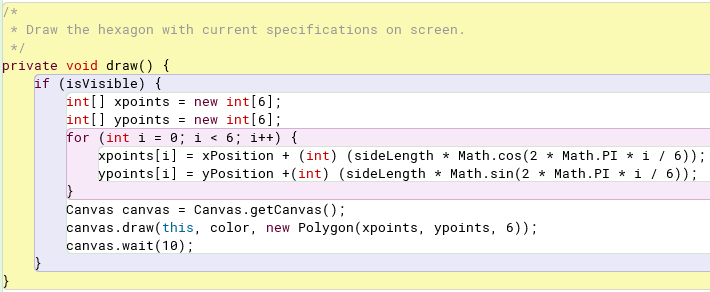


3. Desarrollen la clase Hexagon considerando los 3 mini-ciclos. Al final de cada miniciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes.

**Mini-ciclo: 1**

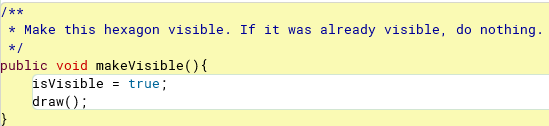
**\_():Hexagon makeVisible() draw()**

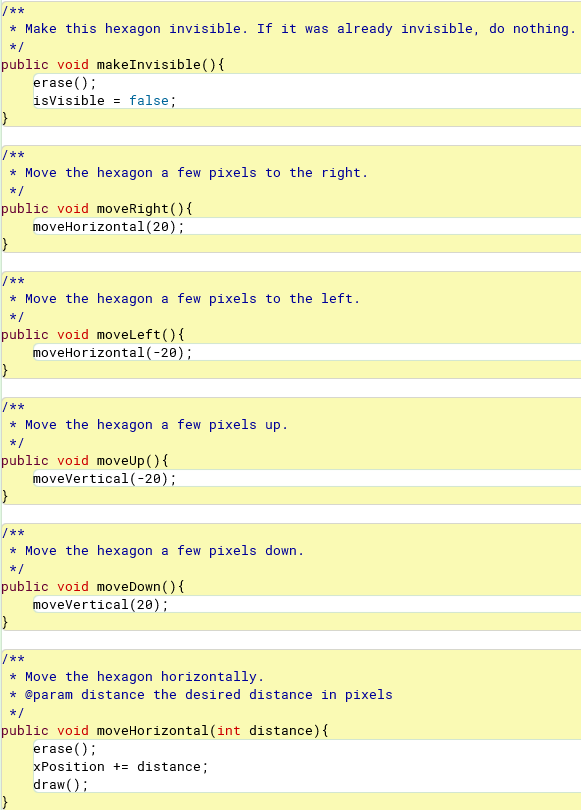


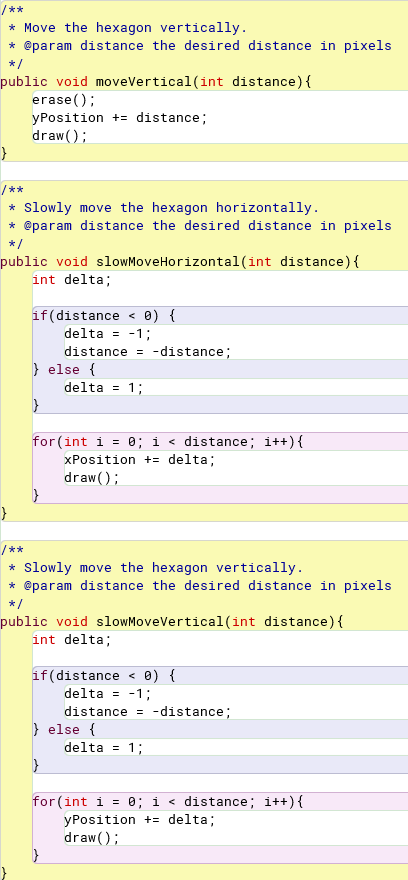


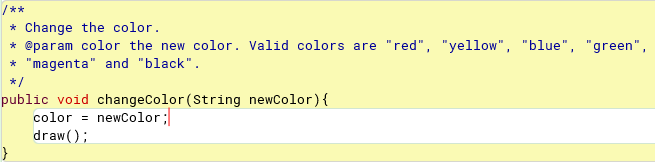
**Mini-ciclo: 2**

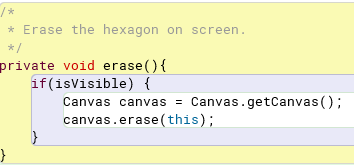
**Métodos que no cambian**

****

****

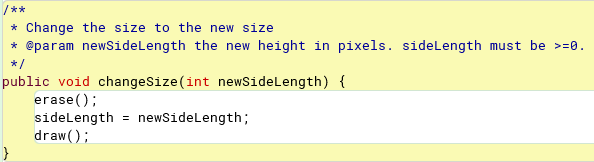
****

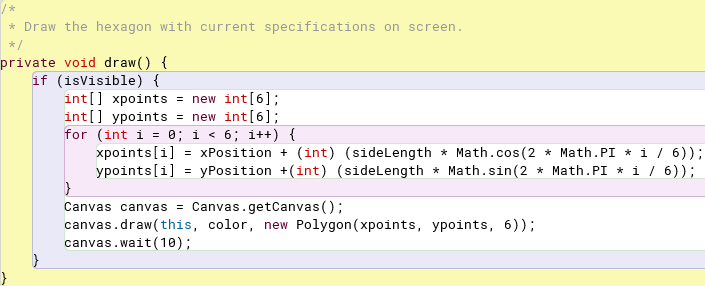
****

****

**Mini-ciclo: 3**

**Los métodos que cambian**

****

****

**F. Definiendo y creando una nueva clase. Replicate.**

[En lab01.doc. SelfAssembly.java]

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

**Requisitos funcionales**

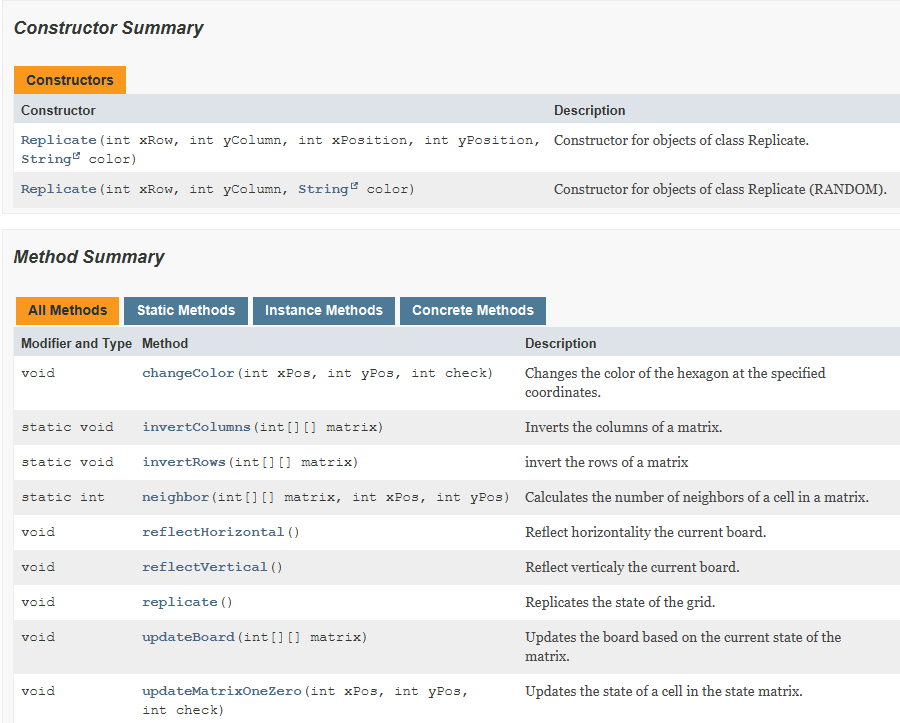
* Crear un replicate indicando su dimensión.
* Llenar una celda. Deben ofrecer dos formas de crear la pieza: (a) dada su posición (b) al azar
* Replicar. Debe seguir las reglas.
* Reflejar vertical u horizontalmente el replicate

**Requisitos de interfaz**

* Las celdas se identifican por su fila y su columna.
* En caso que no sea posible realizar una de las acciones, se debe generar un mensaje de error. Use JoptionPane.

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

11 métodos



1. **Planifiquen la construcción considerando algunos mini-ciclos.**

***Mini-ciclo: 1***

* Se genera el tablero con hexagonos, y se pintan implementando las dos formas (a) dada su posición (b) al azar
  + Replicate(int xRow, int yColumn, int xPosition, int yPosition, String color)
  + Replicate(int xRow, int yColumn, String color)
* changeColor(int xPos, int yPos, int check)

***Mini-ciclo: 2***

* Replicate
  + neighbor(int[][] matrix, int xPos, int yPos)
  + updateMatrixOneZero(int xPos, int yPos, int check)
  + replicate()

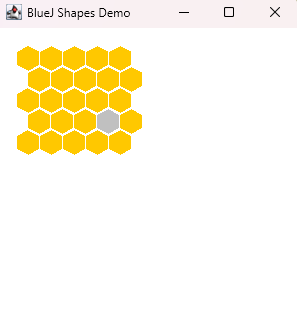
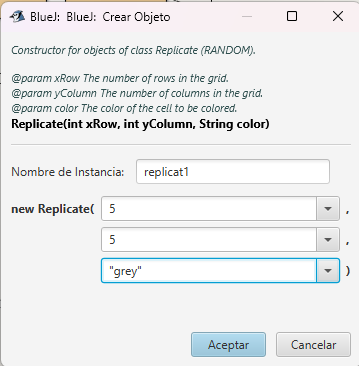
***Mini-ciclo: 3***

* invertColumns(int[][] matrix)
* invertRows(int[][] matrix)
* newBoard(int[][] matrix)
* reflectHorizontal()
* reflectVertical()

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

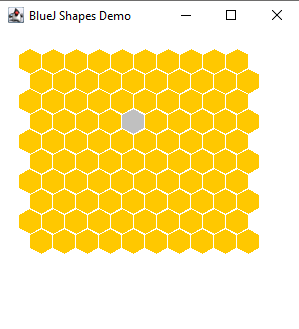
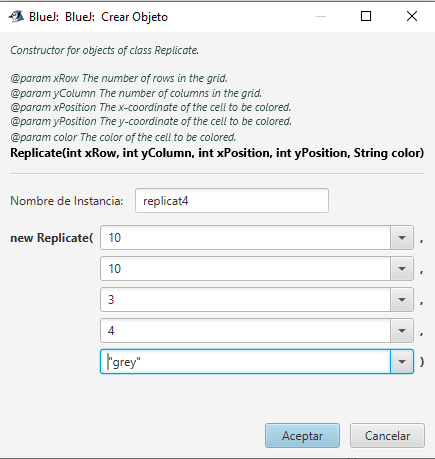
***Mini-ciclo: 1***

* Replicate(int xRow, int yColumn, String color)
* changeColor(int xPos, int yPos, int check) → se implementa en Replicate



*Se genera el tablero con hexágonos, y se pintan dada su posición*

* + Replicate(int xRow, int yColumn, int xPosition, int yPosition, String color)

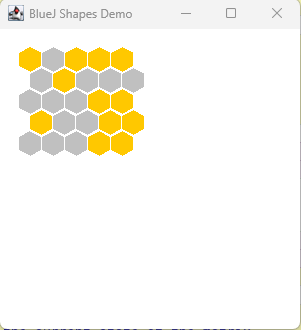
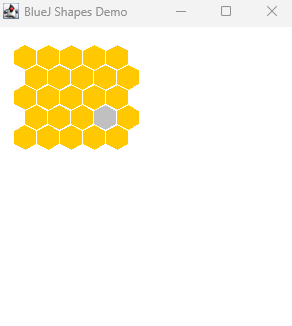


*Se genera el tablero con hexágonos, y se pintan al azar*

***Mini-ciclo: 2***

* Replicate
  1. neighbor(int[][] matrix, int xPos, int yPos)
  2. updateMatrixOneZero(int xPos, int yPos, int check)
  3. replicate()

**Estado inicial Estado final**

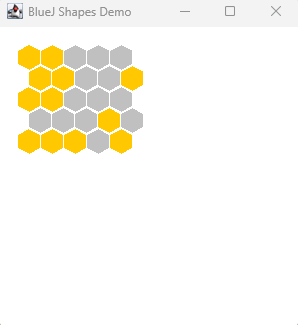
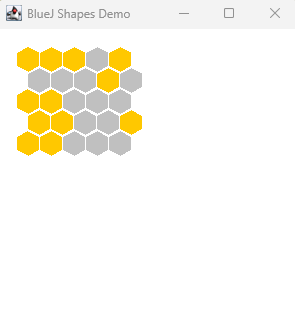


Se ejecuta replicate, cada celda en la red actualiza simultáneamente su estado examinando su propio estado y aquellos de sus vecinos aledaños.

***Mini-ciclo: 3***

* invertColumns(int[][] matrix)
* invertRows(int[][] matrix)
* newBoard(int[][] matrix)
* reflectHorizontal()
* reflectVertical()

**reflectHorizontal() reflectVertical()**



La figura resultante del replicate se refleja horizontal o verticalmente.

1. **Indiquen las extensiones necesarias para reutilizar el paquete shapes y la clase Hexagon. Explique.**

De la clase Hexagon, utiliza el constructor, reutilizamos el método makeVisible() del paquete shapes para hacer visible el hexágono, al igual que changeColor() para cambiar los colores de los hexágonos en la cuadrícula.

**RETROSPECTIVA**

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

15 h y 30 min Natalia Rojas / 15 h y 30 min Ana Duran

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

Completo, abarcamos punto y los desarrollamos en su totalidad.

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

Pair Programming, porque simultáneamente íbamos trabajando para hacer las respectivas retroalimentaciones del código

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

Haber logrado hacer la cuadrìcula en forma de hexágonos y de manera que encajaran.

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

Tuvimos un inconveniente con BlueJ, el cual nos retrasó.

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

Trabajamos a la par, resolvimos dudas entre nosotras y buscamos ayuda cuando lo llegamos a necesitar. Nos comprometemos a seguir trabajando en equipo.