# SSPROGRAMACIÓN ORIENTADA A OBJETOS

**Construcción. Clases y objetos. 2023-2**

# Laboratorio 1/6

**JULIANA BRICEÑO CASTRO – CRISTIAN JAVIER ALVAREZ**

## 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](#_bookmark0).
5. Utilizar el entorno de desarrollo de BlueJ.

## ENTREGA

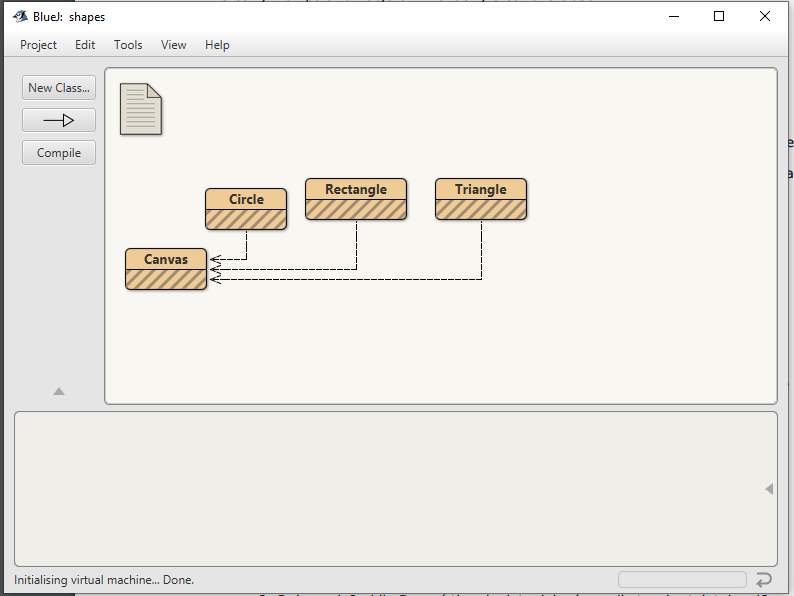
* Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
* Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios correspondientes.

## SHAPES

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.



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

1. Identificamos 4 clases (Triangule, circle, rectangle, canvas).
2. Triangule, circle, rectangle son herencias de Canvas.
   1. La documentación[2](#_bookmark1) 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? (b) ¿Qué atributos tiene la clase Triangle? (c) ¿Cuántos métodos ofrece la clase Triangle? (d) ¿Cuáles métodos ofrece la clase Triangle para que la figura cambie su tamaño (incluya sólo el nombre)?
3. Según la documentación generada, encontramos las clases de: canvas, circle, rectangle, triangle

https://lh4.googleusercontent.com/i5FH38HkAuLTDAlU8uxLZzkRYrSWxN92fJIeBhKEzYAfPGCJBXqcqDmVfi8LiupJsHk9l7BpWstzz36YQIwcYluC4InNWhakmHBiraC8ywLm3JAxZM1PgQfKwc8W_2OTTCjXN71XjPJAm5_qtmkHhyA https://lh6.googleusercontent.com/LpG_MEG2Q89LOLiUG0TNa3Rv_UJyg4HFi4VNnWstqIQ-hS4AfXgAVxP18VadG7UpIIORbkViu-Rb25wNmQbtG522WENh5D36FAUPkHP2GoCq5Tpi-VEoVz-BCGyFjGNxeIAPVKuwDrflfzuraqO7u5Q https://lh5.googleusercontent.com/SJJTNEucMF-JezuH2RlhOTKAvuDe0SN7ZOWVb54Qy92kgjoUlJEcxkO7bYo1Qy1YOWazVg--0MdFFmWEgg40QzJhqhsODiZrIAWRnY-IA0ougL5fMPMQla4_wIg5oxVEK6Kx0ukuioOH12OVlNWWLgE https://lh4.googleusercontent.com/RXLZt8Uz8yXVQVT1bgOME_ug5OiZ_MHALRIcGHYR9PQDnuIUppctuWiYGYYWSeWsfy2CQ-SCMjsuJkS0fGClGif9nNWoxfH86_BkOnteYnyOsaq4LDOQRF7s1GRV92dLKavGMperS-DnzQkE9Z5pAlU

1. Los atributos que tiene la clase triangle según la documentación son: height, width, xPosition, yPosition, color
2. Según la documentación tenemos 13 métodos.
3. 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? (b) ¿Cuáles atributos determinan su forma? Con respecto a los métodos: (c) ¿Cuántos métodos tiene en total? (d) ¿Quiénes usan los métodos privados?
4. VERTICES, height, width, xPosition, yPosition, color, isVisible
5. VERTICES, height, witdh.
6. 15 métodos
7. Los métodos de draw y erase son privados. Otros métodos de la misma clase utilizan los métodos privados, asi:

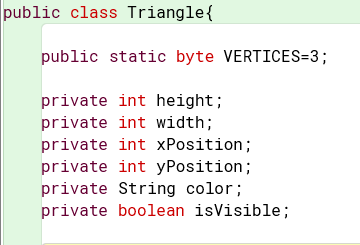
|  |  |
| --- | --- |
| Usa Draw() | Usa Erase() |
| MakeVisible | MakeInvisible |
| MoveVertical | MoveHorizontal |
| MoveHorizontal | MoveVertical |
| SlowMoveHorizontal | ChangeSize |
| SlowMoveVertical |  |
| ChangeSize |  |
| ChangeColor |  |

* 1. Comparando la documentación con el código (a) ¿Qué no se ve en la documentación? (b) ¿por qué debe ser así?

1. No se ve en la documentación: un atributo estático (vertices), no se ve el atributo de IsVisible, No se ven los métodos privados.
2. No se ven porque al ser privados no se pueden usar fuera de la clase, por eso no salen en la documentación.
   1. En el código de la clase Triangle, revise el atributo VERTICES (a) ¿Qué significa que sea public? (b) ¿Qué significa que sea static? (c) ¿Qué significaría que fuera final? ¿Debe serlo? (d) ¿De qué tipo de dato debería ser (byte, short, int, long)?

¿Por qué? (e) Actualícenlo.

1. Que sea public significa que es visible en cualquier clase
2. Que sea static significa que se crea independientemente a que se definan objetos de la misma, en caso de que se creen objetos de la misma clase el atributo es compartido por todas las instancias.
3. Significa que es una variable de tipo constante y no aceptará cambios después de su asignación. Debe serlo porque así se garantiza que la figura generada sea siempre un polígono de 3 lados.
4. Dado su tamaño, debería ser de tipo Byte, porque la variable int ocupa mucha más memoria al abarcar un rango tan grande, y no se necesita.

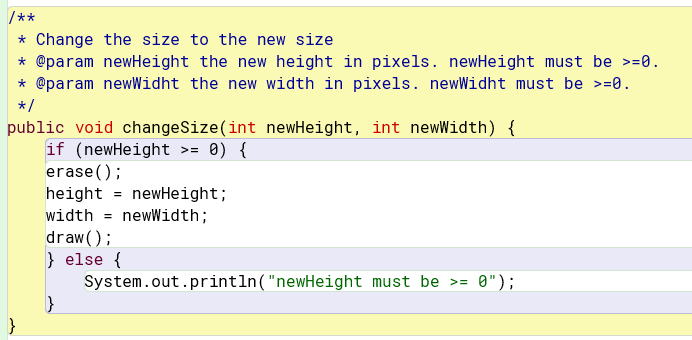


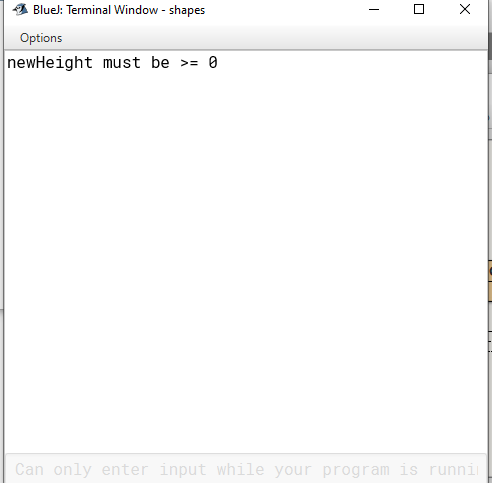
1. <http://docs.oracle.com/javase/8/docs/api/>
2. Menu: Tools-Project Documentation
   1. En el código de la clase Triangle revisen el detalle del tipo del atributo height (a)

¿Qué se está indicándo al decir que es int? (b) Si height fuera byte, ¿cuál sería el triángulo más grande posible? (c) y ¿si fuera long? d) ¿qué restricción adicional deberían tener este atributo? Refactoricen el código considerando (d).

1. Al decir que es de tipo int puede tomar valores desde hasta -1, tiene 32 bits de memoria
2. Si fuera tipo byte tendría 8 bits de memoria, el valor más grande posible para height es de 127
3. Si fuera tipo long tendría 64 bits, por lo cual el valor más grande que podría tomar height es de .
4. Aunque se especifica en la documentación, no se garantiza que la altura sea mayor que 0.

Refactorizando





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

El propósito del proyecto shapes es familiarizar al desarrollador con programación orientada a objetos, con java y con blueJ

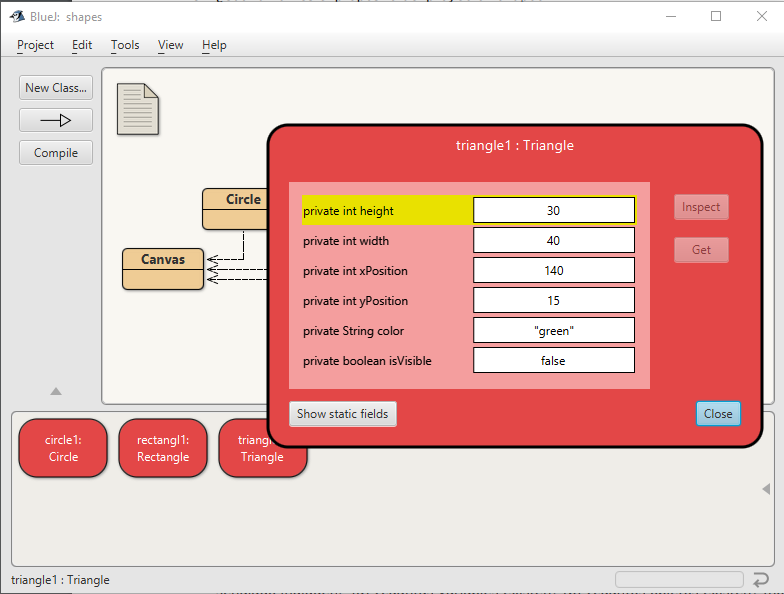
## Manipulando objetos. Usando un objeto.

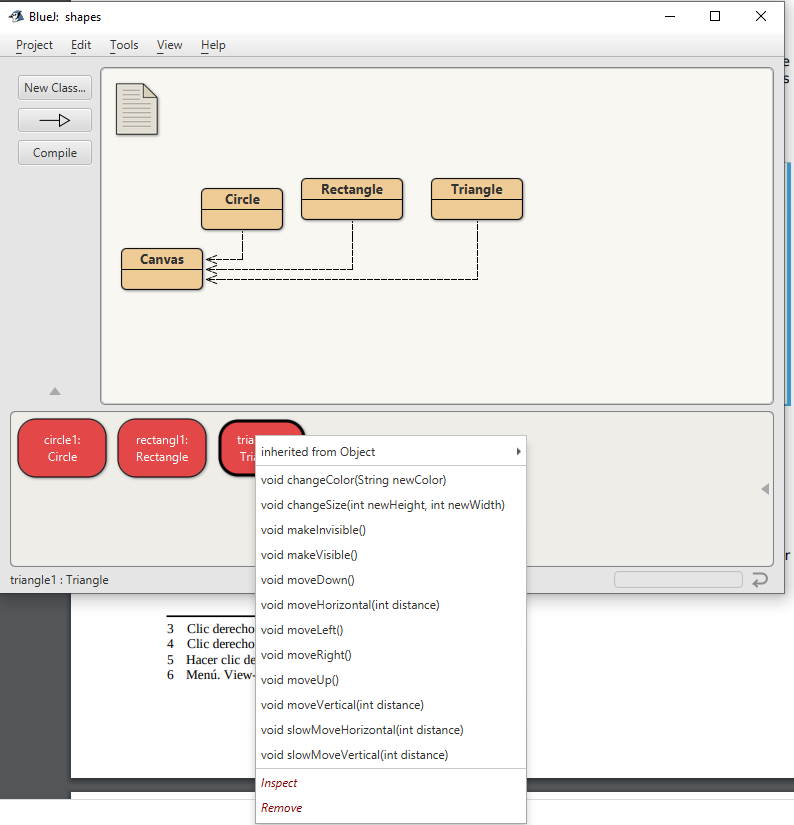
[En lab01.doc]

* 1. Creen un objeto de cada una de las clases que lo permitan[3](#_bookmark2). (a) ¿Cuántas clases hay?

(b) ¿Cuántos objetos crearon? (b) ¿Quién se crea de forma diferente? ¿Por qué?

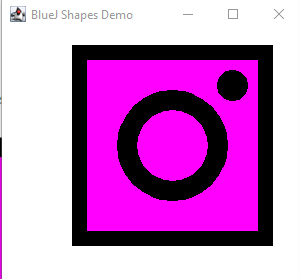
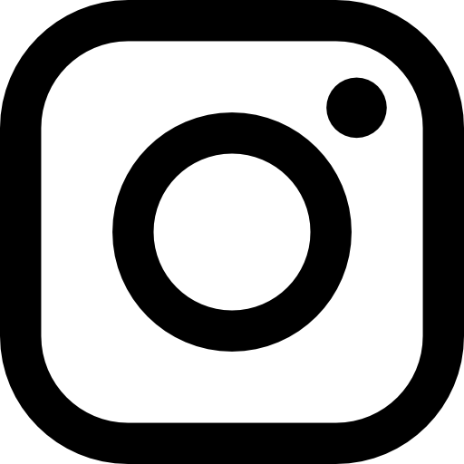
* + - * 1. Hay 4 clases (canvas, triangle, rectangle, circle)
        2. 3 objetos se crean triangle, rectangle, circle.
        3. La clase canvas se crea de manera diferente, generando una ventana y no se visualiza junto a los demás objetos creados. Esto porque en el canvas es donde se mostrarán los objetos cuando estos sean visibles.
  1. Inspeccionen el estado del objeto: Triangle[4](#_bookmark3), (a) ¿Cuáles son los valores de inicio de todos sus atributos? (b) Capturen la pantalla.

1. 
   1. Inspeccionen el comportamiento que ofrece el objeto: Triangle[5](#_bookmark4). (a) Capturen la pantalla. (b) ¿Por qué no aparecen todos los que están en el código?



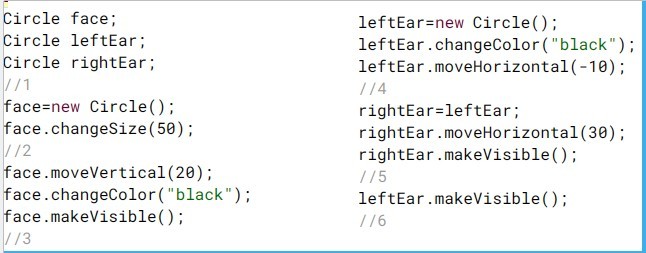
No aparecen todos, porque son privados

* 1. 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? (b) ¿Cuántos objetos se usan en total? (c) Capturen la pantalla. (d) Incluyan el logo original.

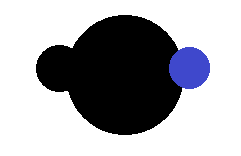
1. Necesitamos 3 clases (canvas, circle, rectangle)
2. 4 objetos y el canvas
3. 
4. 

## Manipulando objetos. Analizando y escribiendo código.

[En lab01.doc]



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

1. Pensamos que era un rostro humano
2. 
   1. Habiliten la ventana de código en línea[6](#_bookmark5), escriban el código. Para cada punto señalado indiquen: (a) ¿cuántas variables existen? (b) ¿cuántos objetos existen? (no cuenten ni los objetos String ni el objeto Canvas) (c) ¿qué color tiene cada uno de ellos? (d) ¿cuántos objetos se ven?

Al final, (e) Expliquen sus respuestas. (f) Capturen la pantalla.

1. Uno
2. 3 (face, leftEar, rightEar)
3. No hay
4. No tienen
5. 0
6. Dos
7. 3 variables
8. 1 objeto
9. face es azul (por defecto)
10. Ninguno
11. Tres
12. 3 variables
13. 1 objeto
14. Fase es Negro
15. Fase es Visible

**4**. Cuatro

1. 3 variables
2. 2 objetos
3. face es Negro, leftEar es negro
4. Es visible face

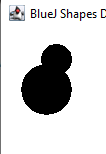
**5**. Cinco

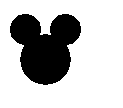
1. 3 variables
2. 2 objetos
3. face es negro, leftEar es negro, rightEar es negro
4. face y rightEar es visible

**6**. Seis

1. 3 variables
2. 2 objetos
3. face es negro, leftEar es negro, rightEar es negro
4. face, rightEar, leftEar es visible.

**e)** La razón por la cual solo hay dos objetos es porque se aloja en la misma posición de memoria, volviendo las variables de leftEar y rightEar queden apuntando al mismo objeto.

**f)**



* 1. Compare figura pintada en 1. con la figura capturada en 2., (a) ¿son iguales? (b) ¿por qué?

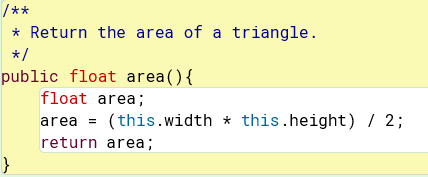
R// No son iguales, porque un objeto es de diferente color y la posición de las orejas están diferentes. Además, no consideramos que al igual left y right Ear lo que sucedería es que se alojará en la misma posición de memoria, no solo copio los atributos, sino que se volvió el mismo objeto

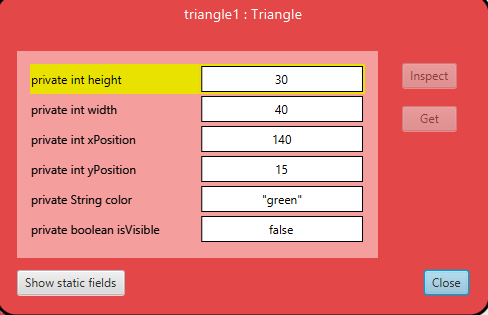
1. Clic derecho sobre la clase
2. Clic derecho sobre el objeto
3. Hacer clic derecho sobre el objeto.
4. Menú. View-Show Code Pad.

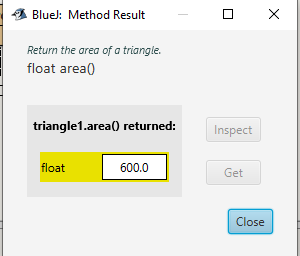
## Extendiendo una clase. Triangle.

[En lab01.doc y \*.java]

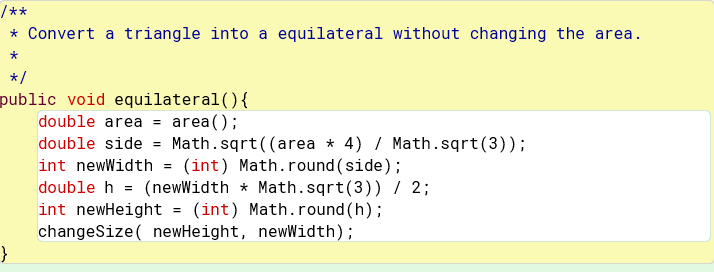
* 1. Desarrollen en Triangle el método área(). ¡Pruébenlo! Capturen una pantalla.



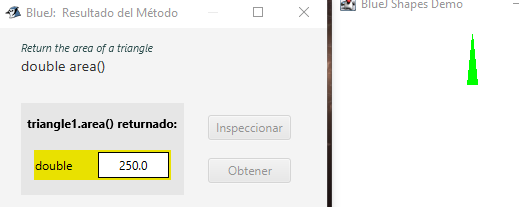




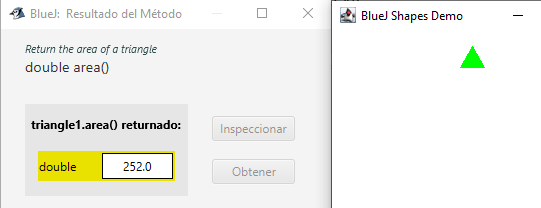
* 1. Desarrollen en Triangle el método equilateral () (convierte el triángulo en un equilátero manteniendo el área). [7](#_bookmark6) ¡Pruébenlo! Capturen dos pantallas.



Captura 1: triangulo no equilátero (área actual = 250)



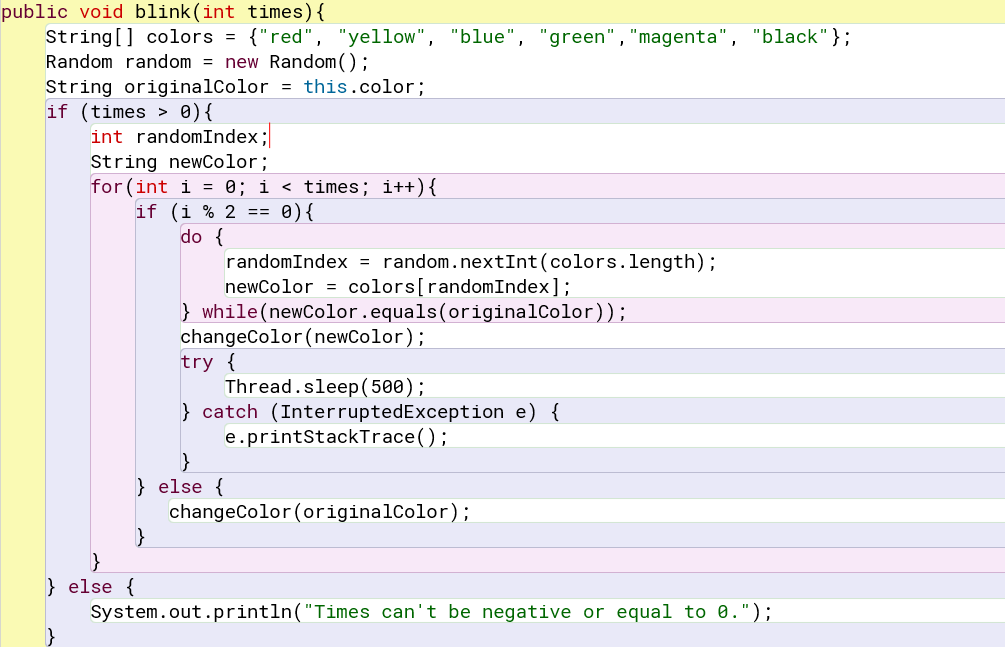
Captura 2: triángulo equilátero



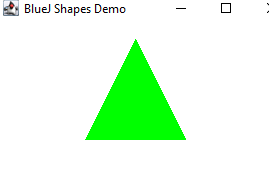
Explicación: Debido a que nos obligan a redondear de double a int los valores de height y width, hay una pequeña variación en la construcción del triángulo.

* 1. Desarrollen en Triangle el método blink(times:int) (si times>0, cambia de color el

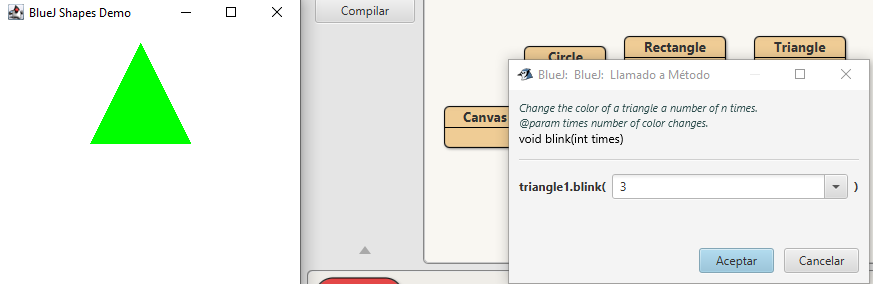
número de veces indicado (Por ejemplo, original – nuevo – original serían dos veces. El color nuevo se escoge al azar entre los disponibles [8](#_bookmark7)). ¡Pruébenlo! Capturen dos pantallas.

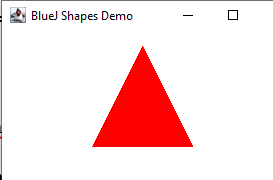


Captura 1:

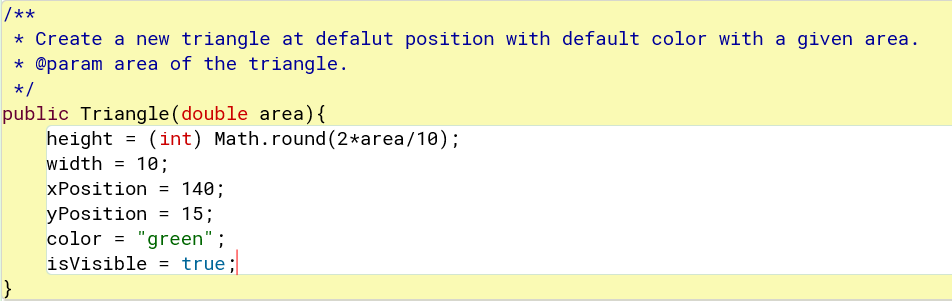


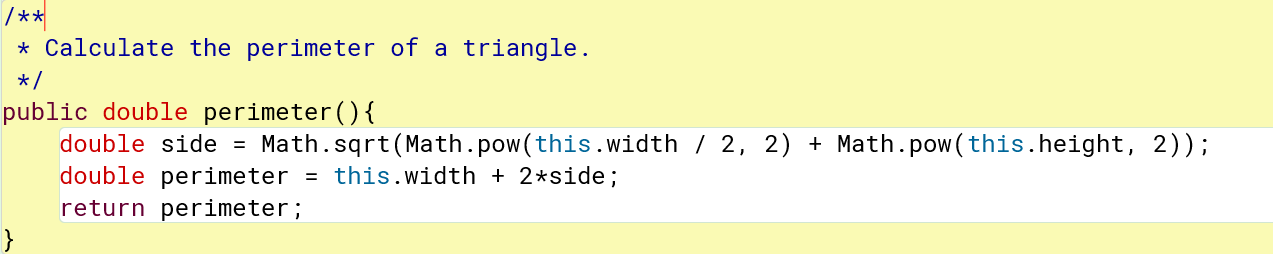
Captura 2: Con times = 3

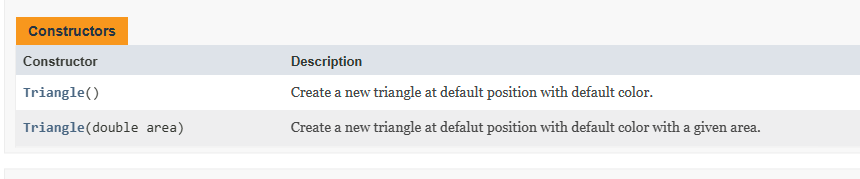


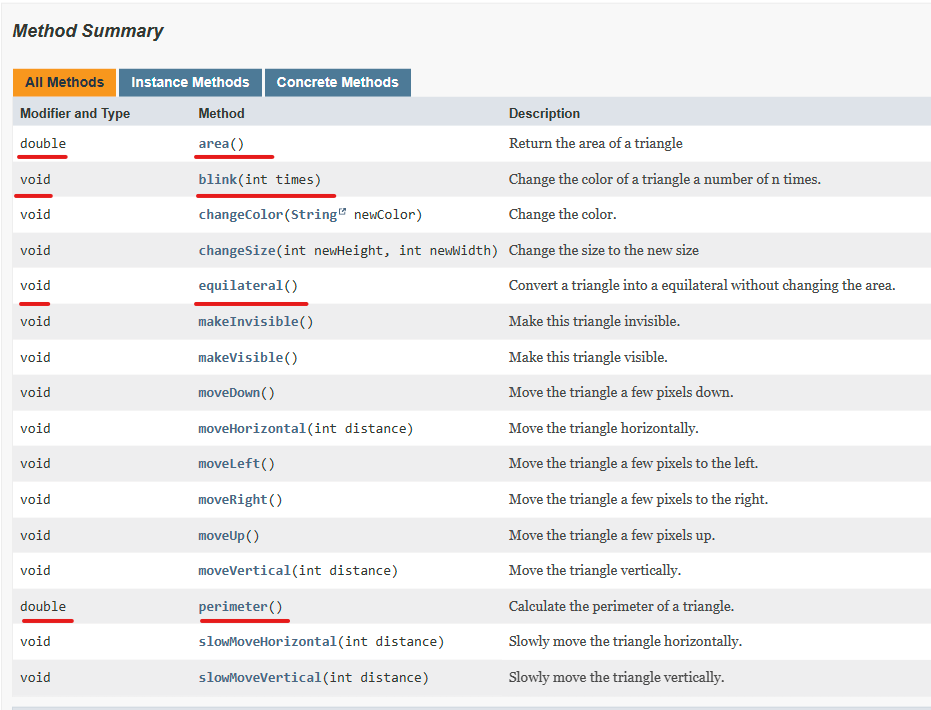


* 1. Desarrollen en Triangle un nuevo creador que dada el área crea un triángulo al azar con dicha área.



* 1. Propongan un nuevo método para esta clase. Desarrolle y prueban el método.
  2. Generen nuevamente la documentación y revise la información de estos nuevos métodos. Capturen la pantalla.



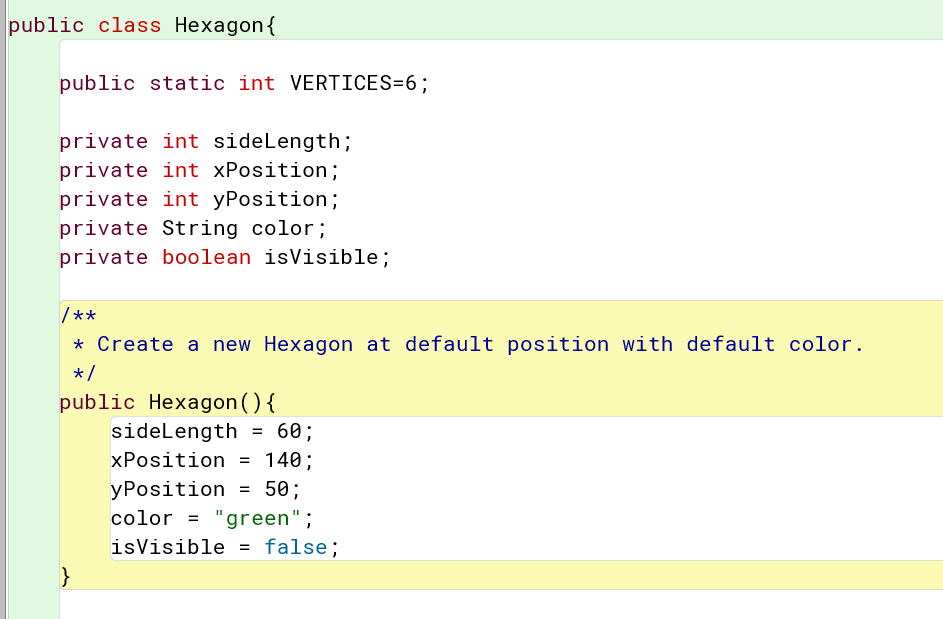


## Extendiendo un paquete. shapes + Hexágono

[En lab01.doc y \*.java]

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

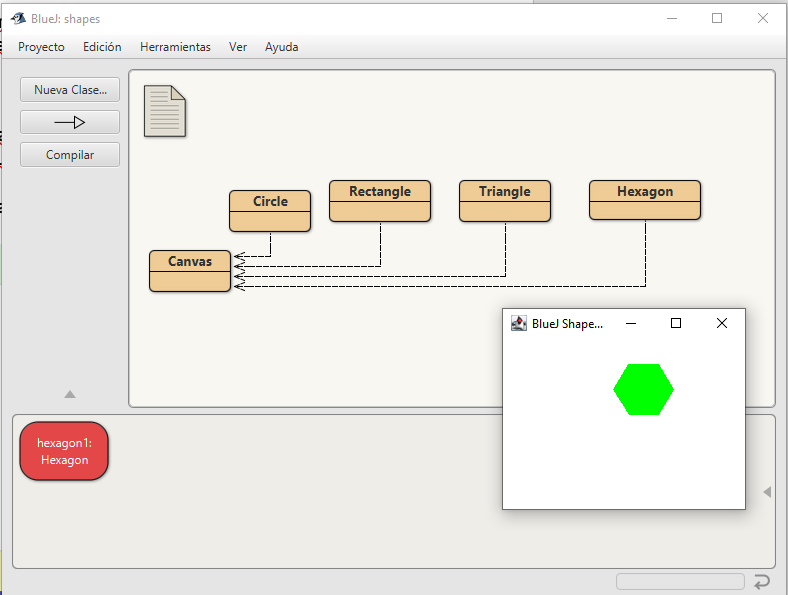
* 1. Para la construcción, tomen como base la clase Triangle inicial. (a) ¿Qué métodos podrían quedar iguales? (b) ¿Qué atributos podrían quedar iguales? (c) Justifiquen su respuesta.
  2. makeVisible, makeInvisible, moveRight, moveLeft, moveUp, moveDown, moveHorizontal, moveVertical, slowMoveHorizontal, slowMoveVertical, changeColor, erase.
  3. xPosition, yPosition, color, isVisible
  4. Se tuvo que modificar principalmente draw, además no necesitamos el atributo heigth, y en vez de width se utiliza sideLength, también se modificó changeSize y ahora solo recibe un parámetro.
  5. Inicie la construcción de la clase Hexagon únicamente con los atributos. (a) Adicione pantallazo con los atributos.



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

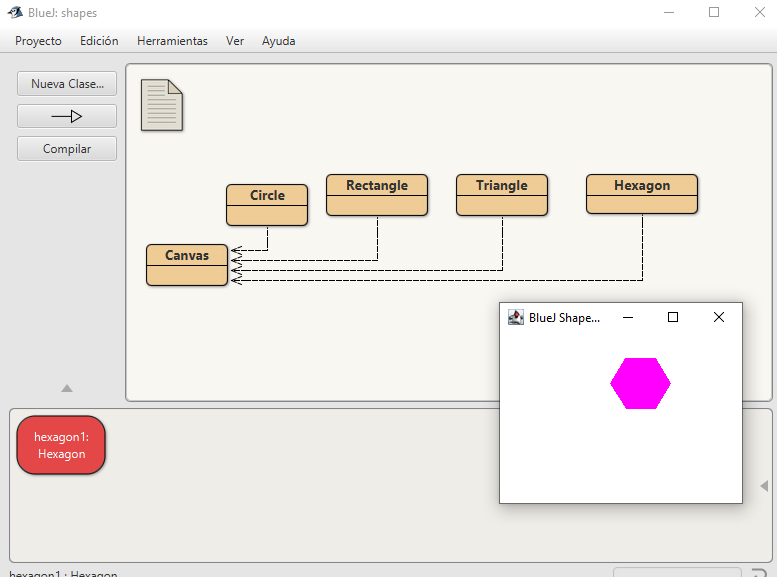
Mini-ciclo 1:

Constructor, makeVisible()

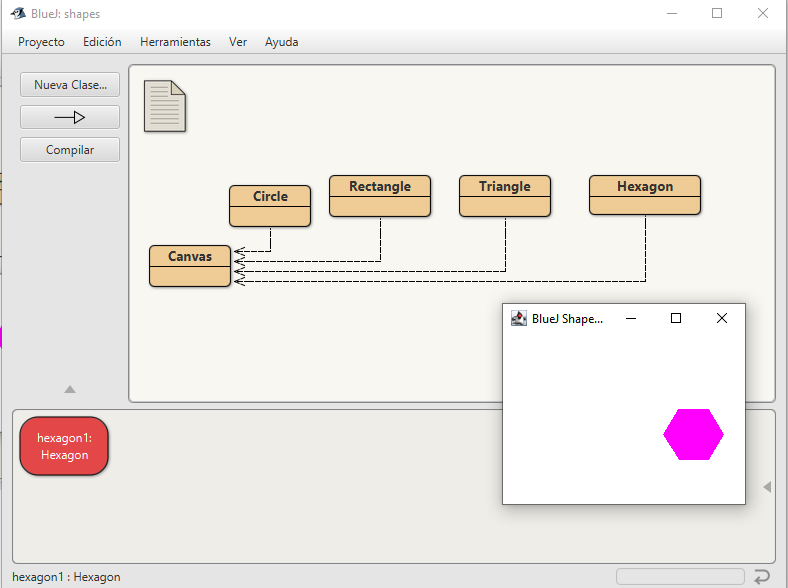


Mini-ciclo 2:

changeColor

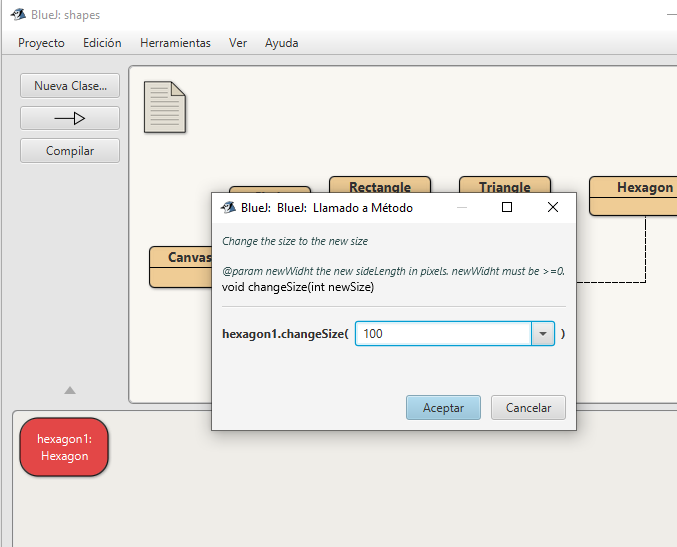


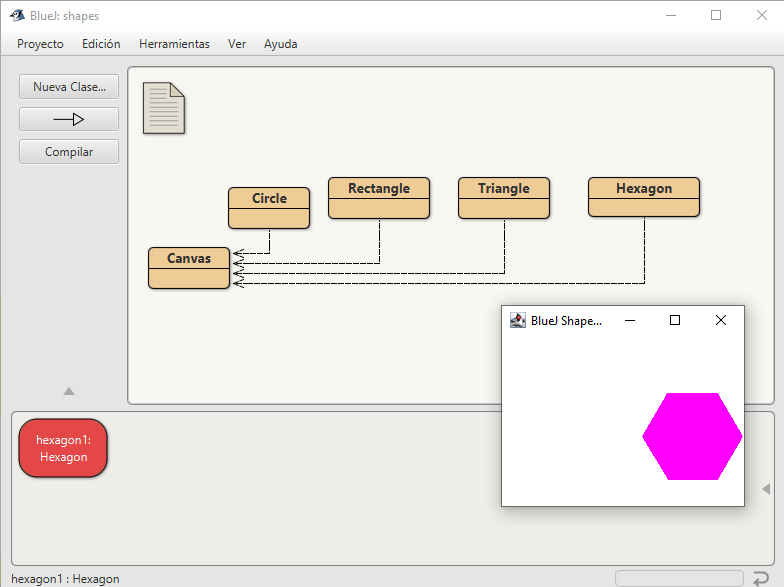
moveHorizontal, moveVertical



Mini-ciclo 3:

changeSize





1. Consulte la clase [Math](https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html) del API java
2. Consulte la clase [Math](https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html) del API java

## REPLICATE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Inicialmente, un conjunto de celdas se llena con una copia de la pieza que se va a replicar. En una secuencia de pasos discretos, cada celda en la red actualiza simultáneamente su estado examinando su propio estado y aquellos de sus vecinos aledaños.  Si cuenta un número impar de estas celdas vecinas llenas, el siguiente estado de la celda será llena; de lo contrario, será vacío.  La Figura G.1 muestra varios pasos en el proceso de replicación.  [De World Final Problem G Replicate Rfplicbte]  **NO DEBEN RESOLVER EL PROBLEMA DE LA MARATÓN** | | | | |  |
| Nuestro replicate sigue las celdas van a ser hexágonos. | mismas | reglas | pero | las |  |

1. 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, debe generar un mensaje de error. Use JoptionPane.
  1. Diseñen la clase, es decir, definan los métodos que debe ofrecer.

Replicate:

//Atributos

- ArrayList<Hexagon> hexagonalMatrix

- int[][] stateMatix

//Metodos

- Constructores:

a. Constructor random: Replicate(int rows, int columns)

b. Constructor dada la celda incial pintada: Replicate(int rows, int columns, int xPositionPainted, int yPositionPainted)

- replicate(): Hace el raplicate según las regal cada ver que se llama la funcion

- reflectVertical(): refleja verticalmente el replicate

- reflectHorizontal(): refleja horizontalmente el replicate

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

Replicate(6,6);

//Mini ciclo 1

replicate();

reflectVertical();

//Mini ciclo 2

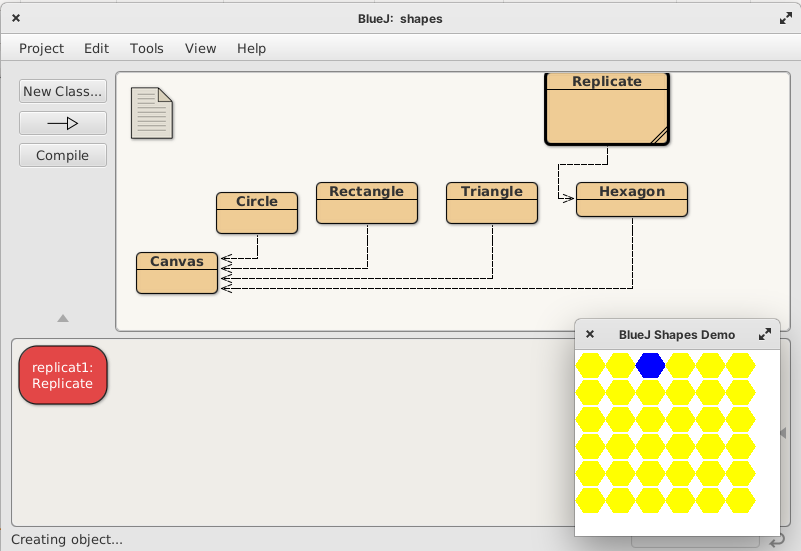
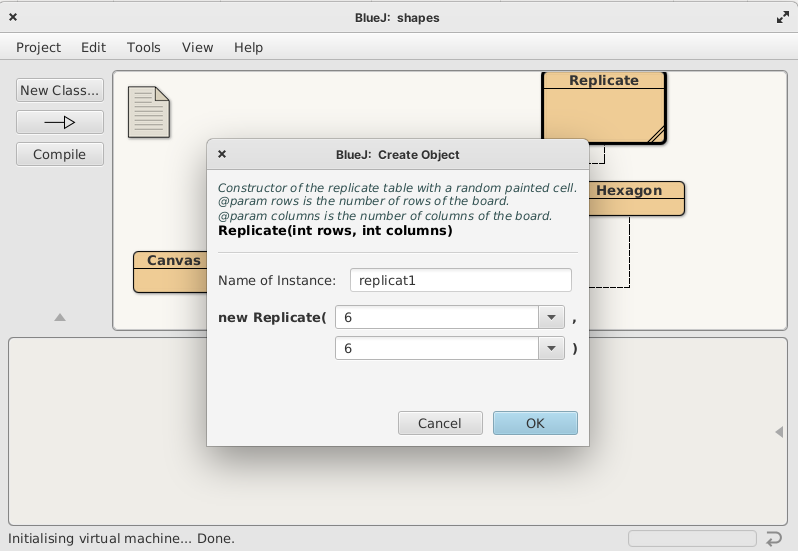
replicate();

reflectHorizontal();

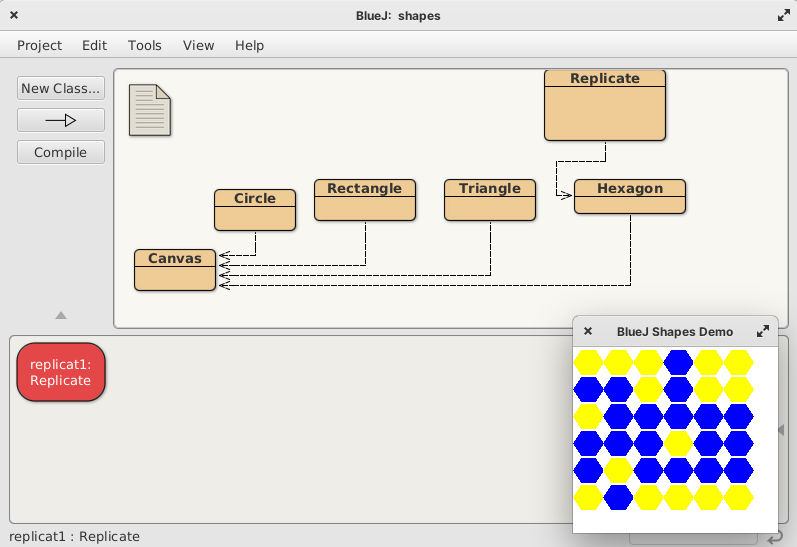
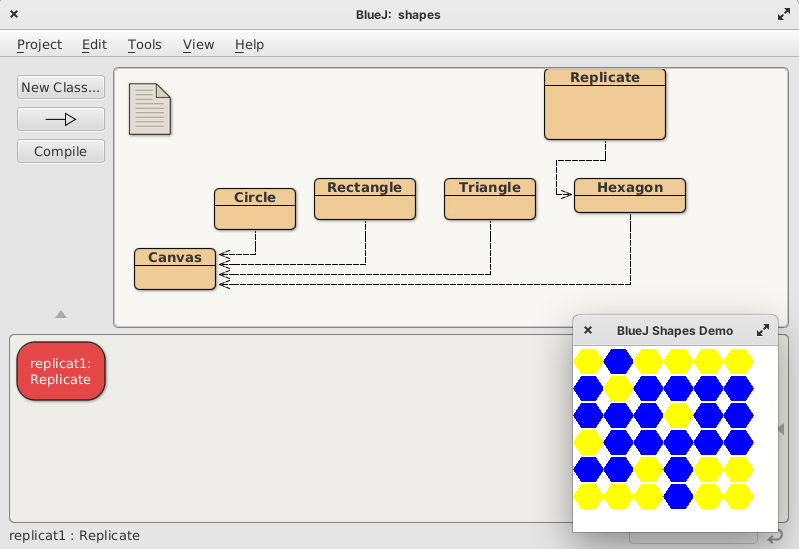
//Mini cliclo 3

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

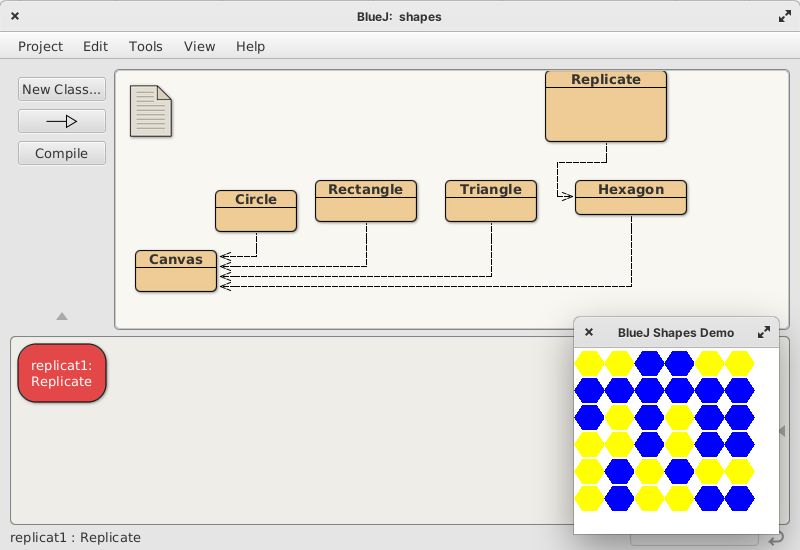
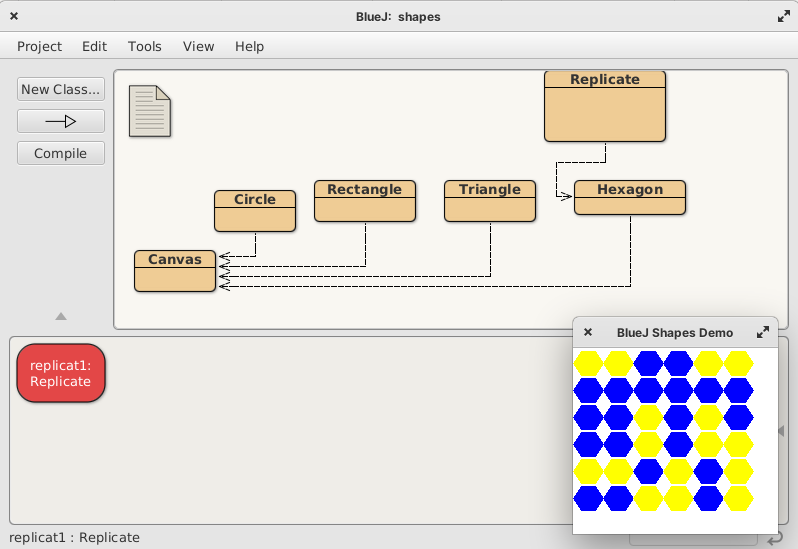
//Mini cliclo 1: crear la matriz de hexagonos con una celda incial pintada de forma random (para comprobar que funciona el constructor)



//Mini ciclo 2: hacer el primer replicate y reflejar verticalmente (se comprueba que funciona el metodo replicate y reflectVertical)



//Mini ciclo 3: hacer el segundo replicate y reflejar horizantalmente (se comprueba que funciona el metodo replicate por segunda vez y reflectHorizontal)



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

Hexagon. Explique.

Para reutilizar la clse Hexagon implementamos dos cosas: creamos nuevos atributos (row y column) que me mostraran su posición en la matriz, además de implementar getters para estos atributos, también implementamos un nuevo constructor que se adaptara a nuestras necesidades, es decir, un constructor que instanciara un hexágono de acuerdo a una posición en X y en Y, que guardara en los nuevos atributos su fila y columna, y que pintara el hexágono con un tamaño y un color elegidos previamente. La clase shapes ya estaba siendo utilizada cada vez que instanciamos un hexágono.

## BONO. Nuevos requisitos funcionales. Replicate.

[En lab01.doc. SelfAssembly.java]

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

Nuevos requisitos funcionales

* Rotar 45% a derecha o izquierda.
* Llenar una celda para que en la próximo replicate se llene un número dado de celdas, si es posible. Explique la estrategia.

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) 12 horas/ Juliana, 12 horas/ Cristian
2. ¿Cuál es el estado actual del laboratorio? ¿Por qué? Esta completo porque lo hicimos con tiempo y detenidamente
3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué? Consideramos que la programación a pares fue muy útil porque nos permitió compartir nuestras ideas y revisar el código mientras estaba siendo construido.
4. ¿Cuál consideran fue el mayor logro? ¿Por qué?

Hacer replicate, ya que fue el ejercicio más complejo del laboratorio.

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

Entender como funcionaban algunas cosas propias de Java, lo resolvimos investigando y leyendo documentación.

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

Hicimos bien el hecho de que no procrastinamos y que tuvimos mucha y buena comunicación durante el desarrollo del laboratorio.