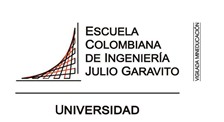
****

**Escuela Colombiana de Ingeniería Julio Garavito**

**Programación Orientada a Objetos 2024-2**

**Laboratorio I**

**Cristian Santiago Pedraza Rodríguez**

**Andersson David Sánchez Méndez**

**23 de agosto de 2024**

# PROGRAMACIÓN ORIENTADA A OBJETOS

**Construcción. Clases y objetos. 2024-2**

# 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](#_gjdgxs).
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

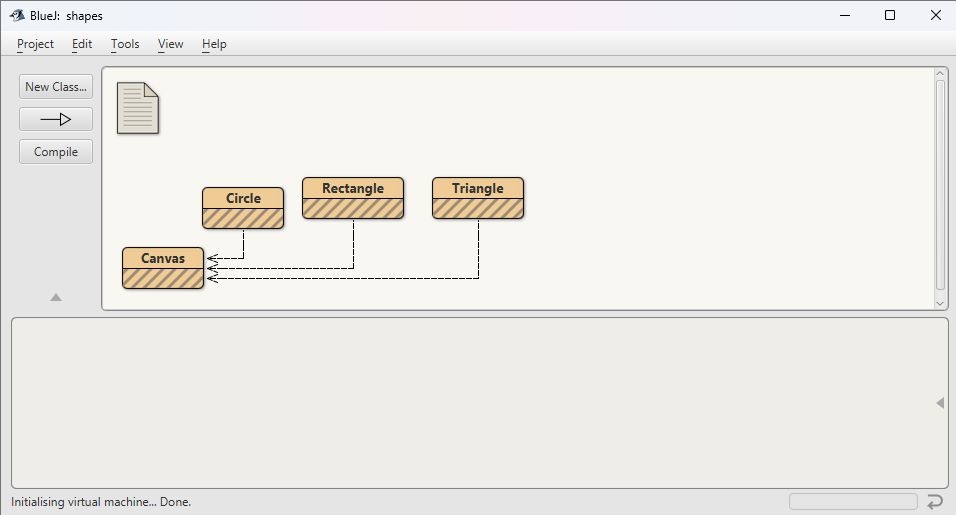
* Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
* Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios correspondientes.

## SHAPES

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.



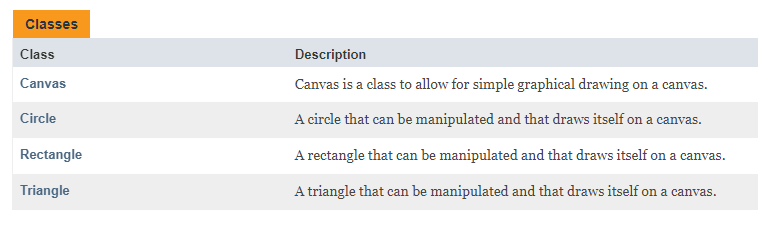
* 1. El **diagrama de clases** permite visualizar las clases de un artefacto software y las relaciones entre ellas. Considerando el diagrama de clases de “shapes” (a) ¿Qué clases ofrece? (b) ¿Qué relaciones existen entre ellas?

1. En el diagrama de clases podemos evidenciar cuatro clases:
   1. Canvas
   2. Circle
   3. Rectangle
   4. Triangle
2. Las relaciones existentes entre los cuatros se dan de la siguiente forma.

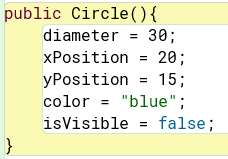
Las tres clases Circle, Rectangle y Triangle están asociadas por medio de flechas de dependencia a la clase canvas, lo cual indica que, si modificamos la clase Canvas, este cambio se verá reflejado en sus clases asociadas.

* 1. La **documentación**[2](#_gjdgxs) 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 Circle? (c) ¿Cuántos métodos ofrece la clase Circle? (d) ¿Cuáles métodos ofrece la clase Circle para que la figura cambie su tamaño (incluya sólo el nombre)?

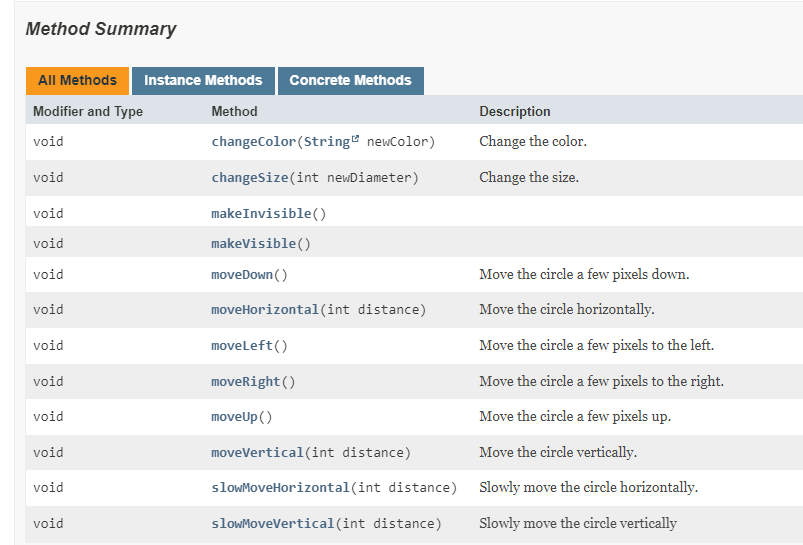
a) ¿Qué clases tiene el paquete shapes?

1. 

b) ¿Qué atributos tiene la clase Circle?



c) ¿Cuántos métodos ofrece la clase Circle?

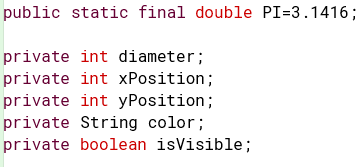


d) ¿Cuáles métodos ofrece la clase Circle para que la figura cambie su tamaño (incluya sólo el nombre)?



* 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:

(a) ¿Cuántos atributos realmente tiene?



(b) ¿Cuáles atributos determinan su forma?

Son:

* diameter
* xPosition
* yPosition
* PI

Con respecto a los métodos:

(c)¿Cuántos métodos tiene en total?

- public Circle()

Métodos Públicos

- public void makeVisible()

- public void makeInvisible()

- public void moveRight()

- public void moveLeft()

- public void moveUp()

- public void moveDown()

- public void moveHorizontal(int distance)

- public void moveVertical(int distance)

- public void slowMoveHorizontal(int distance)

- public void slowMoveVertical(int distance)

- public void changeSize(int newDiameter)

- public void changeColor(String newColor)

Métodos Privados

- private void draw()

- private void erase()

En total hay 15 métodos.

(d) ¿Quiénes usan los métodos privados?

Solo la clase Circle utiliza los métodos privados, dado que solo ella puede acceder a ellos.

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

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

Los métodos y atributos privados.

(b) ¿por qué debe ser así?

Porque al momento de un tercero querer utilizar la clase Circle, no tiene la necesidad de conocer los métodos y atributos privados.

* 1. **En el código de la clase Circle, revise el atributo PI**

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

Que el atributo es accesible para la clase en que se define y para las demás clases.

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

Que el atributo es declarado y su valor asignado una vez creada la clase, para cada instancia del tipo en la clase. Además, en cualquier instancia de la clase puede ser reescrito.

**(c) ¿Qué significa que fuera final?**

Para declarar constantes, no se puede asignar un nuevo valor a la variable una vez inicializada

**¿Debe serlo?**

Sí debería serlo, el valor de PI no debería estar cambiando. Debe ser inmutable.

**(d) ¿De qué tipo de dato debería ser (float, double)? ¿Por qué?**

Debería ser double porque esta tiene mayor cantidad de memoria que float, así que ayudará a la precisión del valor PI, pero también puede ser float, para almacenar gran cantidad de datos, ya que ocupa 4 bytes en memoria.

**(e) Actualícenlo.**

****



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

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

El tipo int al ser un dato primitivo en Java, que representa un número entero de 32 bits, significa que el diámetro tiene que ser cualquier número, pero que esté dentro de ese intervalo; así que el diámetro se asegura que sea entero, con una cantidad fija de memoria de 4 bytes.

(b) Si width fuera byte, ¿cuál sería el rectángulo más grande posible?

El valor máximo al que el rectángulo adoptaría sería de 127 porque el rango de un byte es de -128 a 127, entonces el diámetro sería de 127 píxeles, así que el círculo tendría un área máxima (considerando el mayor diámetro posible) es π\* (127/2)2

(c) y ¿si fuera long?

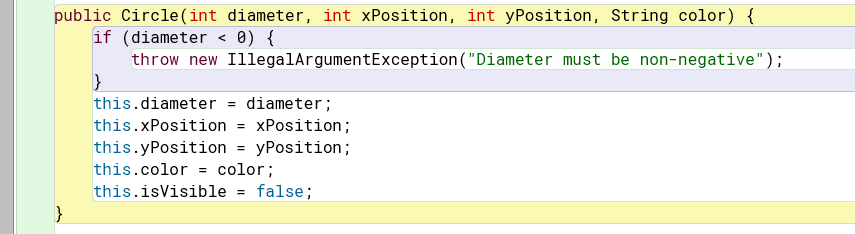
Si fuera long adoptaría un diámetro extremadamente largo, es decir, el círculo que se crea es demasiado grande, lo cual sería imposible representar en un lienzo(canvas) debido a las limitaciones de hardware y memoria.

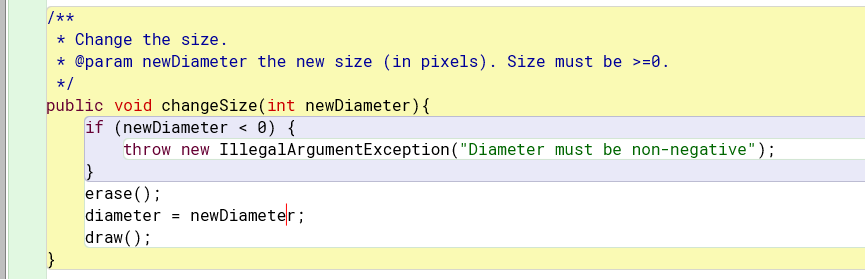
d) ¿qué restricción adicional deberían tener este atributo?

El diámetro debería tener la condición de que siempre fuera positivo, y posiblemente que no sea menor a 1 pixel. Esto se podría hacer agregando en el método changeSize() o en el constructor de que el diámetro no sea negativo, y si es así, marcar un mensaje de error.

(e) Refactoricen el código considerando

Se modifica en el método changeSize() y se agrega una clase donde van a estar los setters de cada atributo.





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

El propósito de shapes a nivel didáctico es ayudarnos a verificar y entender algunas características de los objetos a través de la creación de diferentes formas (Circle, Triangle, Rectangle )que se pintarán en la pantalla. Todo esto con el fin de ilustrar la programación orientada a objeto.

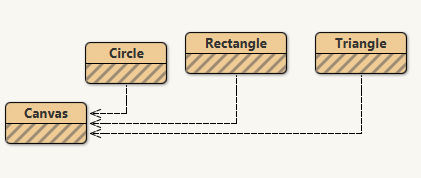
## Manipulando objetos. Usando un objeto.

**[En lab01.doc]**

* 1. Creen un objeto de cada una de las clases que lo permitan[3](#_30j0zll).

(a) ¿Cuántas clases hay?

Hay cuatro clases.



(b) ¿Cuántos objetos crearon?

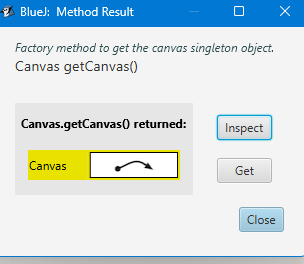
Se crearon cuatro objetos  


(c) ¿Quién se crea de forma diferente?

Canvas se crea de forma diferente.

¿Por qué?

Porque a la hora de seleccionar crear una nueva instancia de la clase Canvas. Aparece este recuadro de sistema. En donde no se crea una nueva instancia name = newCanvas, en su lugar aparece Canvas Get Canvas().

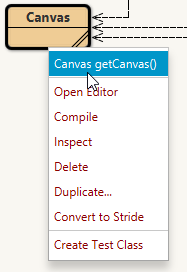


* 1. Inspeccionen los creadores de cada una de las clases.

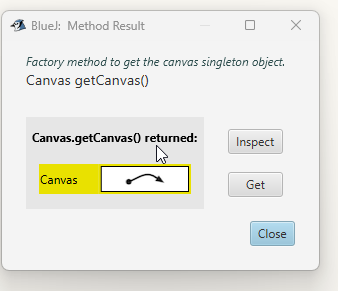
(a) ¿Cuál es la principal diferencia entre ellos?

La principal diferencia entre el creador de la clase canvas y las demás clases (Circle, Triangle, Rectangle) radica en lo siguiente:

**Canvas:**

****

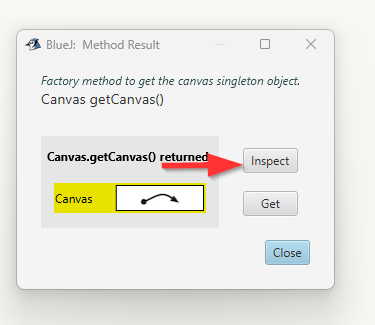
A la hora de querer crear una nueva instancia de la clase Canvas, sale la siguiente ventana de sistema:



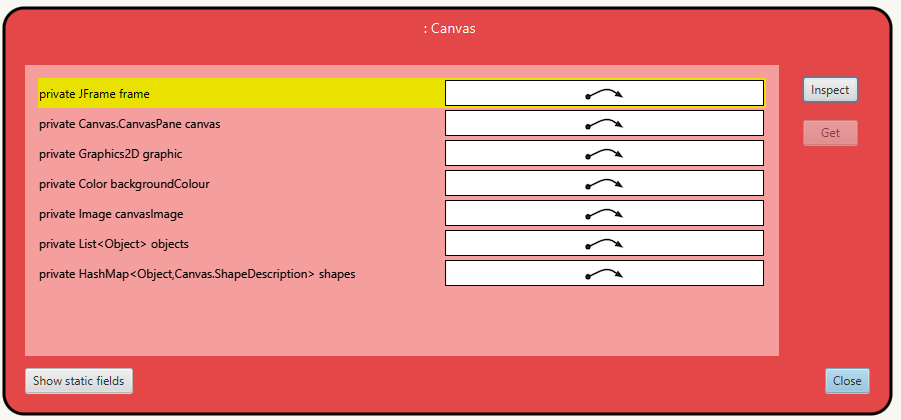
Y al mismo tiempo sale esta ventana de compilación con el lienzo



Sí damos clic en el botón inspect



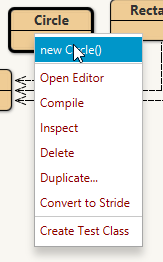
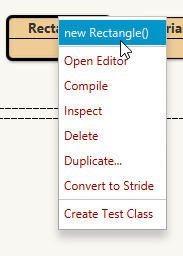
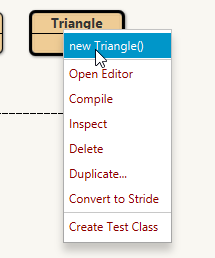
Nos enviará al siguiente apartado donde podremos interactuar con diferentes opciones de la clase canvas.



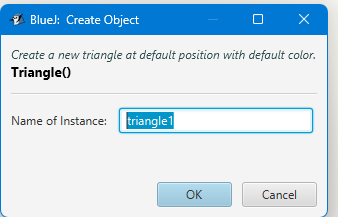
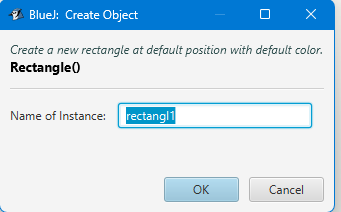
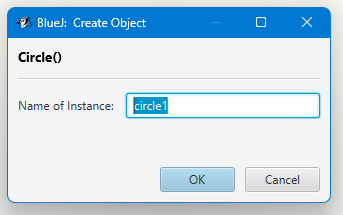
Donde cada elemento a la hora de seleccionar Inspect tiene su propio módulo de opciones.



Ahora, a la hora de querer instanciar un objeto de las demás clases (Circle, Rectangle, Triangle)



Tan solo aparecerá el recuadro que nos indica que asignemos el nombre del nuevo objeto.



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

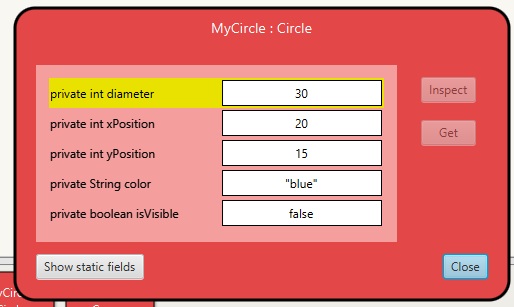
Con la clase que tiene el creador diferente, canvas, se busca crear el lienzo donde se pintaran las formas. Ahora, la información que nos proporciona el recuadro de creación nos indica que Canvas es un método de fábrica para obtener el objeto singleton de lienzo.

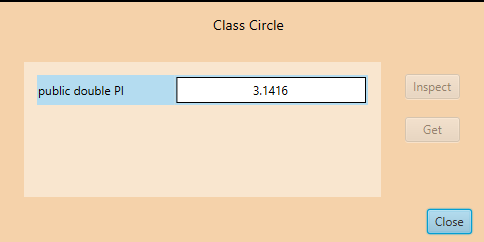
* 1. Inspeccionen el **estado** del objeto :Circle[4](#_30j0zll),

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

diameter = 30, xPosition = 20, yPosition = 15, color = “blue”, isVisible = false, PI = 3.1416

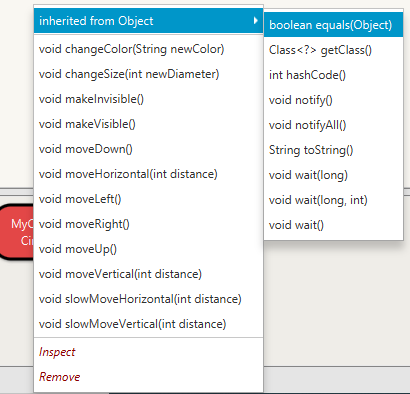
(b) Capturen la pantalla.





* 1. Inspeccionen el **comportamiento** que ofrece el objeto :Circle[5](#_1fob9te).

(a) Capturen la pantalla.



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

Porque al asimilar comportamiento con métodos, hay otra clase (Canvas) que es la que encierra los métodos draw() y erase(). Por esto en la clase Circle al momento de ver los comportamientos no aparecen porque en el código son privados, es decir, el método solo puede ser accedido y utilizado dentro de la misma clase que fue definido.

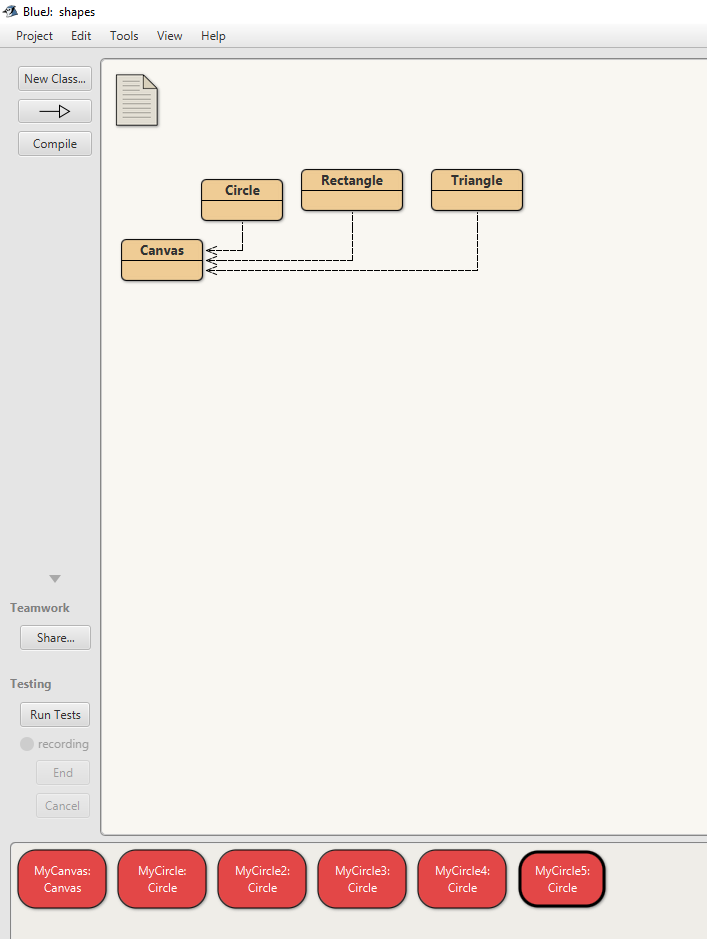
* 1. Construyan, con “shapes” sin escribir código, una propuesta de la imagen del logo de su empresa de software favorita.

(a) ¿Cuántas y cuáles clases se necesitan?

Para dar una idea de la propuesta de imagen del logo de Apple necesitaríamos solo usar la clase Canvas y la clase Circle, por lo tanto, serían dos clases.

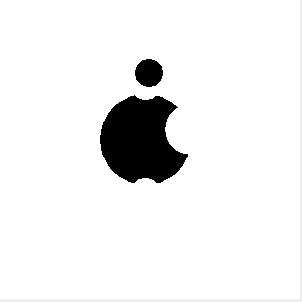
(b)¿Cuántos objetos se usan en total?

Necesitaríamos 1 lienzo y 5 círculos( 3 blancos y 2 negros).

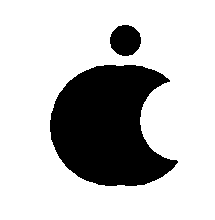


(c) Capturen la pantalla.

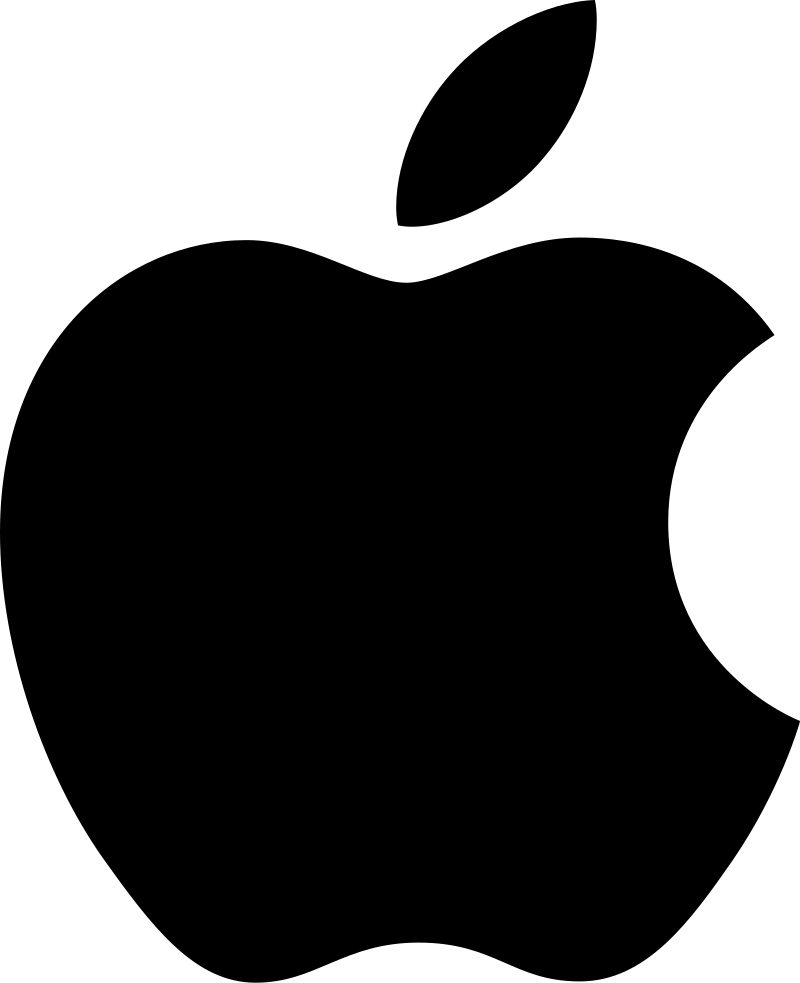
**CAPTURA ANDERSSON**

****

**CAPTURA CRISTIAN**

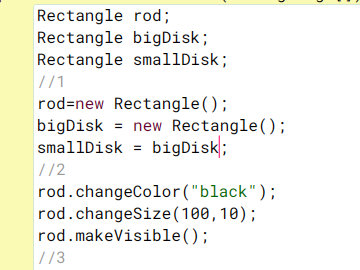
****

(d) Incluyan el logo original.



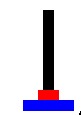
## Manipulando objetos. Analizando y escribiendo código.

**[En lab01.doc]**



* 1. Lean el código anterior.

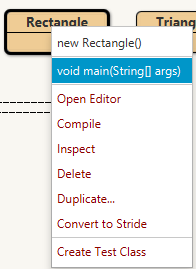
(a) ¿cuál creen que es la figura resultante?

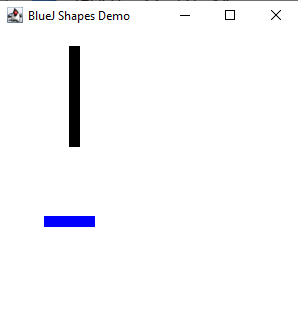
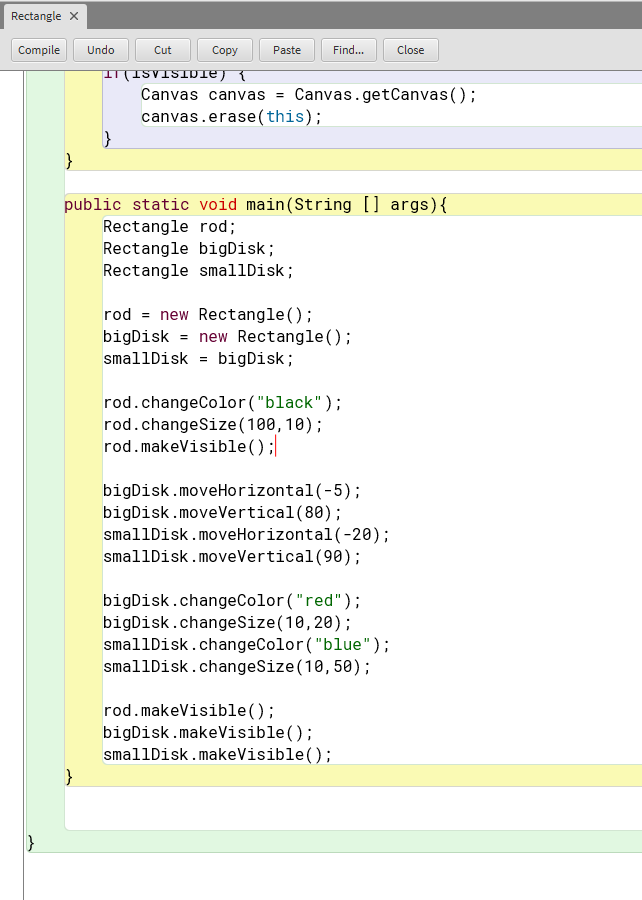




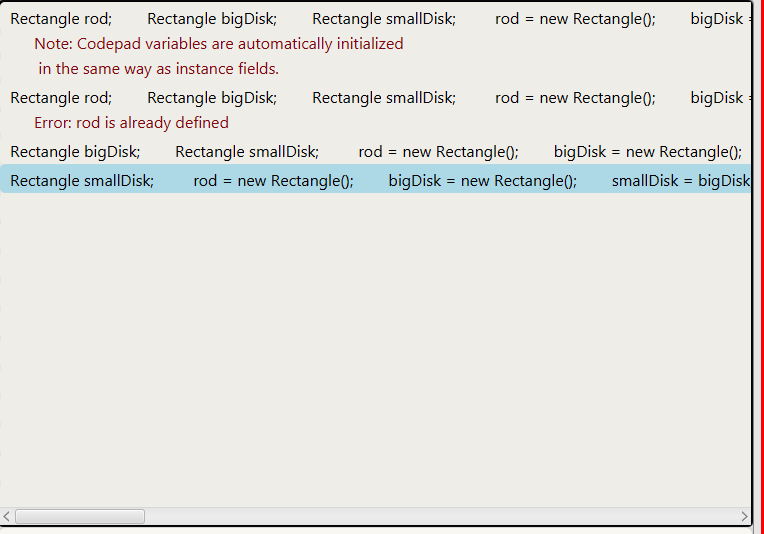
(b) Píntenla.

Compilando el código, porque al hacer manual cada paso no encontramos la opción en la que se pueda hacer la asignación de que smallDisk sea igual a bigDisk, así que haciendo el método main() en la clase Rectangle se compiló el código para generar esta figura:





* 1. Habiliten la ventana de código en línea[6](#_1fob9te), escriban el código. Para cada punto señalado indiquen:



(a) ¿cuántas variables existen?

Existen 3 variables declaradas de tipo Rectangle y son rod, bigDisk, smallDisk.

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

Se crean dos objetos de la clase Rectangle:

* Rod
* bigDisk y smallDisk apuntan al mismo objeto en memoria, ya que, smallDisk = bigDisk.

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

El color de rod es “black” cambiado con el método changeColor(“black”).

Y el color de bigDisk, smallDisk es azul porque son el mismo objeto. Inicialmente el color empieza en magenta (por defecto), luego cambia a “red” , y finalmente queda en “blue”.

(d) ¿cuántos objetos se ven?

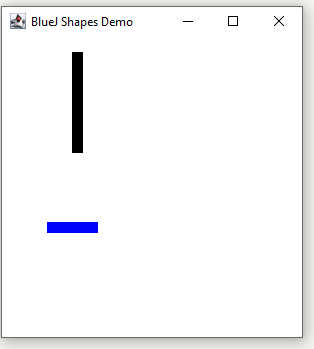
Al final del código se hacen visibles los 3 rectángulos, pero como bigDisk y smallDisk son el mismo objeto, entonces solo se ven 2 objetos diferentes.

Al final…

(e) Expliquen sus respuestas.

Todas las respuestas dadas anteriormente fueron basadas de acuerdo a la lógica del código y la respectiva figura creada, así que es importante primero analizar el código, para tener predicciones sobre lo que el programa hace cuando se compila.

(f) Capturen la pantalla.



* 1. Comparen la figura pintada en 1. con la figura capturada en 2. ,

(a) ¿son iguales?

Si

(b) ¿por qué?

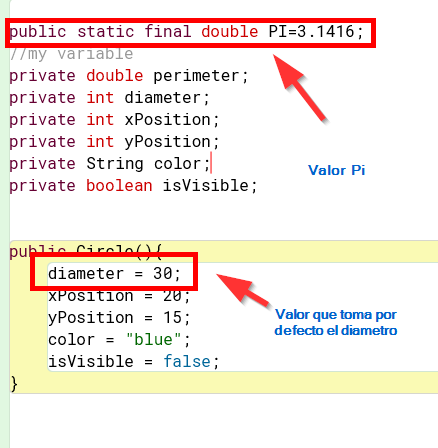
Son iguales porque para generar la figura correspondiente se utilizaron dos métodos o herramientas diferentes:

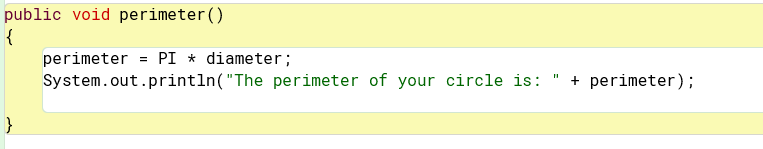
* main = Se hizo el método main dentro de la clase Rectangle para que al momento de compilar generara el apartado de void main(String [] args) y después de darle ok, creara la figura.
* show code pad = Se compiló el código para mirar el comportamiento de métodos individuales de la clase Rectangle, sin necesidad de compilar todo el programa; generando el mismo resultado.

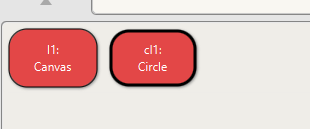
1. Clic derecho sobre la clase
2. Clic derecho sobre el objeto
3. Hacer clic derecho sobre el objeto.
4. Menú. View-Show Code Pad.

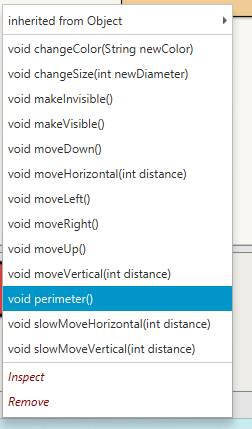
## Extendiendo una clase. Circle.

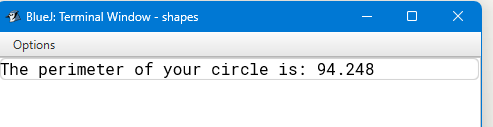
**[En lab01.doc y \*.java]**

* 1. Desarrollen en Circle el método perimeter(). ¡Pruébenlo! Capturen una pantalla.







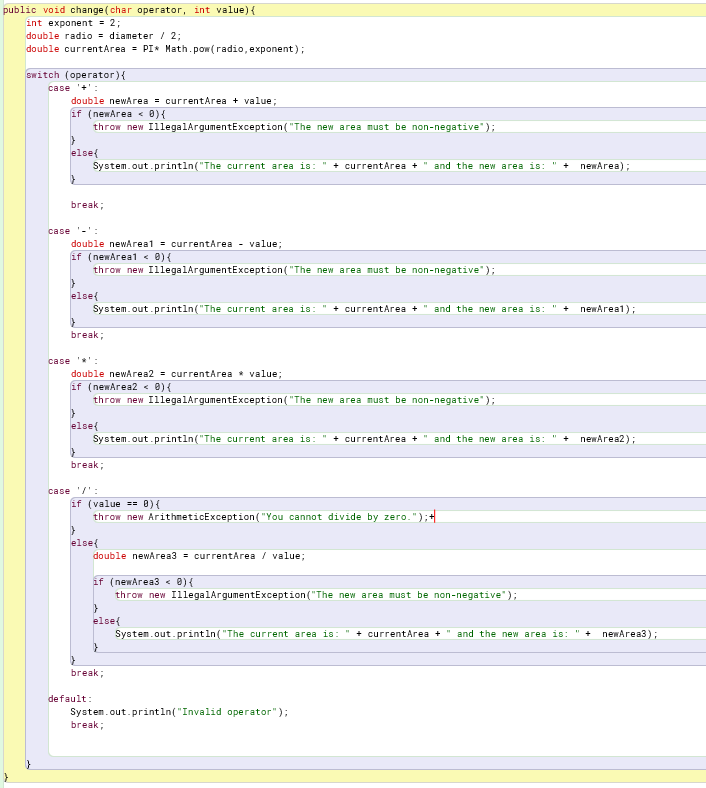


* 1. Desarrollen en Circle el método change(operator:char,value:int)

(operador: +, -, \*, /. Cambia el área del círculo operando el área actual con el valor dado como parámetro ) .

¡Pruébenlo! Capturen dos pantallas.

Prueba Andersson:



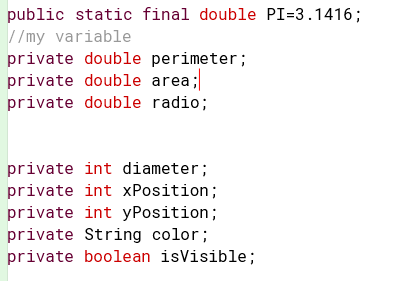
CON operator = ‘/’ y VALUE = 100

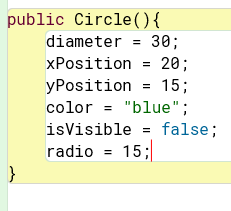


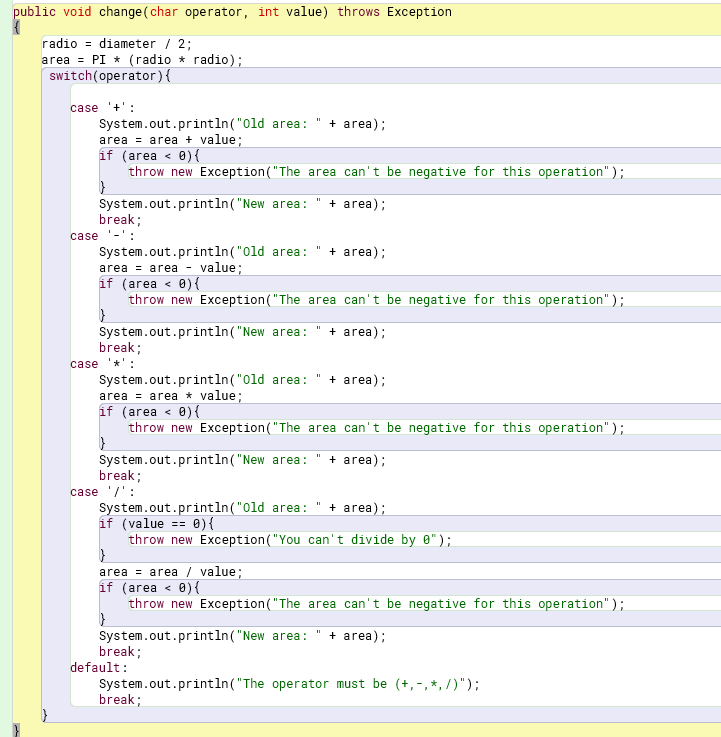
Ahora operator = ‘/’ y VALUE = 0.



Prueba Cristian:



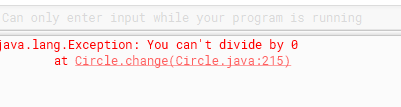




CON operator = ‘+’ y VALUE = 10



Mostrando la excepción - división por 0

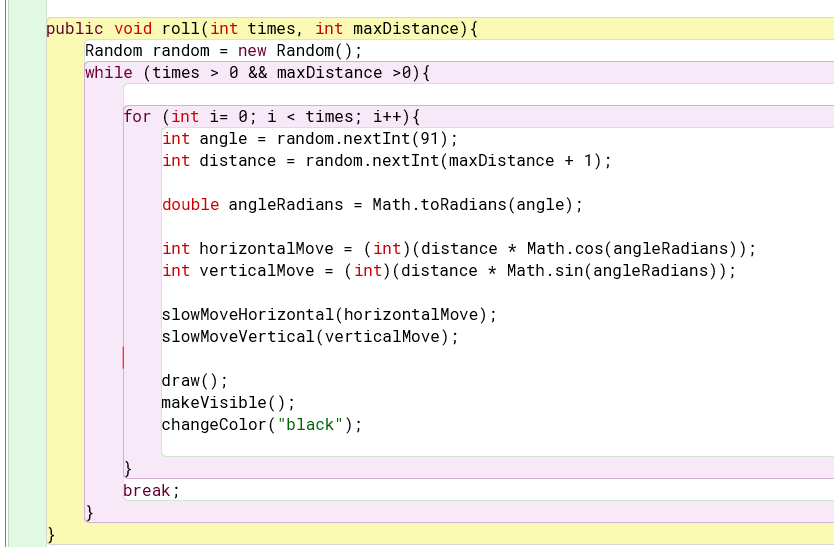


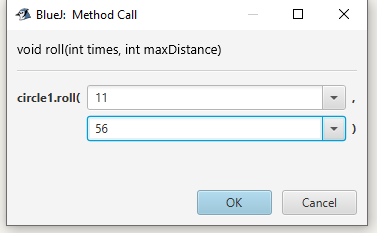
* 1. Desarrollen en Circle el método roll(times:int, maxDistance: int)

(rueda el número de veces indicado en times. La distancia y el ángulo de cada movimiento se selecciona al azar con valores menores a maxDistance y 90, respectivamente. [7](#_3znysh7) ). ¡Pruébenlo! Capturen dos pantallas.

PRUEBA ANDERSSON

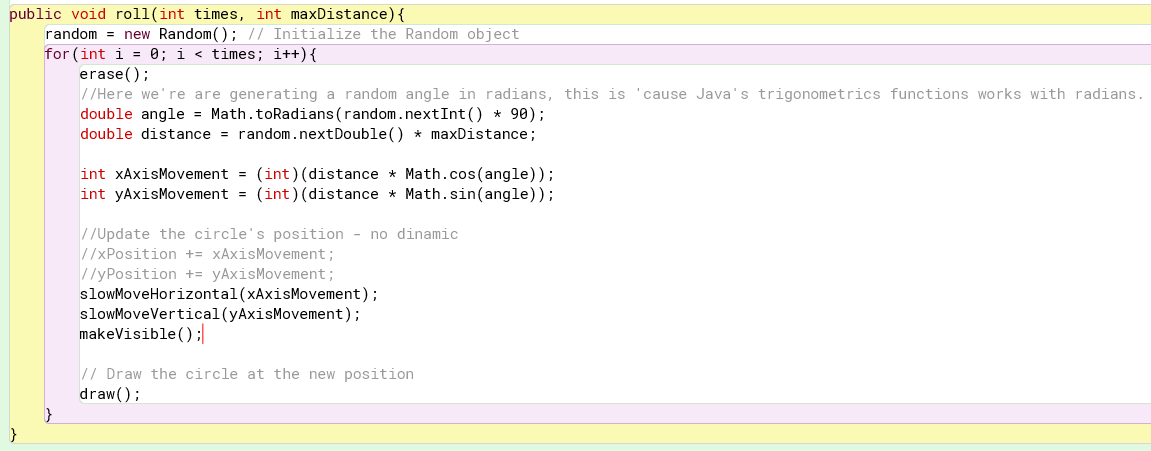
Usando la API de Math, import java.util.Math;





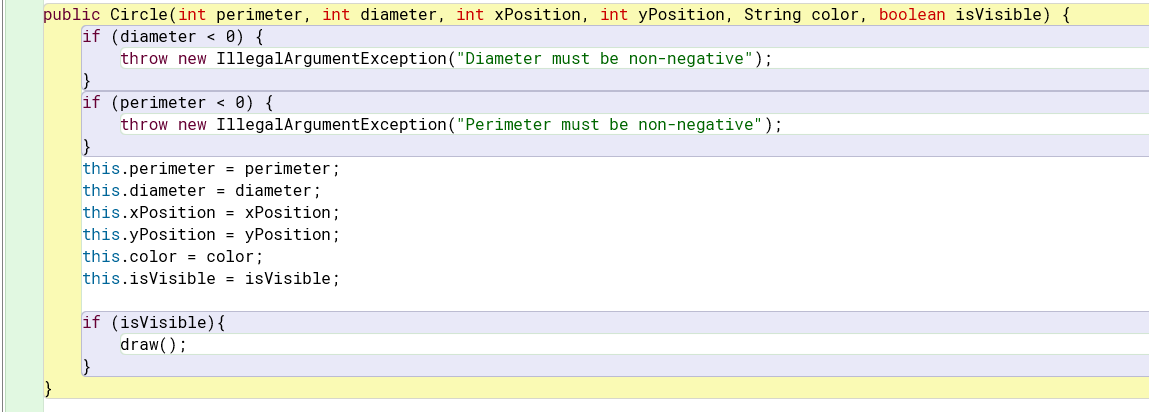
[roll Functionality 1.mp4](https://pruebacorreoescuelaingeduco-my.sharepoint.com/:v:/g/personal/andersson_sanchez-m_mail_escuelaing_edu_co/ETshF7cl2vBIi7NYGrqPbGEBv8YZPghztLudRqC5_S8C8Q?e=60mnOq&nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiLCJyZWZlcnJhbFZpZXciOiJTaGFyZURpYWxvZy1MaW5rIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXcifX0%3D)

Prueba Cristian - Mismas condiciones Andersson

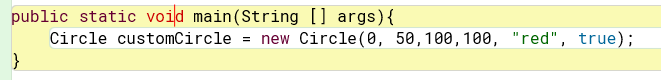


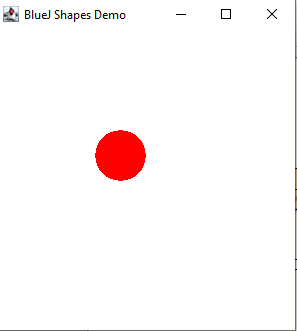
* 1. Desarrollen en Circle un nuevo creador que permita crear un círculo dada toda su información.¡Pruébenlo! Capturen una pantalla.

Para desarrollar el nuevo creador (constructor), es decir, el método para inicializar objetos y crear el círculo se requiere de GETTERS(en este caso no se usarán porque no queremos obtener el valor de las variables) Y SETTERS(queremos es setearlos para crear un círculo nuevo).



Ahora se va a probar la creación del nuevo círculo con el main:





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

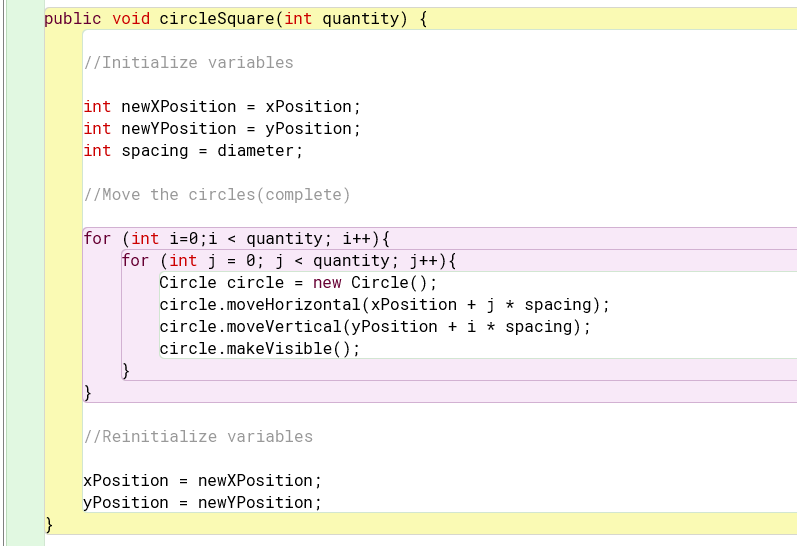
El método consiste en que dada una cantidad (quantity) tiene que dibujar un cuadrado de lado **quantity**, pero con círculos.

En el código se especifica que la separación entre cada círculo es el diámetro, por comodidad y precisión se hizo esto.

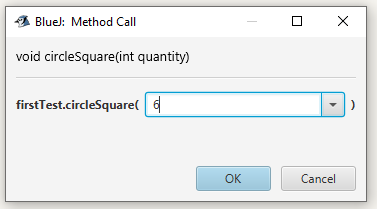
Hay dos códigos, uno desde la perspectiva de generar el cuadrado lleno, y el otro solo con los bordes.

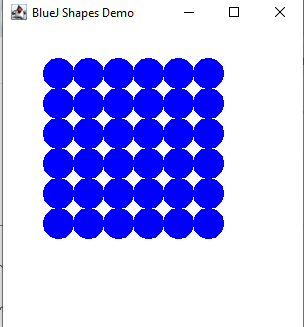
FIRST TEST



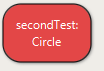


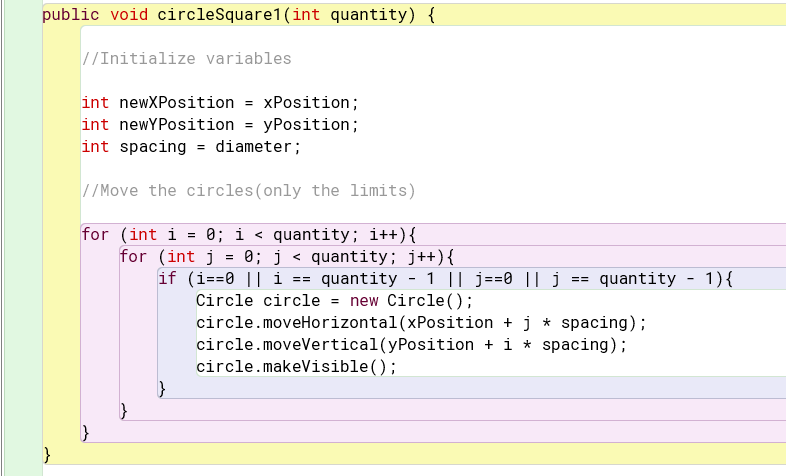




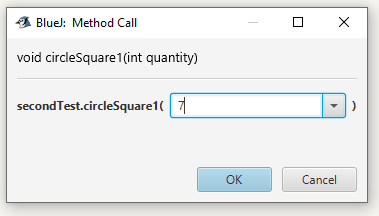


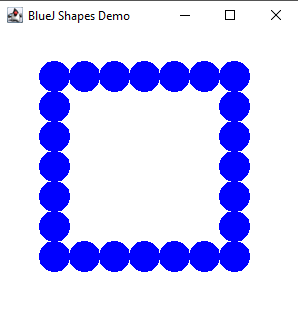
SECOND TEST





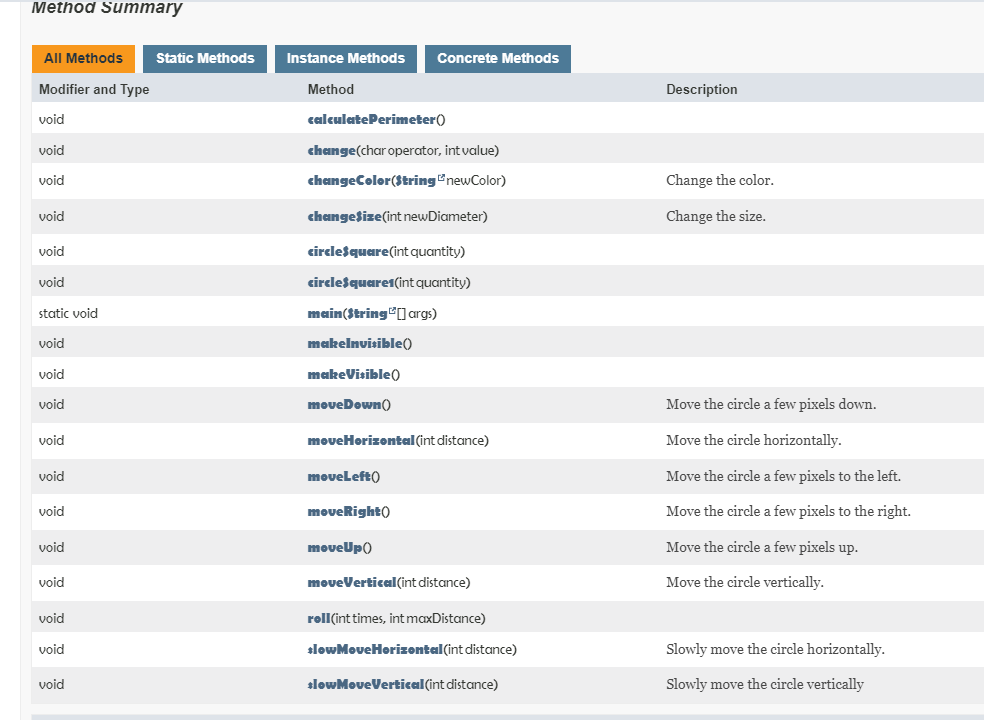


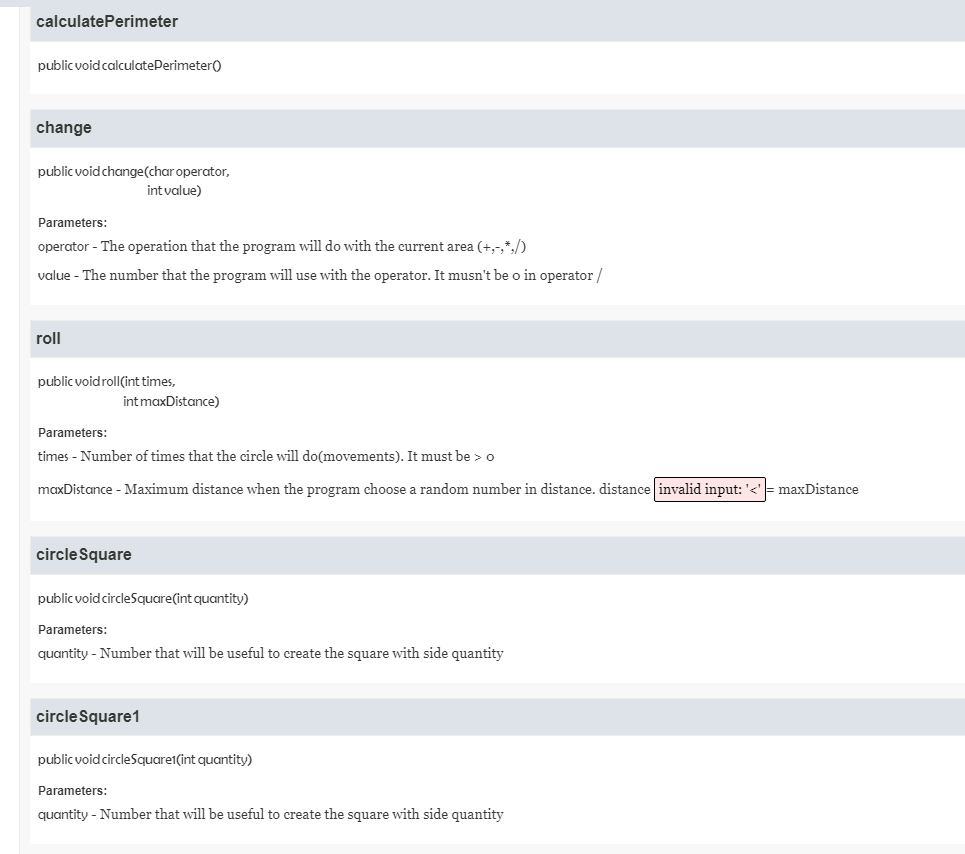




* 1. Generen nuevamente la documentación y revise la información de estos nuevos métodos. Capturen la pantalla.







La documentación se genera en el mismo orden en el que el código está estructurado, en este caso, hablando de los métodos con sus respectivos parámetros y su descripción.

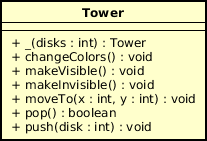
## Usando un paquete. shapes

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

En este punto vamos a crear torres de Hanoi con discos de colores. El diseño gráfico lo definen ustedes. Estos son algunos ejemplos.

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

Las tres figuras deben ofrecer los siguientes métodos.



**Mini-ciclo: 1**

\_(disks:int):Tower makeVisible() makeInvisible() **Mini-ciclo: 2** moveTo changeColors

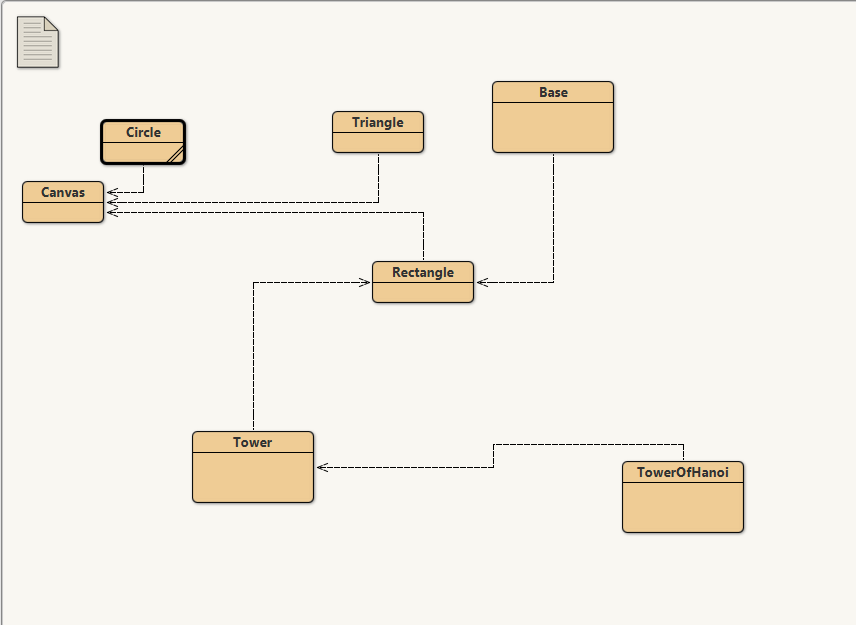
**Mini-ciclo: 3**

**push**

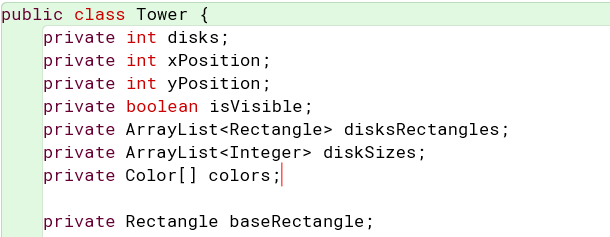
**pop**

El diseño que escogimos es usando la clase Rectangle para la creación de las nuevas clases Tower & TowerOfHanoi, y que cada disco sea de forma rectangular, la única diferencia es que no dejamos espacio entre cada disco. Así la figura es:





* 1. Inicie la construcción únicamente con los atributos. Adicione pantallazo con los atributos.



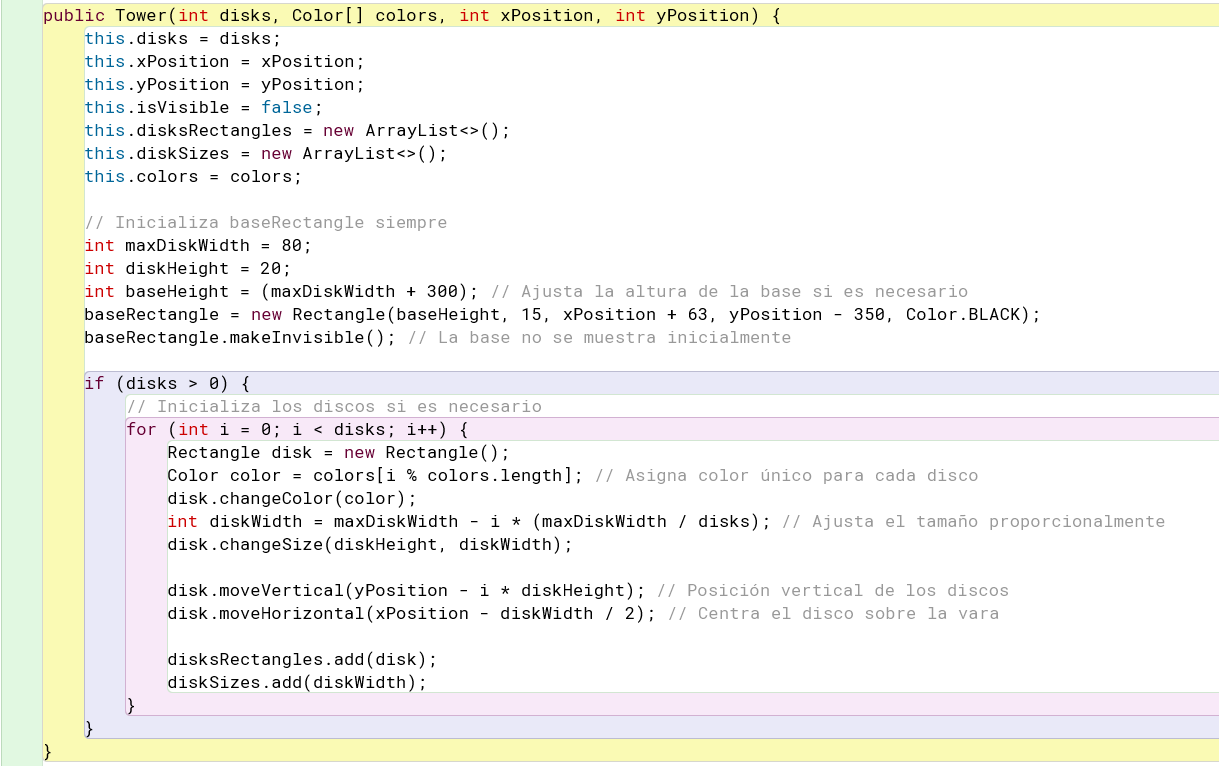
* 1. Desarrollen la clase considerando los 3 mini-ciclos. Al final de cada mini-

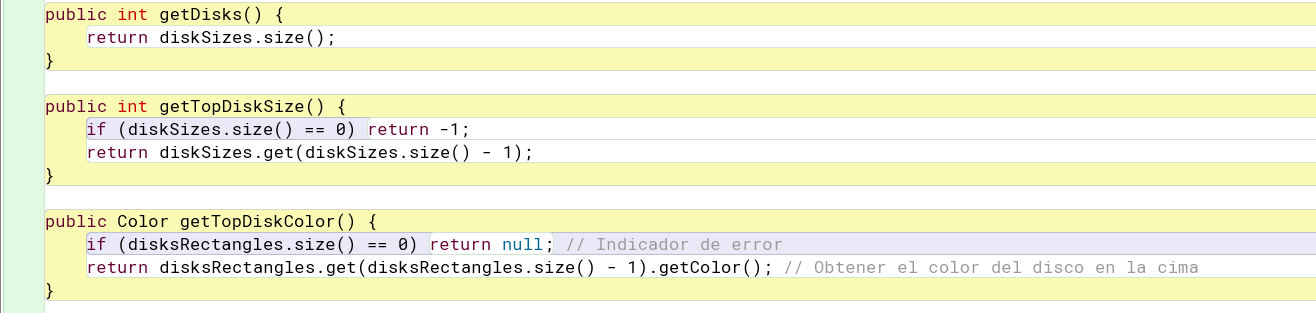
ciclo realicen dos pruebas indicando su propósito. Capturen las pantallas relevantes.

MINI-CICLO: 1

\_(disks:int): Tower

**Método constructor para disk:**

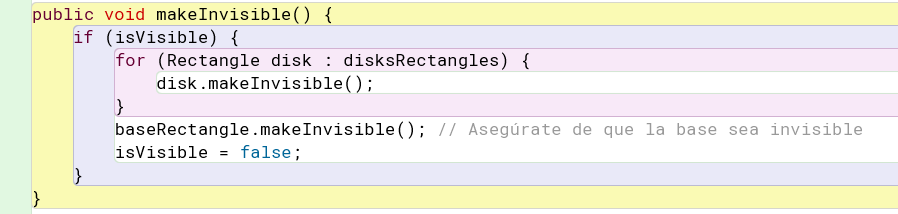
****

****

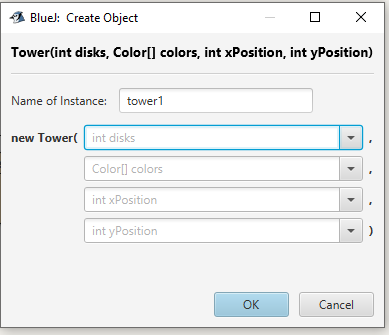
makeVisible();



makeInvisible();

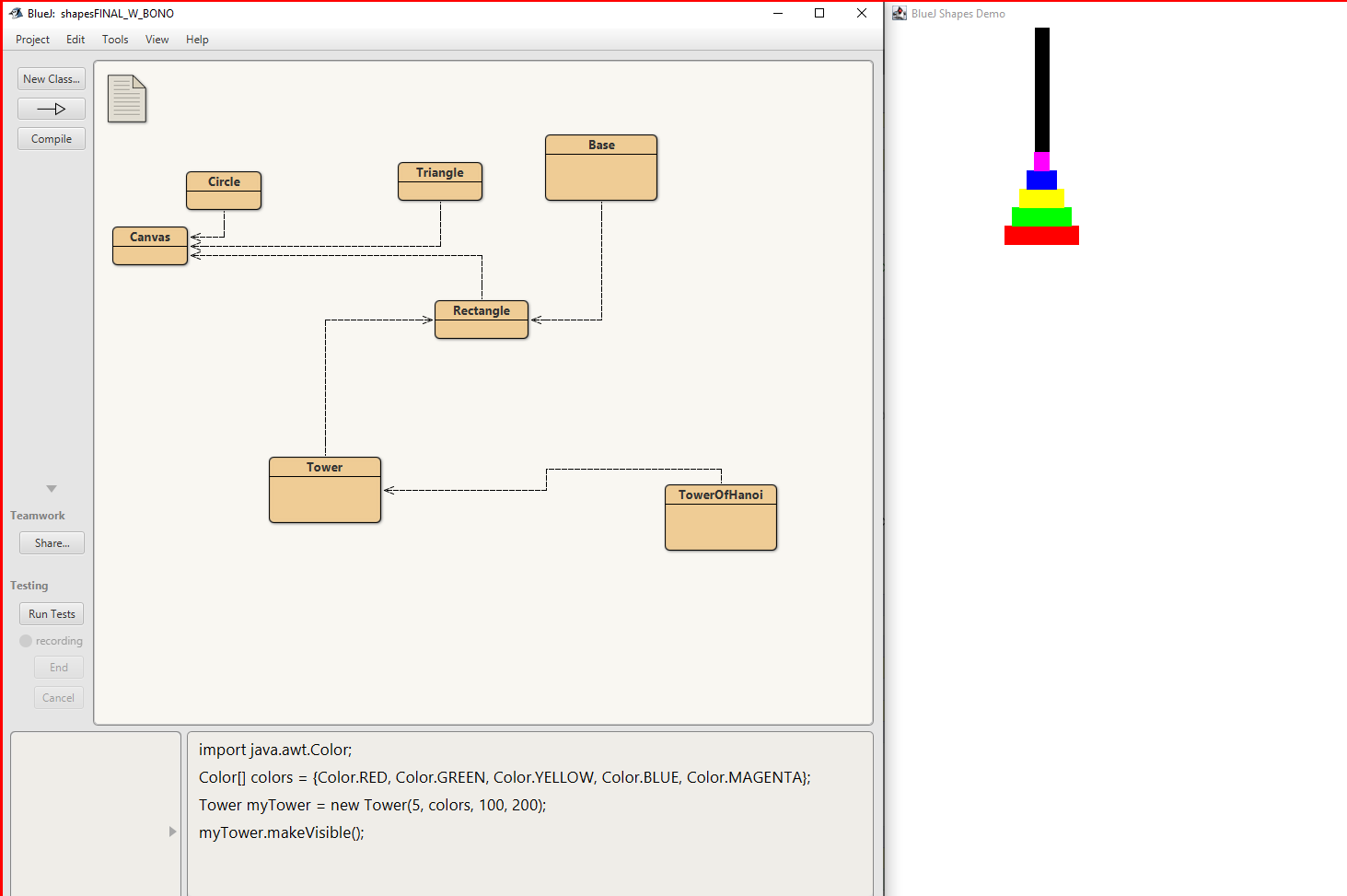


**PRUEBAS MINI-CICLO 1**

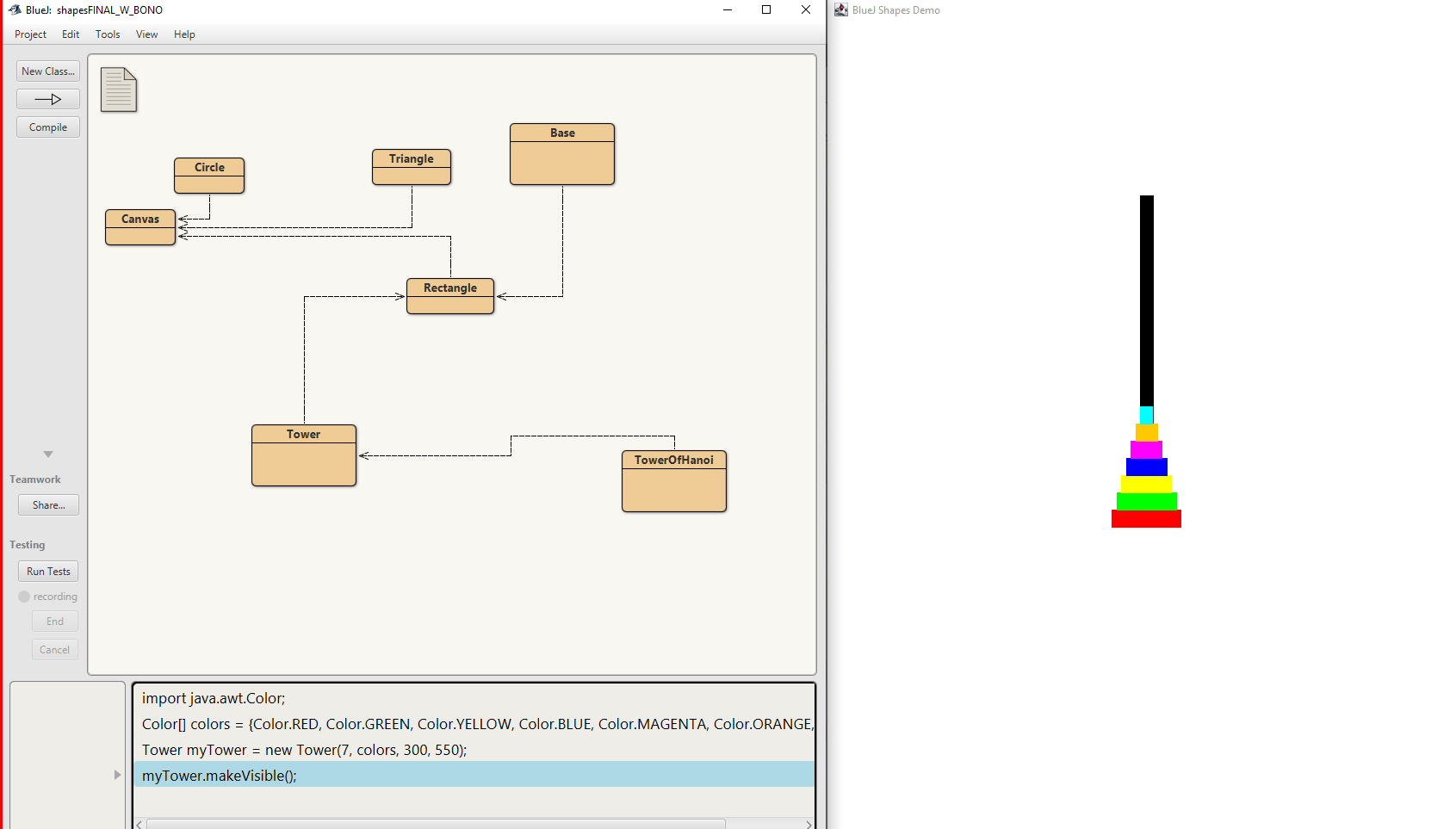
****

CON SHOW CODE PAD SE INSTANCIA UNA NUEVA TORRE PARA PROBAR LOS MÉTODOS Y LA CREACIÓN DE CADA DISCO, SE VAN A MOSTRAR 3 EJEMPLOS CON DISTINTA CANTIDAD DE DISCOS, DIFERENTES COLORES Y CON LOS DOS MÉTODOS.

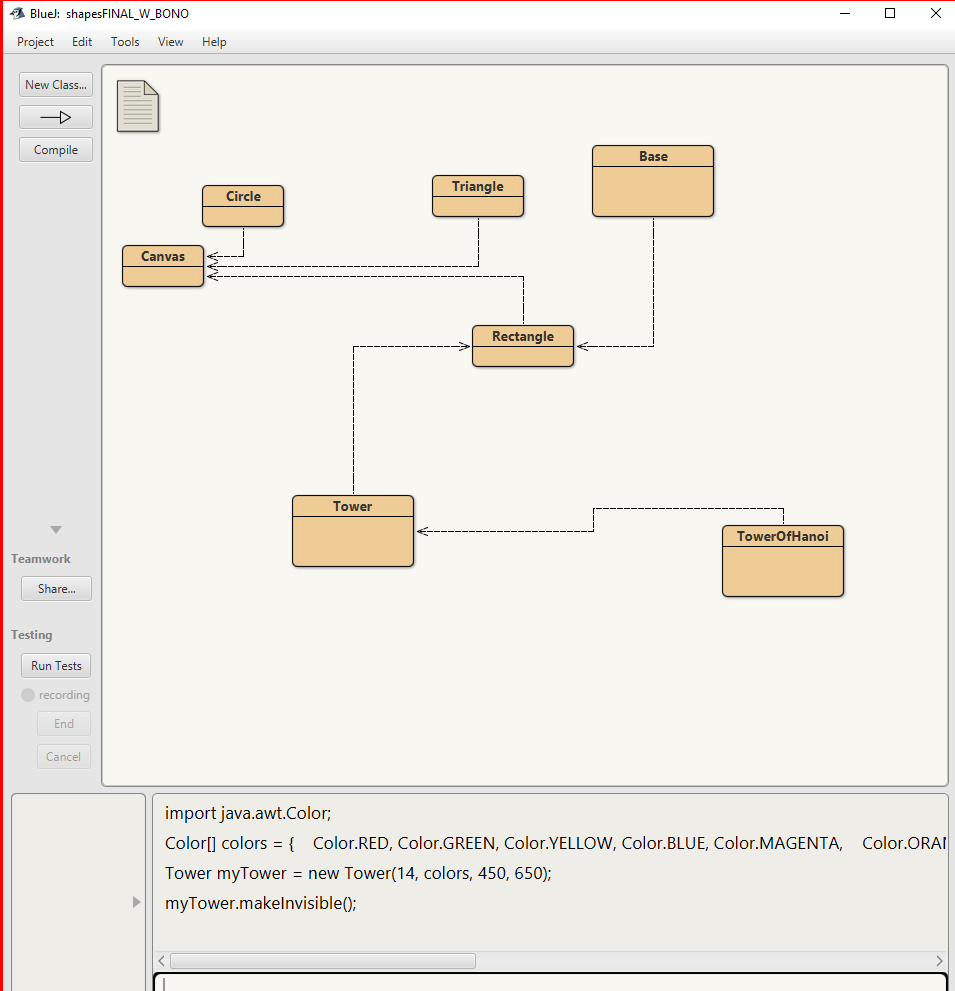
1.

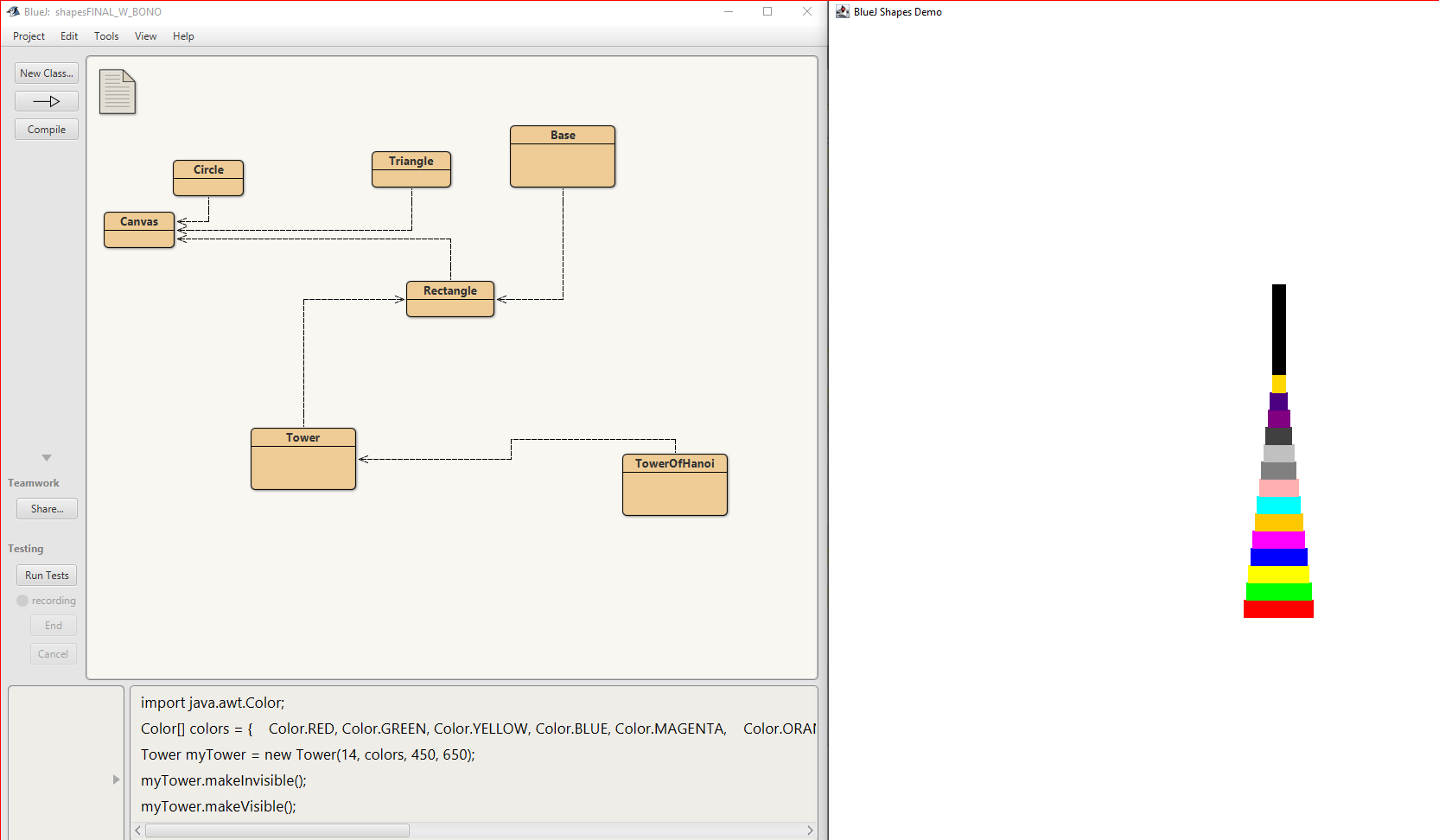


2.



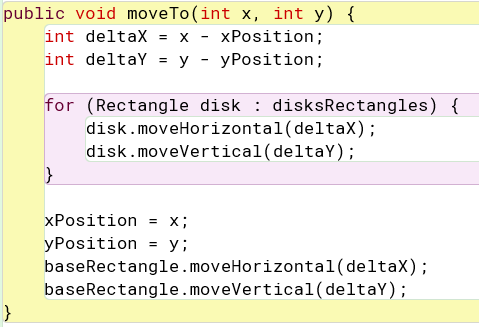
3.



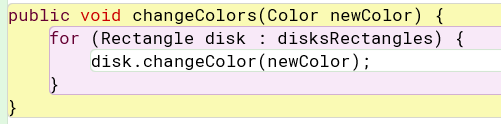


MINI-CICLO: 2

moveTo(int x, int y);



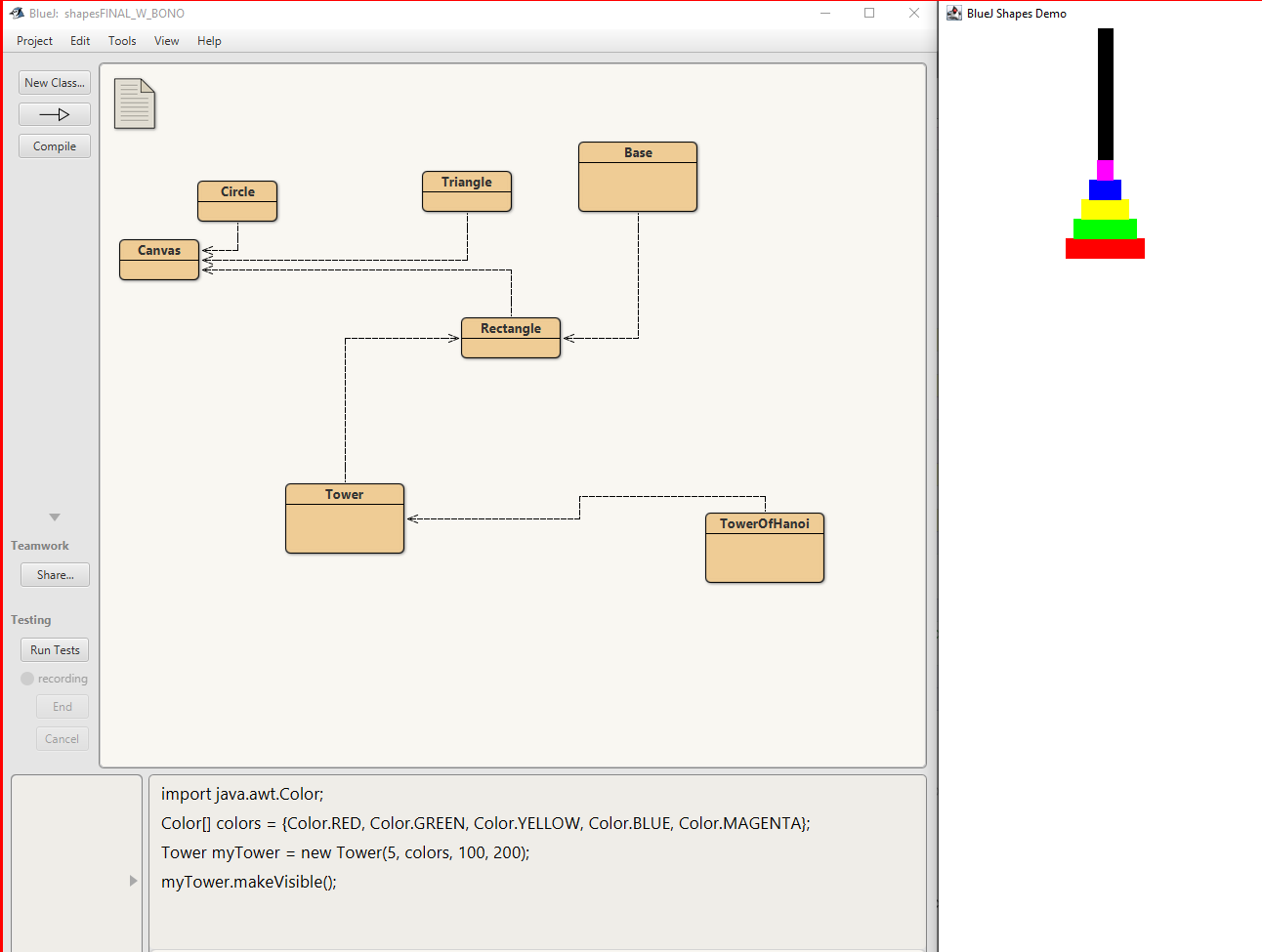
changeColors();

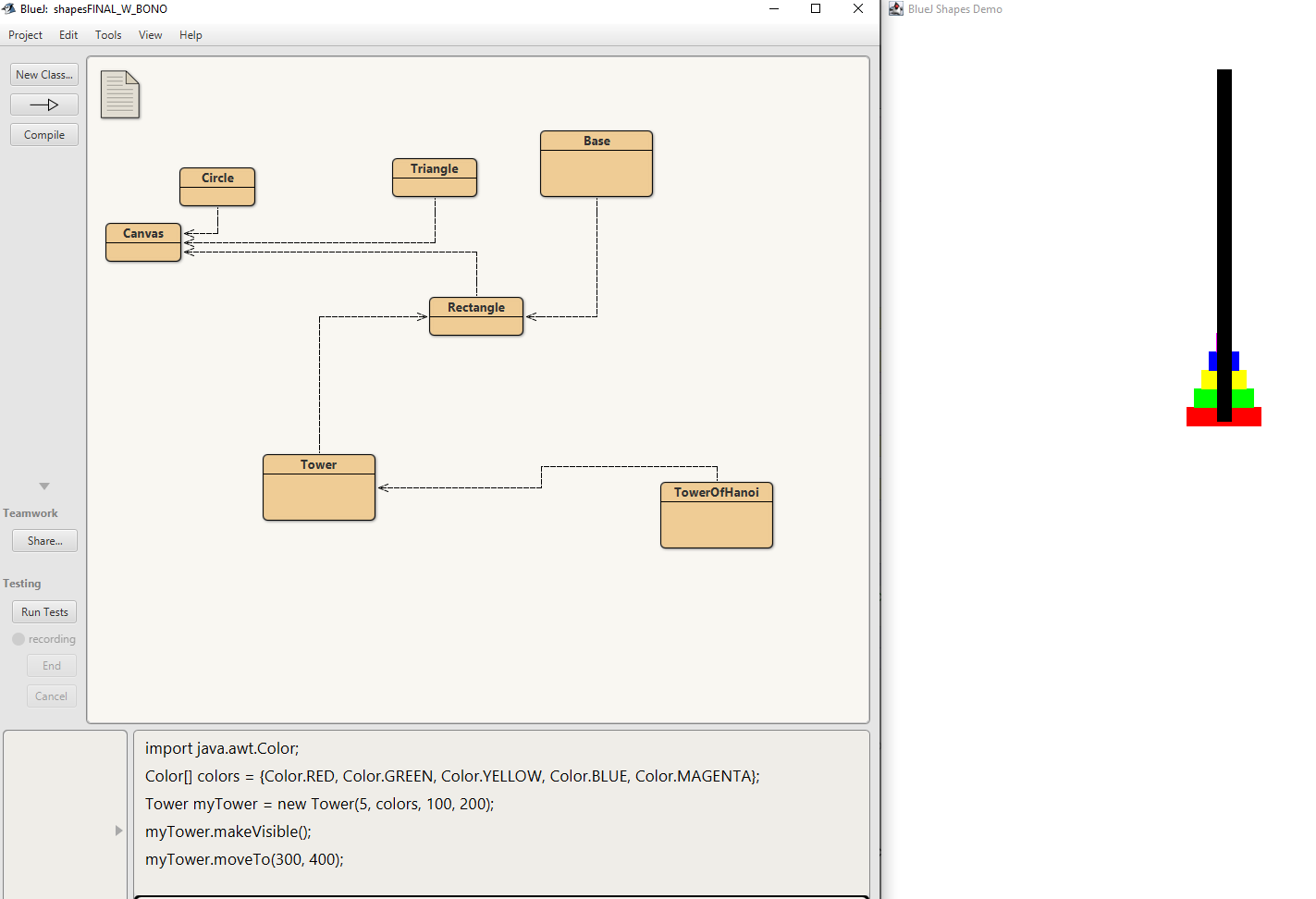


**PRUEBAS MINI-CICLO 2**

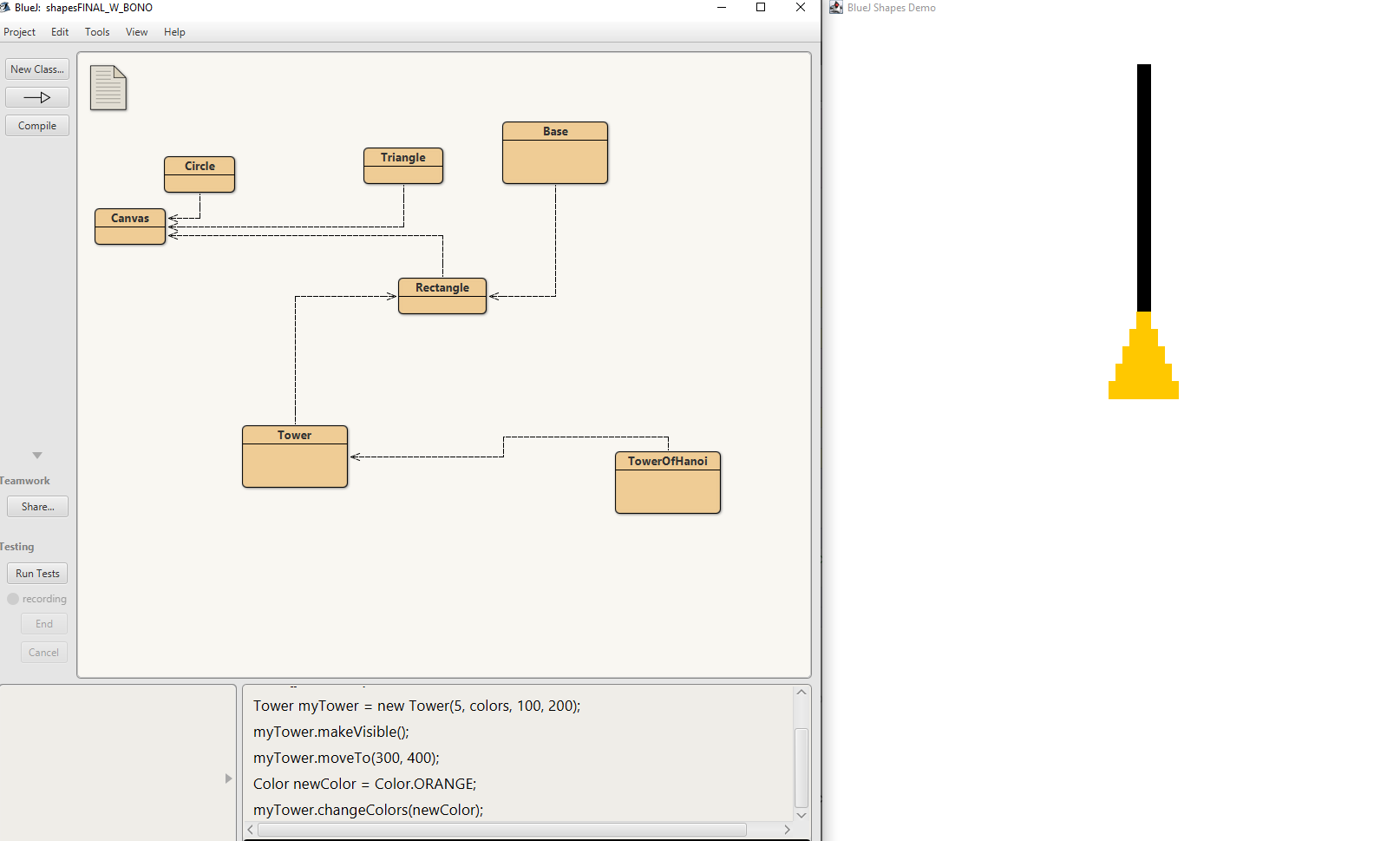
Ya que funcionó la creación de torres con distinta cantidad de discos, entonces en este caso se van a hacer dos ejemplos en donde muestre la aplicación de estos dos métodos, pero a la misma cantidad de discos.

* **Con moveTo():**

****

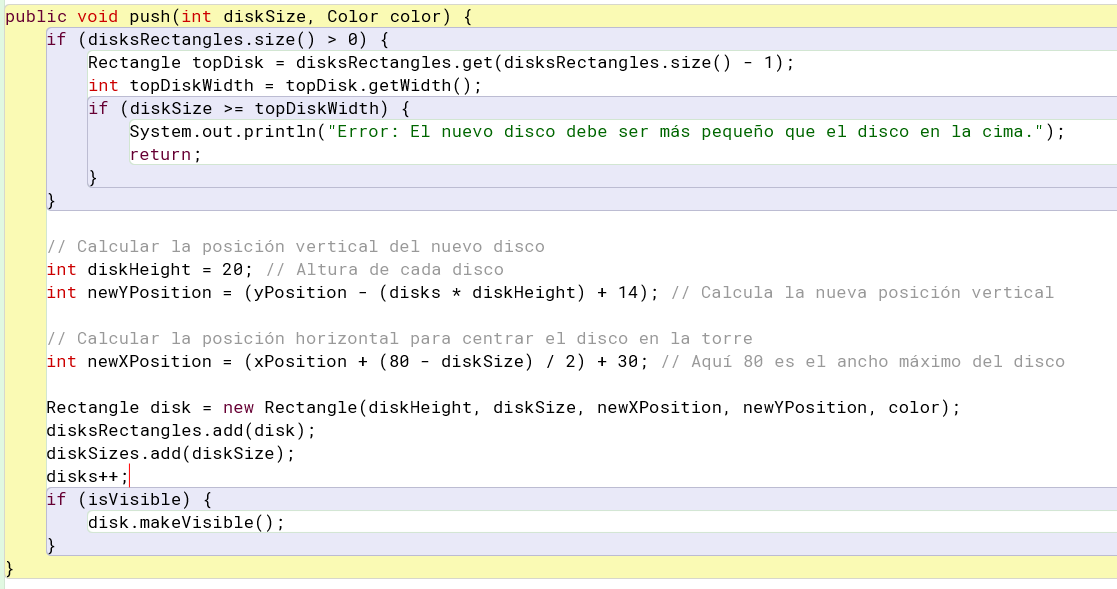
****

* **Con changeColors():**

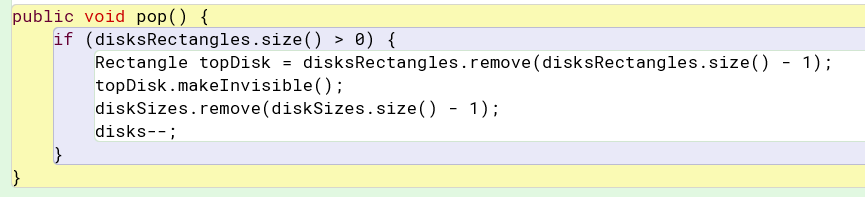


MINI-CICLO: 3

push(int disks);



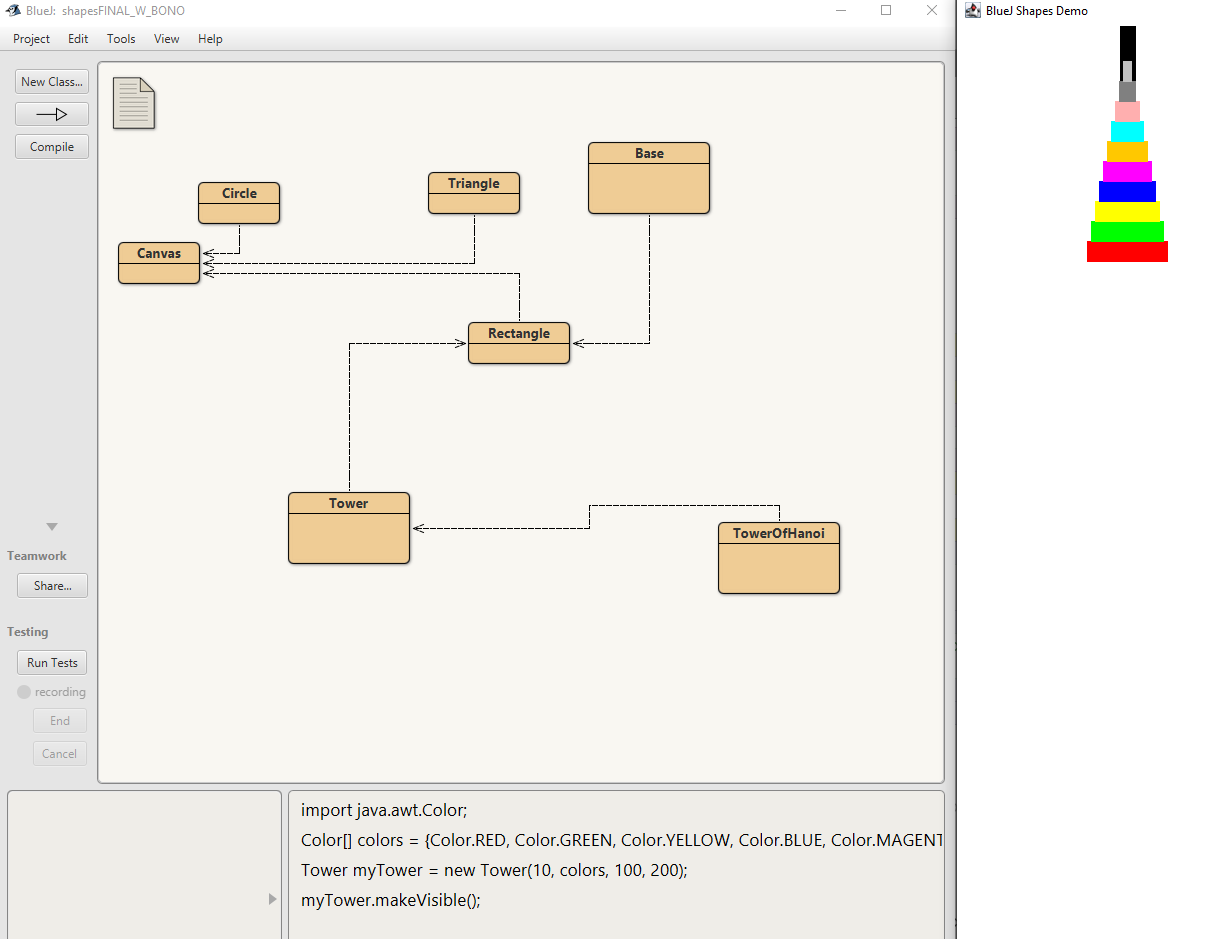
pop();

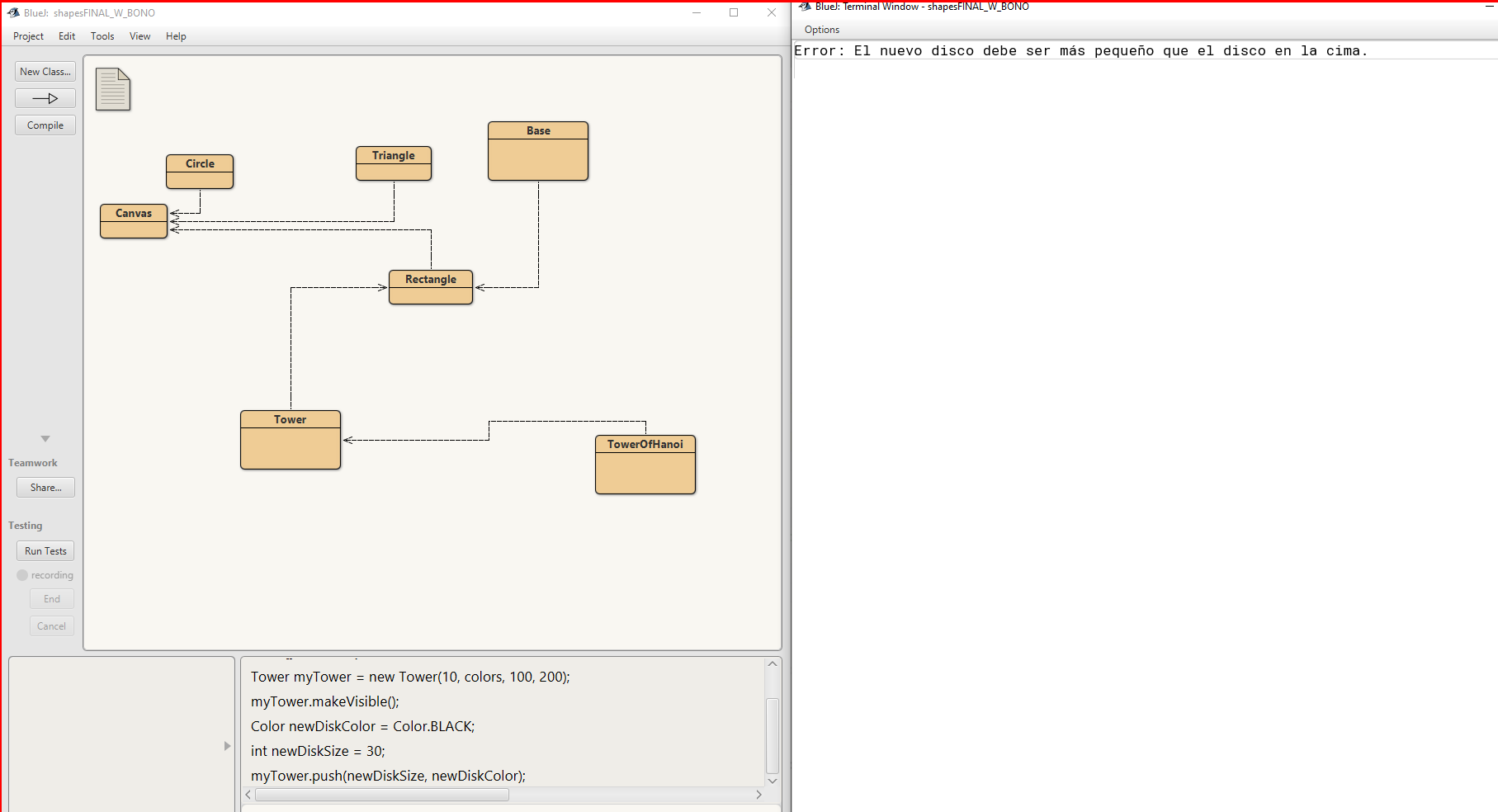


**PRUEBAS MINI-CICLO 3**

Con la misma idea, pero con otra cantidad de discos, se mirará qué pasa cuando se quita o se añade un disco, dependiendo del tamaño de este.

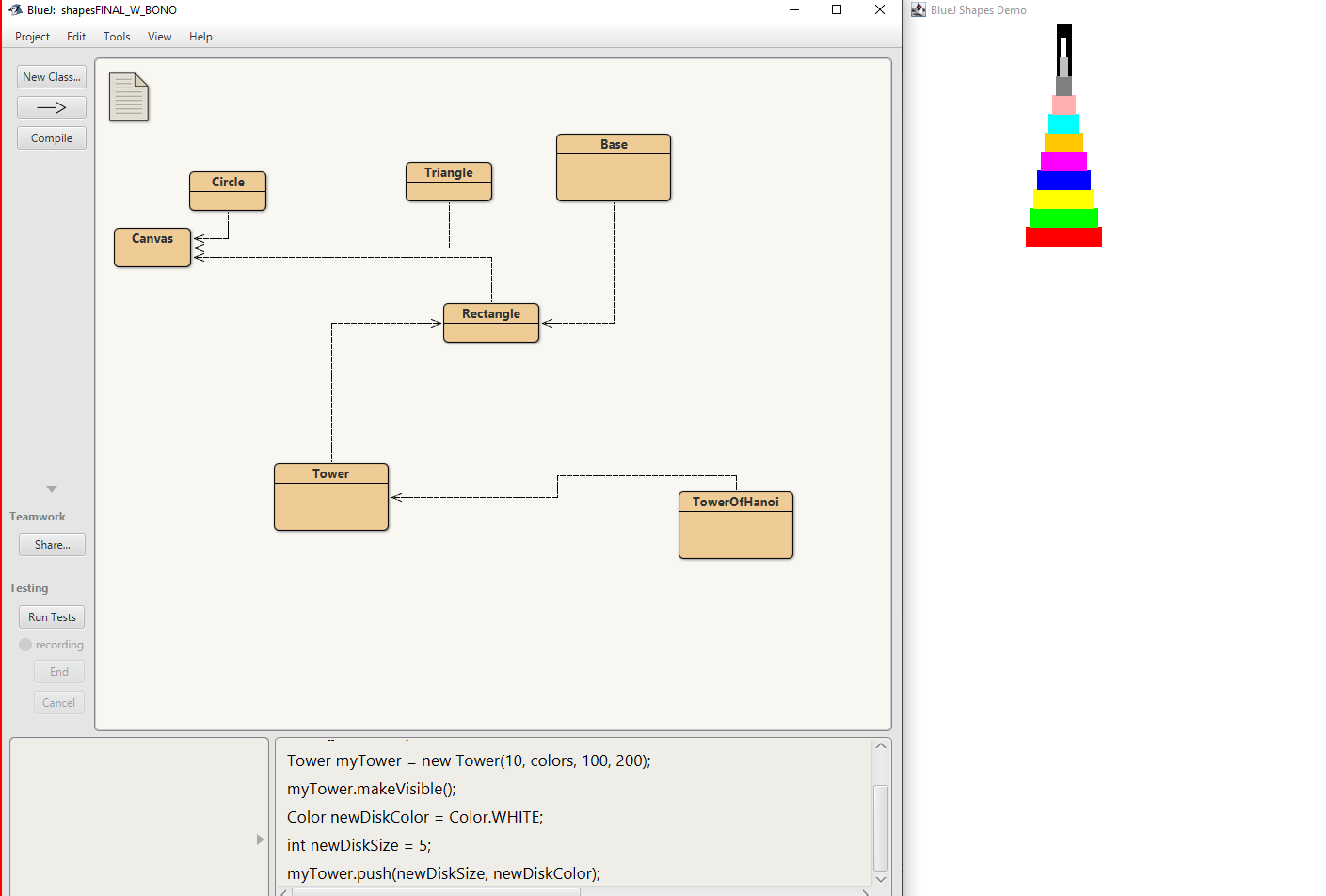
* Con push():



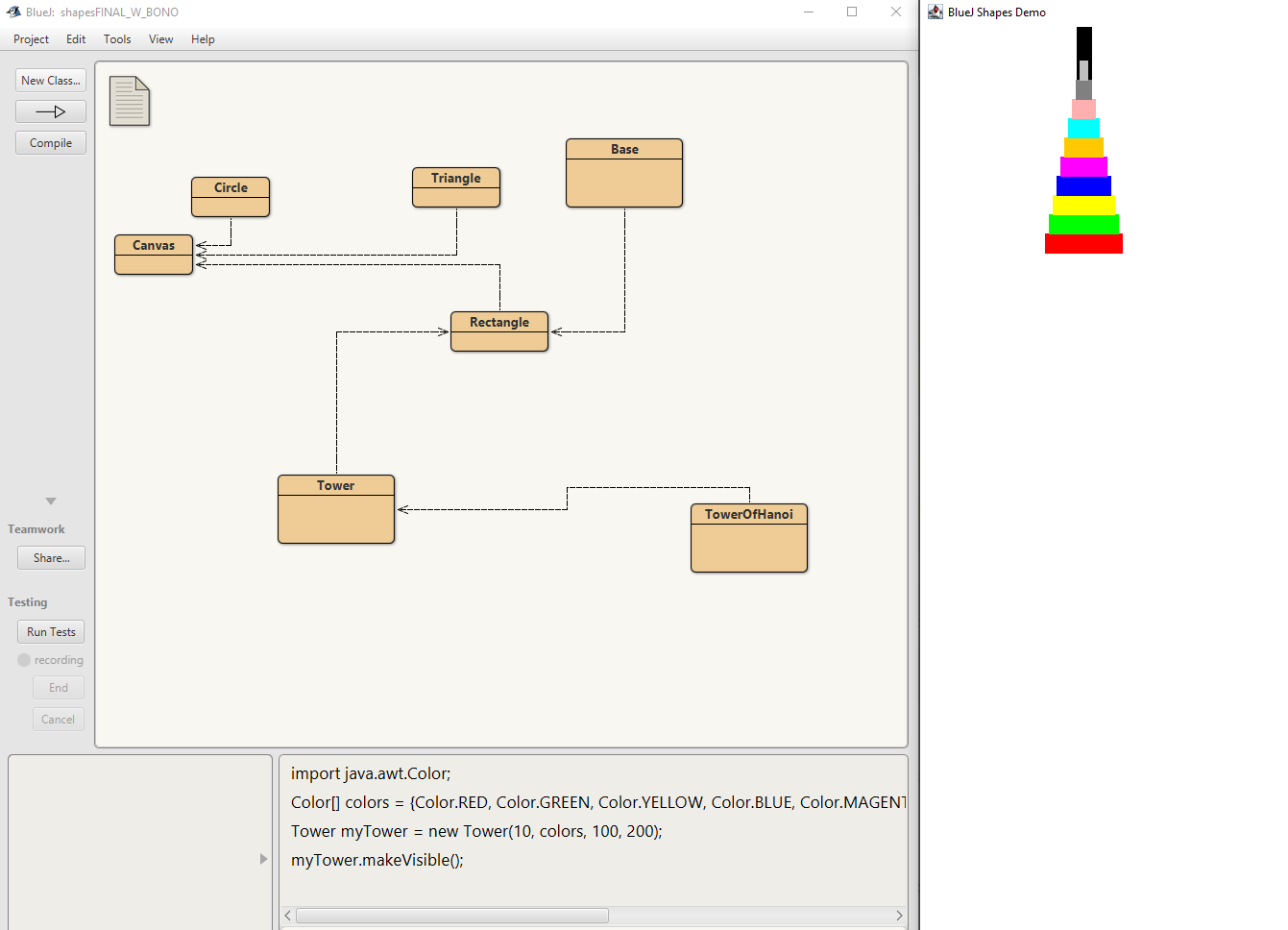


GENERA ESTE ERROR, LO CUAL ESTÁ BIEN PORQUE EL DISCO NUEVO NO DEBE SER MÁS GRANDE QUE EL QUE ESTÁ EN LA CIMA.

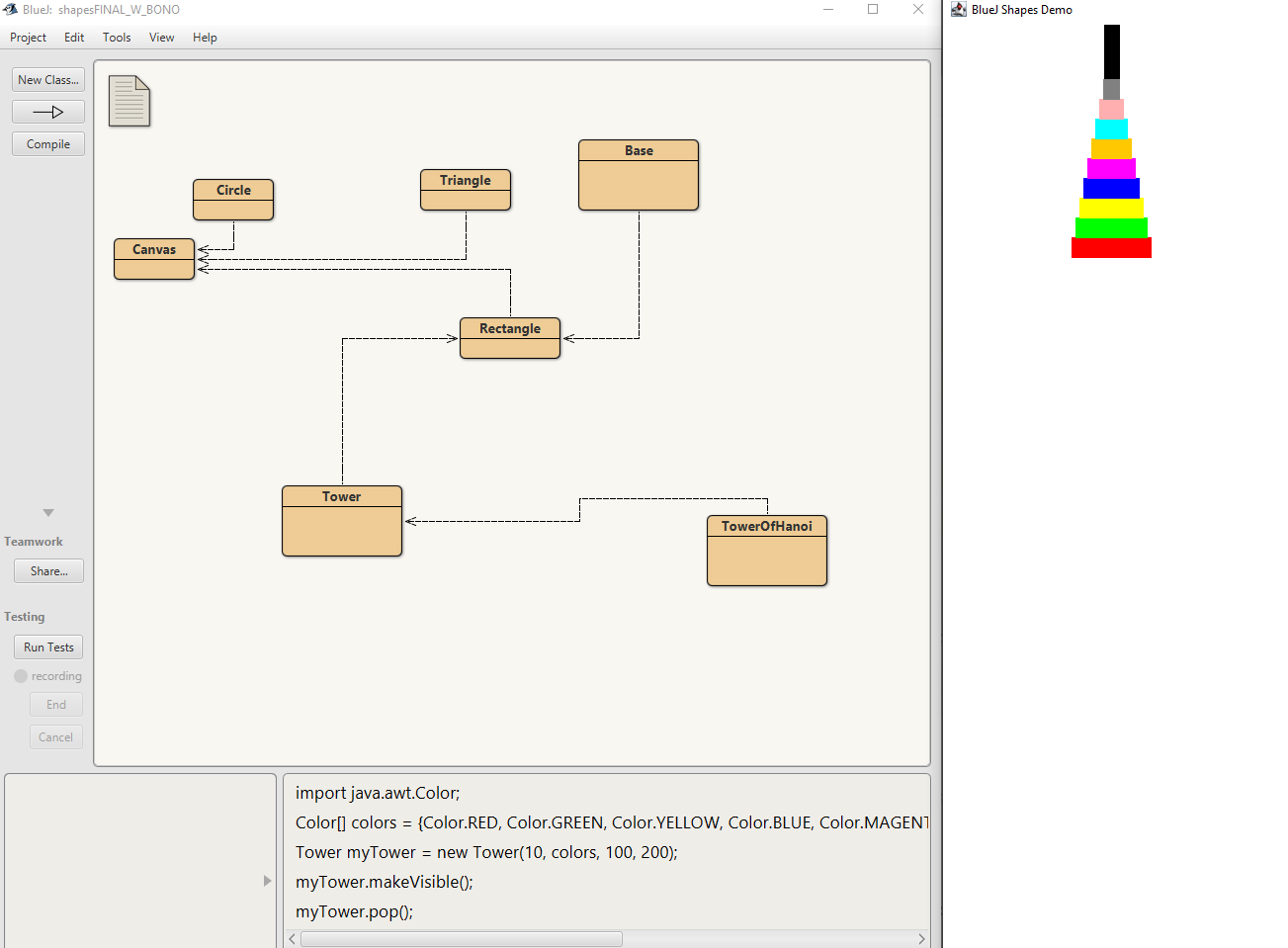
Con un disco de tamaño menor, si lo crea



* Con pop():



Listo, ahora con el método pop() aplicaría la política de datos(LIFO- Last In First Out), es decir, se quita el disco que está en la cima.



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

## F. De python a java prompts

**[En lab01.doc]**

En este punto vamos usar un recurso de prompts para la transición de python y java y evaluarlo en la encuesta preparada con ese objetivo. Uno del equipo diligencia la encuesta.

¿En cuál punto de la encuesta discutieron más? Expliquen.

El punto de la encuesta en el que más discutimos fue en la pregunta sobre los **errores encontrados en las respuestas generadas**. Este tema generó bastante debate porque ambos teníamos percepciones diferentes sobre qué errores eran los más significativos y cómo afectan la calidad de las respuestas. Mientras uno de nosotros se centraba más en los errores técnicos y de precisión en los ejemplos de código, el otro estaba más preocupado por la claridad de las explicaciones y la comparabilidad entre los lenguajes, lo que llevó a una discusión sobre qué aspectos eran realmente prioritarios al evaluar las respuestas. A pesar de nuestras diferencias, logramos llegar a un consenso, pero fue definitivamente el punto donde nuestras opiniones más divergieron.

Por ejemplo, al analizar la respuesta sobre la comparación de los tipos de datos entre Java y Python, uno de nosotros señaló que el tamaño de los tipos de datos en Java no estaba bien especificado en la tabla comparativa, lo que podría llevar a confusión para alguien que no esté familiarizado con estos conceptos. Mientras tanto, el otro compañero argumentó que, aunque era importante, la prioridad debía ser explicar de manera clara cómo se usan esos tipos en ambos lenguajes, y no tanto enfocarse en los detalles técnicos como el tamaño. Este desacuerdo sobre si debíamos priorizar la precisión técnica o la claridad en la explicación generó una discusión bastante extensa, pero finalmente decidimos que ambos aspectos eran igualmente importantes y debían ser tratados con cuidado en la respuesta final.

## TORRES DE HANOI



Las Torres de Hanói es un rompecabezas o juego matemático inventado en 1883 por el matemático francés Édouard Lucas.1 Este juego de mesa individual consiste en un número de discos perforados de radio creciente que se apilan insertándose en uno de los tres postes fijados a un tablero. El objetivo del juego es trasladar la pila a otro de los postes siguiendo ciertas reglas, como que no se puede colocar un disco más grande encima de un disco más pequeño. [WIKIPEDIA]

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

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

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

**Requisitos funcionales**

* Crear el estado inicial [con un máximo de 20 colores]
* Mover un disco de una vara a otra

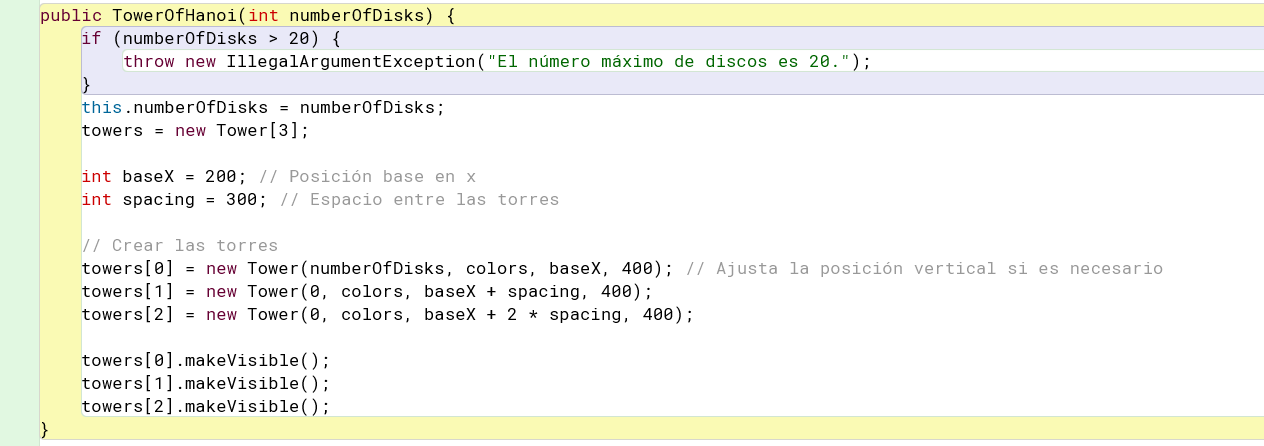
**Requisitos de interfaz**

* Las torres se identifican por su posición [1,2,3] de izquierda a derecha.
* 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.

**ATRIBUTOS INICIALES**

****

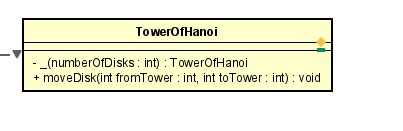
**CONSTRUCTOR**

****

**MÉTODO A OFRECER**

Solo moveDisk porque cumple de acuerdo a los requisitos funcionales y de interfaz del problema, ya si queremos implementar más métodos, por ejemplo, saber cuando el programa termina de acuerdo a la cantidad de movimientos mínimos ((2 ^n) - 1) sería otro método aparte que no está como requisito en este ejercicio. También se identifican las torres de 1 a 3 y se usa JoptionPane para identificar casos como:

* Torre no está dentro del intervalo de 1 a 3
* El número máximo de discos es 20
* No hay discos en la torre de origen
* No se puede colocar un disco más grande sobre uno más pequeño
  1. Planifiquen la construcción considerando algunos mini-ciclos.

****

**MINI-CICLO 1**

-\_(numberOfDisks:int): TowerOfHanoi

-\_(towers: List):Tower

-\_(colors: List):TowerOfHanoi

HECHO ANTERIORMENTE

**MINI-CICLO 2**

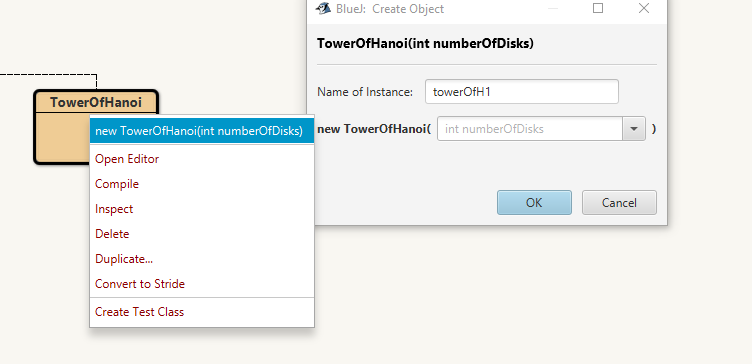
+moveDisk(int fromTower, int toTower):void



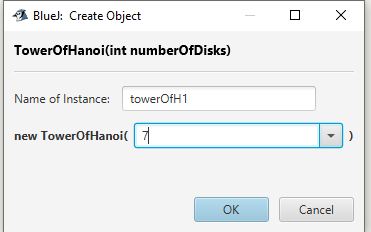
* 1. Implementen la clase . Al final de cada mini-ciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes.

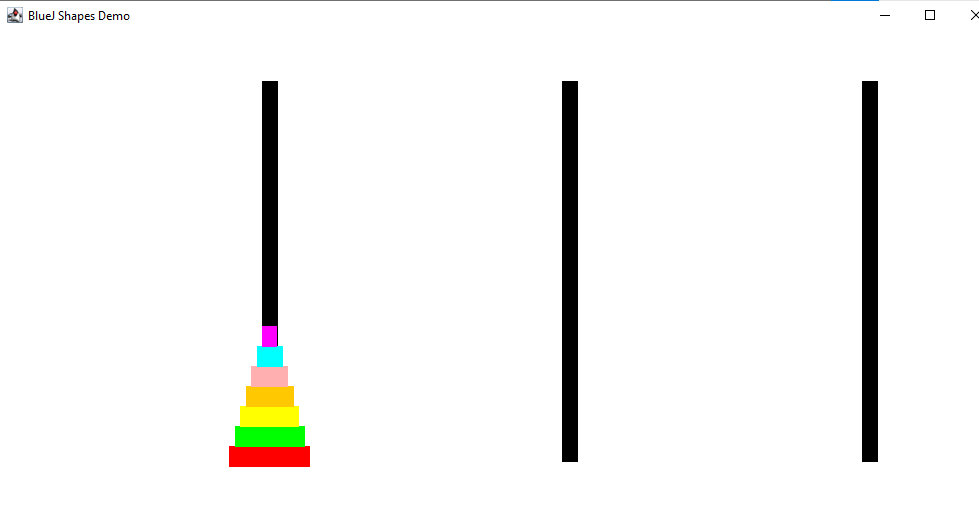
**PRUEBAS MINI-CICLO 1**

Se van a hacer 3 pruebas

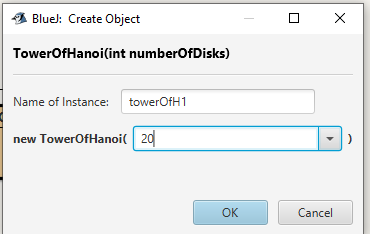


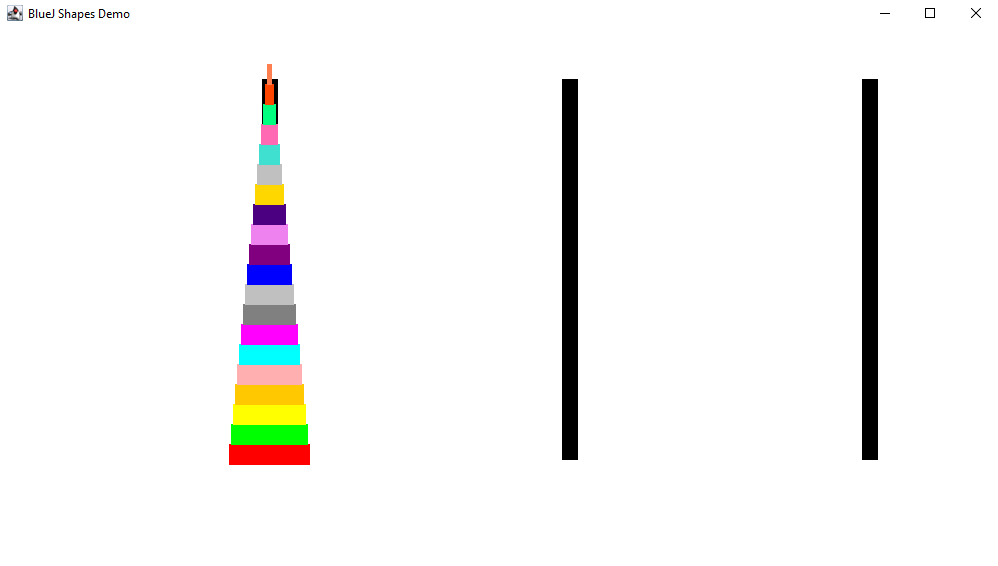
1. Con cantidad de discos = 7: Debería crear las 3 torres, en donde la primera torre tenga los discos, y las 3 torres tienen una vara que es donde van a estar soportadas, cada disco tiene color diferente.



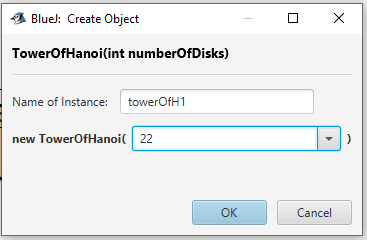


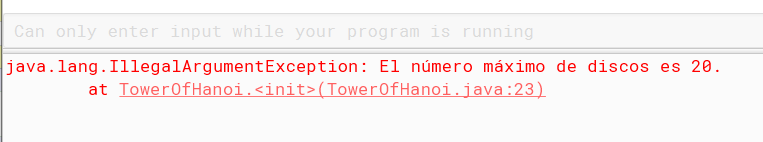
1. Con cantidad de discos = 20: Debería crear las 3 torres, en donde la primera torre tenga los discos, y las 3 torres tienen una vara que es donde van a estar soportadas, cada disco tiene color diferente.





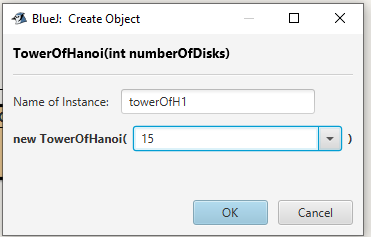
1. Con cantidad de discos = 22: Debería pasar que no deja crear los discos porque se pasa del tamaño máximo, entonces mandaría un error.



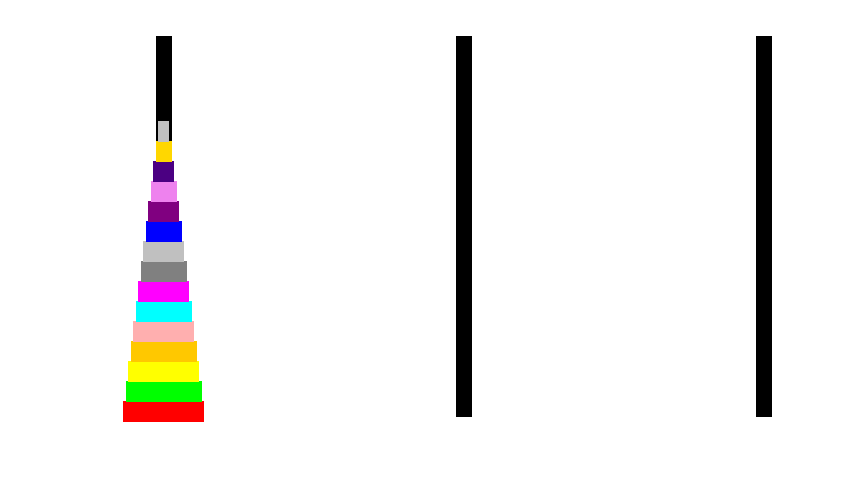


**PRUEBAS MINI-CICLO 2**

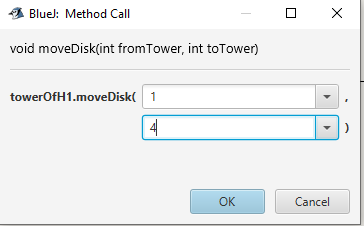
Ahora si se prueba el método moveDisk() en donde su nombre lo dice es mover los discos de una torre a otra, teniendo en cuenta las reglas definidas en el inciso anterior. Se harán pruebas mostrando cuando se incumplen las reglas y el funcionamiento de este método cuando la cantidad de discos es igual a 15.

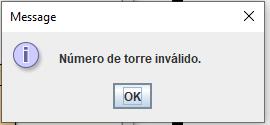


**Prueba**

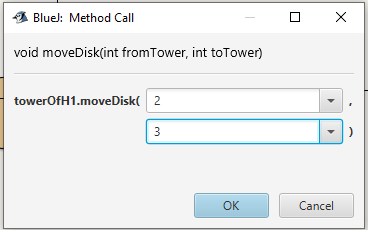
****

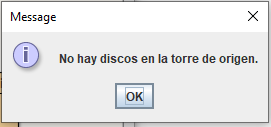
* Torre no está dentro del intervalo de 1 a 3



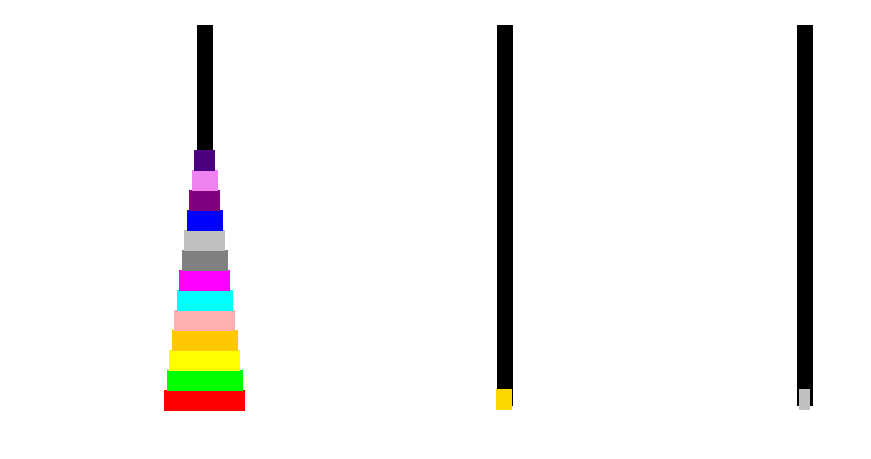


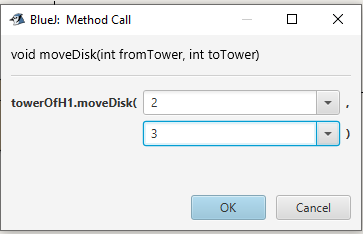
* No hay discos en la torre de origen

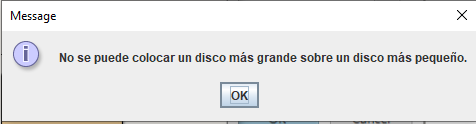




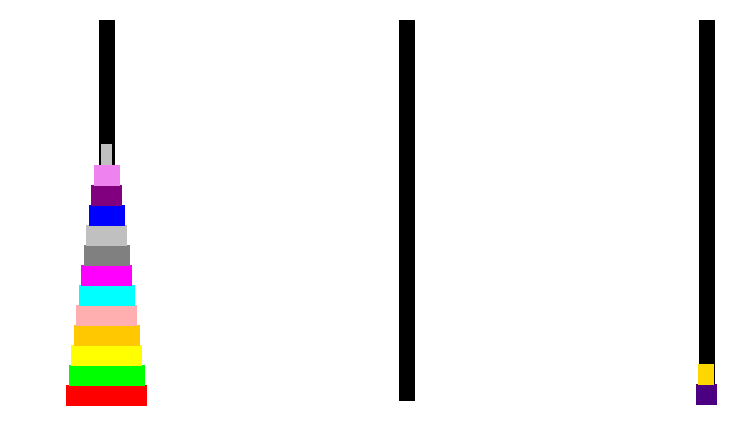
* No se puede colocar un disco más grande sobre uno más pequeño



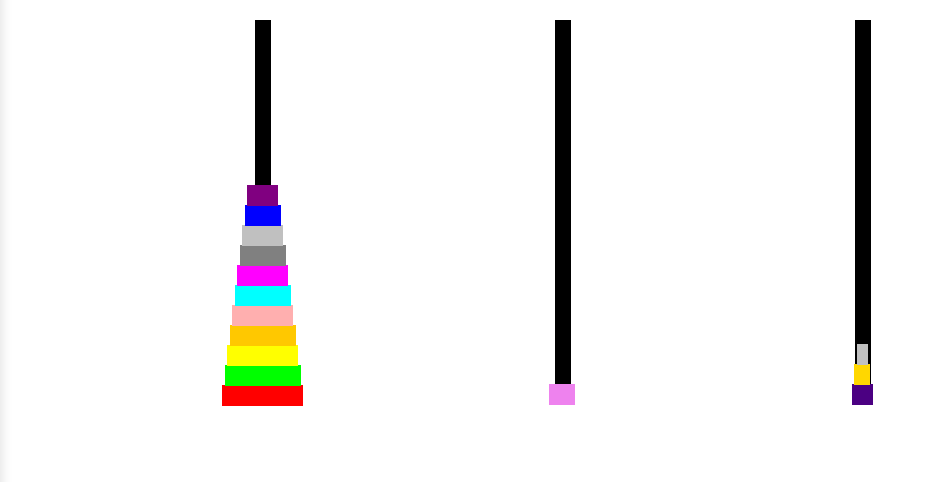




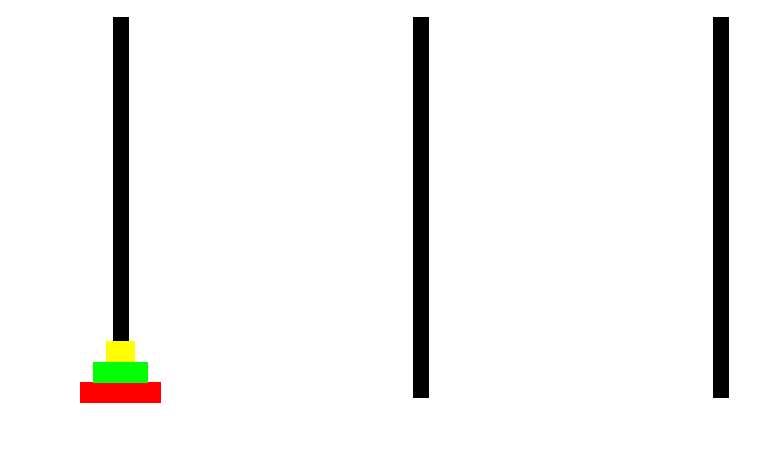
**Funcionamiento**

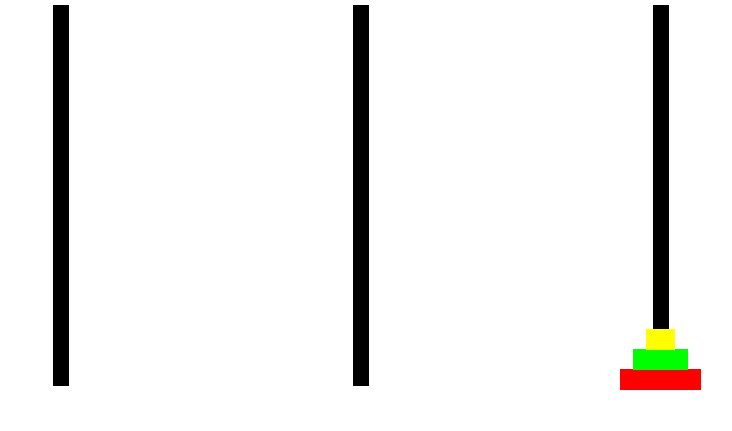
****

**Sólo se trata de mover los discos cumpliendo las reglas hasta lograr que todos los discos queden en la torre 3.**

****

**Haremos el ejemplo más fácil para verificar que todos los discos estén en la torre 3 con numDisks = 3**

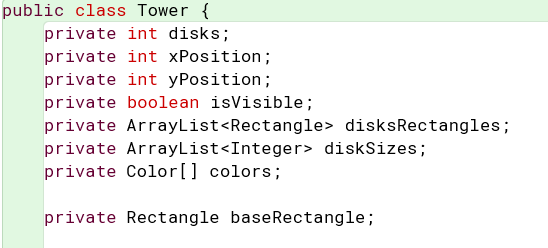
****

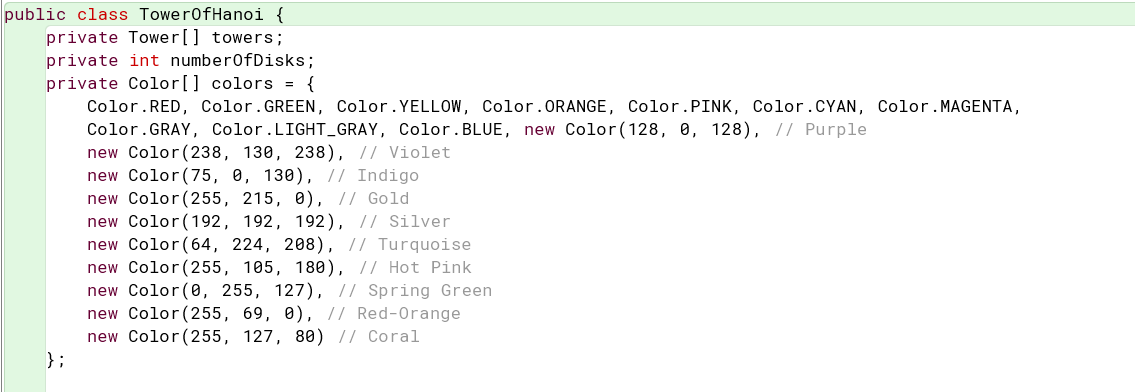
****

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

shapes. Expliquen.

En nuestro caso no usamos en ningún momento extends(para heredar clases),la única forma que hicimos de relacionar las clases fue en instancias (crear objetos a partir de las clases principales), es decir, usamos Rectangle para crear Tower y usamos Tower para usar TowerOfHanoi. Implícitamente está usando un principio SOLID (Liskov’s Substitution Principle) que se basa que cualquier clase que sea hijo de una clase padre puede ser reemplazada en vez de su padre ( en este caso, se instanció para reutilizar lo de la clase padre, añadiendo otros métodos, pero sin tener un comportamiento inesperado del programa).





## H. De python a java video

**[En lab01.doc]**

En este punto vamos usar un video para la transición de python y java y evaluarlo en la encuesta preparada con ese objetivo. Uno del equipo diligencia la encuesta. ¿En cuál punto de la encuesta discutieron más? Expliquen.

Discutimos en gran mayoría cuando preguntaron ¿qué aprendimos? No recordamos la pregunta textual, pero es más o menos así. Nos llevamos del video varios conocimientos de la transición de Python a Java, desde los conceptos básicos de que es clase, objeto, atributos, métodos; hasta codificar un código de binarySearch en Python a Java, este reto fue gratificante porque nos ayudó a recordar esa función importante en la programación para ordenar secuencias. Encontramos puntos en común entre ambos lenguajes de programación, como Random(), en los dos lenguajes si o si se tienen que importar, claramente de distinta manera, y digamos también es curioso que en Java esté la API de String.

La pregunta nos puso a pensar para responder adecuadamente porque de tanta información nueva que no sabíamos, nos generó más expectativas de lo que va a ser POOB, y en general Java. También la última parte que propusieron el ejercicio de la clase Estudiante fue interesante porque en AYED el último tercio trabajamos así con clases pero en Python, entonces saber cómo se codifica en Java es innovador.

## BONO. Nuevos requisitos funcionales. TowerOfHanoi.

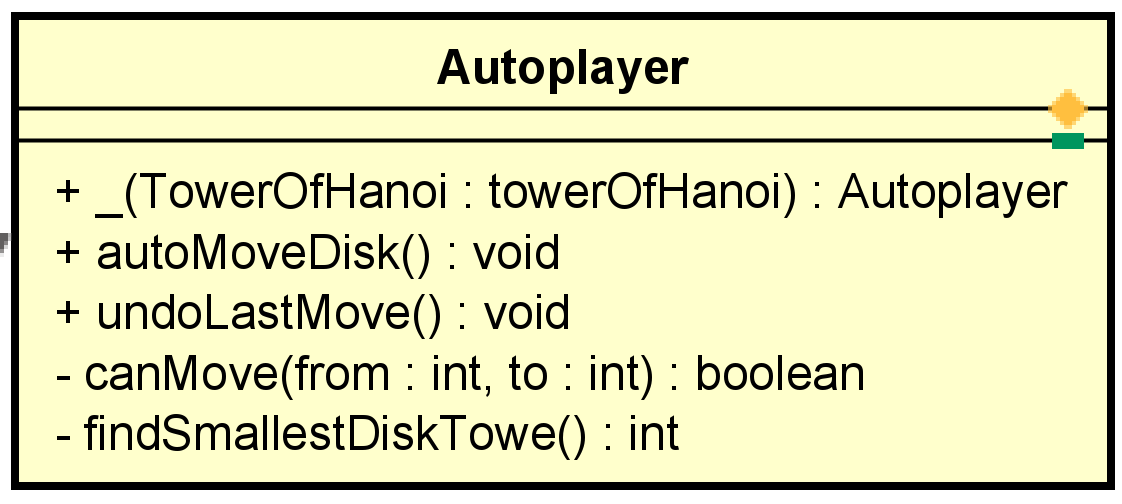
**[En lab01.doc. TowerOfHanoi.java]**

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

**Nuevos requisitos funcionales**

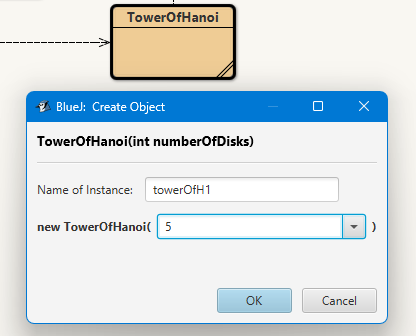
* Mover un disco de una torre a la otra (la máquina decide qué movimiento hacer). Explique la estrategia.
* Deshacer el último movimiento
  1. Diseñen, es decir, definan los métodos que debe ofrecer.

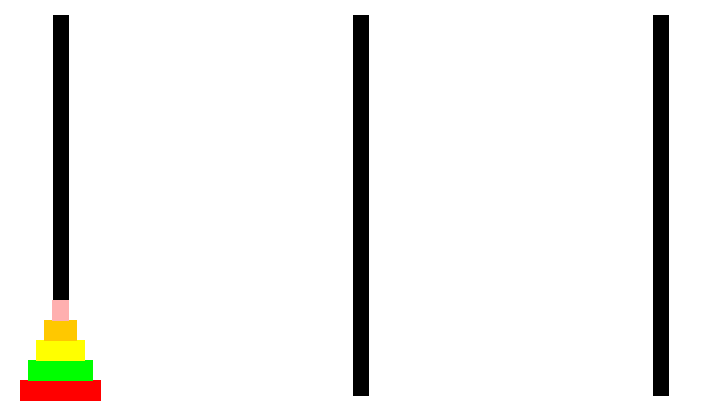
Como tal, no se implementaron los nuevos métodos sobre la clase TowerOfHanoi, lo que se hizo fue crear una nueva clase llamada Autoplayer en donde se agregaran los nuevos métodos, esto con el fin de ceñirnos al principio de single responsible y mantener un desacoplamiento de la mini aplicación.



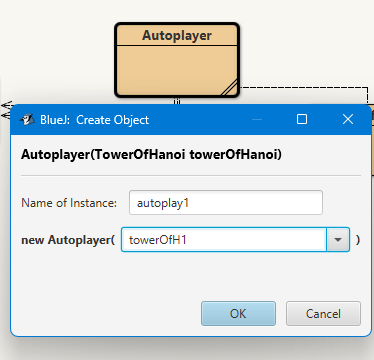
* 1. Implementen los nuevos métodos . Al final de cada método realicen una prueba indicando su propósito. Capturen las pantallas relevantes.

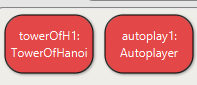
Crearemos una instancia de la torre Hanoi





Ahora crearemos una nueva instancia de la clase Autoplayer en donde le indicaremos que jugará con la torre de Hanoi previamente creada.



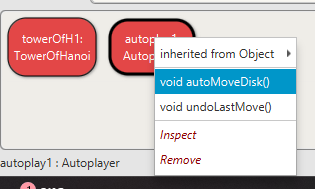


El primer método que se creó es autoMove.

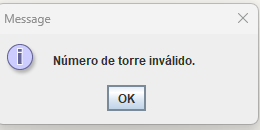


Y estos son los resultados en la interfaz gráfica.

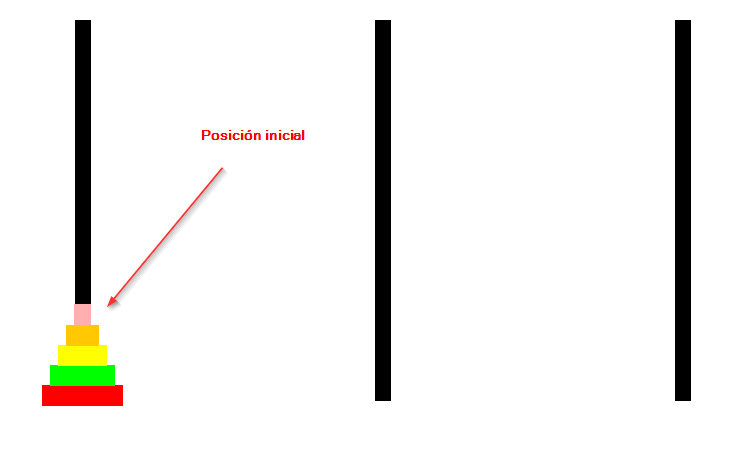
1. Hacemos un movimiento automático determinado por la máquina

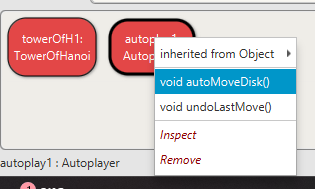


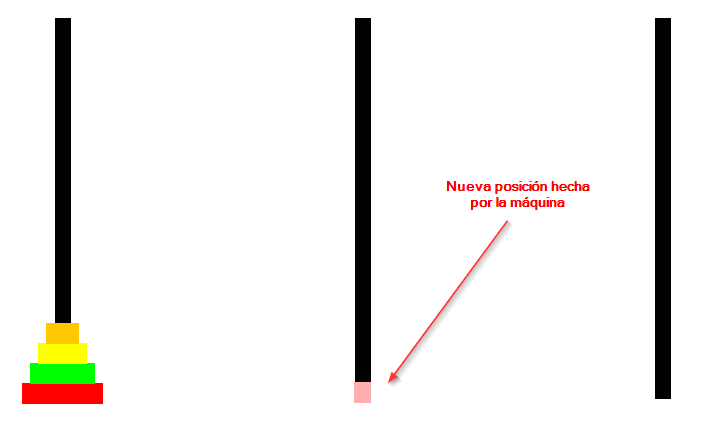
La respuesta es, de acuerdo con la validación, es número de torre no válido, lo que indica que la máquina eligió un número de torre no válido.



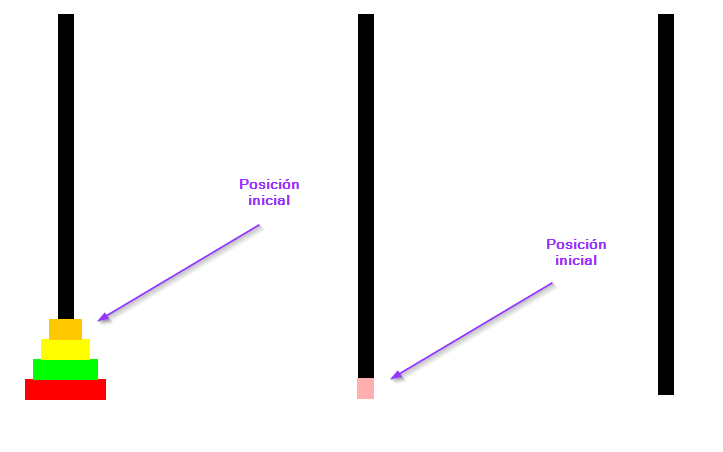
1. Volvemos a ejecutar el comando



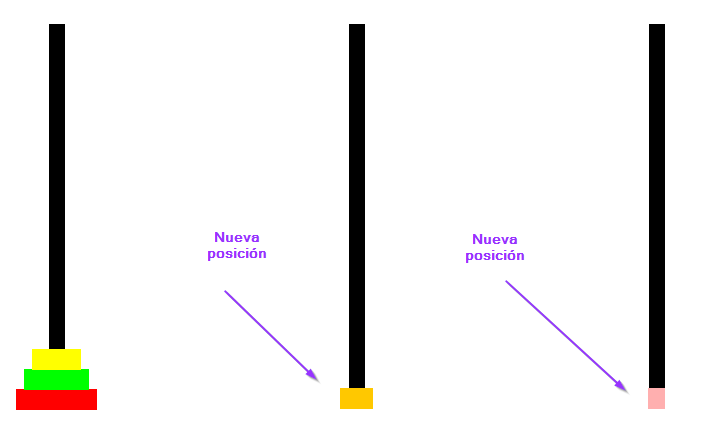




1. Volvemos a ejecutar el comando dos veces



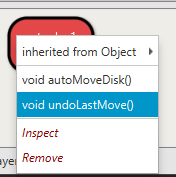
1. Disco que ha sido movido por la máquina.



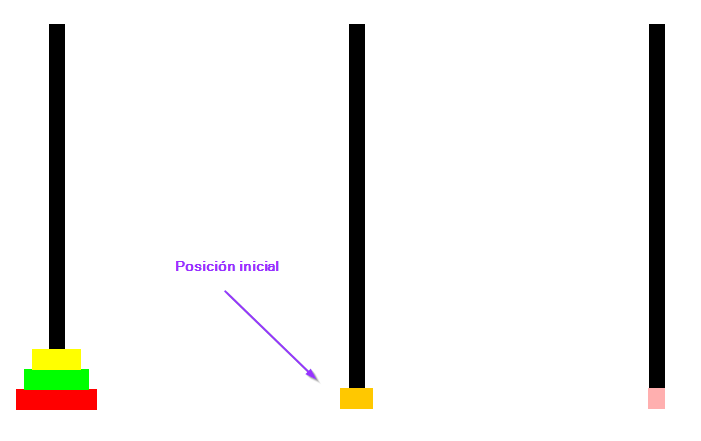
La explicación es la siguiente: para mover un disco de una torre a otra y hacer que la máquina decida qué movimiento realizar, podemos usar una estrategia clásica que sigue las reglas del juego de Torres de Hanoi.

* **Mover el disco más pequeño**: La primera cosa que la máquina debe hacer es mover el disco más pequeño disponible. Este disco generalmente se mueve en sentido horario entre las torres (de la primera torre a la segunda, de la segunda a la tercera, y de la tercera de vuelta a la primera).
* **Mover otro disco**: Luego, la máquina debe mover el disco más grande disponible que pueda moverse sin romper las reglas (no se puede poner un disco más grande sobre uno más pequeño). Este movimiento será entre las dos torres restantes que no participaron en el movimiento del disco más pequeño.
* **Repetir el proceso**: La máquina repetirá estos pasos, alternando entre mover el disco más pequeño y mover otro disco, siempre respetando las reglas del juego. Con esta estrategia, la máquina resolverá el juego moviendo los discos de la primera torre a la tercera torre.

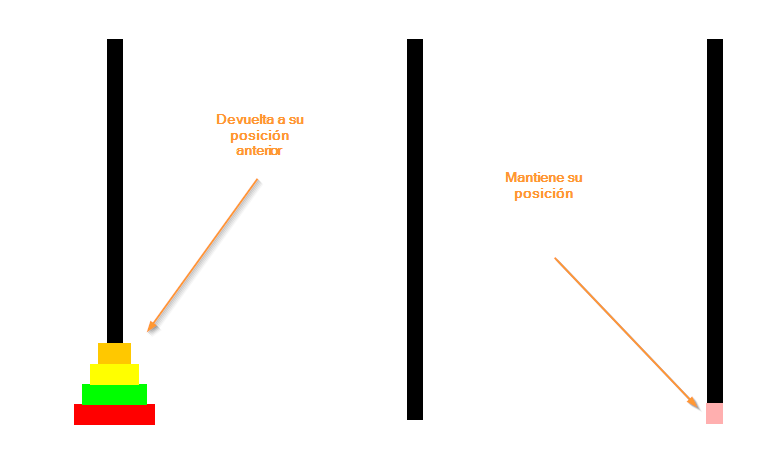
El otro método público que se implementó fue el de undoLastMove



Al invocarlo con



Da como resultado:



**RETROSPECTIVAS**

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

El tiempo total invertido fue más de 24 horas, sumando todas las horas trabajadas por distintos días, siempre trabajamos en Pair Programming.

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

Logramos cumplir con todo lo establecido en el laboratorio y con el bono, porque gracias al trabajo en equipo y la investigación se cumplieron los objetivos. También la persistencia fue vital para el desarrollo de este laboratorio, se presentaron dificultades, pero se resolvieron.

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

Pair Programming consideramos que fue la más útil porque mientras alguno programaba, el otro podía dar más ideas y encontrar posibles fallas al código, mientras que el programaba no se daba cuenta; luego se intercambiaron los roles, y pasó lo mismo. De este intercambio logramos coordinación y efectividad en los puntos.

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

En general el mayor logro fue terminar el laboratorio a tiempo, y no estar corriendo a última hora, ya que, después de estar trabajando al laboratorio varios días, estábamos en un punto de que no íbamos a terminar, a causa de errores mínimos que no lográbamos solucionar.

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

El mayor problema, donde le dedicamos más tiempo fue en el método push de la Torre de Hanoi, y en general Tower, porque al mover los discos de 1 a 2 o de 2 a 3 funcionaban, pero de 3 a 1 o de 2 a 1, no ponía el disco en la cima; y en el Tower porque al hacer push ocurría lo mismo.

Se solucionó haciendo distintas pruebas unitarias para ver desde qué punto estábamos haciendo algo mal.

En conclusión, nos dimos cuenta que solo era cuestión de una línea de código, en donde estaba calculando mal la posición en Y, se arregló haciendo pruebas, hasta lograr con lo establecido.

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

Insistir, persistir y nunca desistir (pero en trabajo en equipo). Esta frase es fundamental en todos los laboratorios porque al tener eso en cuenta cumplimos con los objetivos definidos.

Nos comprometemos a investigar más antes de aplicar en código los problemas. Nos pasó que creíamos entender cómo funcionaba la Torre de Hanoi, lo aplicamos en código, y nos dimos cuenta que había elementos extraños. Recurrimos a buscar el juego online y entender bien las reglas establecidas del juego.

## REFERENCIAS

* <https://www.geogebra.org/m/NqyWJVra> //Juego de Torre de Hanoi
* <https://www.geeksforgeeks.org/c-program-for-tower-of-hanoi/>
* IA gen

* <https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>

* <https://docs.oracle.com/javase/8/docs/api/>