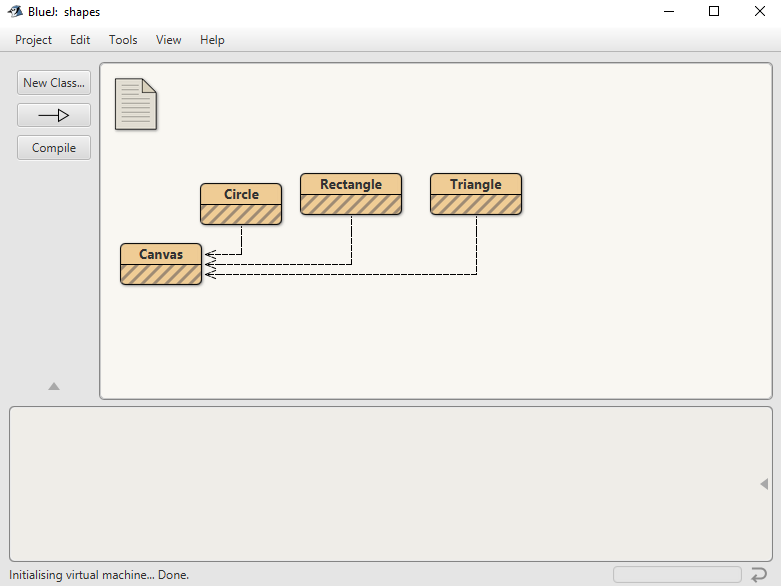
**Laboratorio 01. POOB-02**

Jefer Alexis González Romero  
Angel Nicolas Cuervo Naranjo

**SHAPES**

**Conociendo el proyecto shapes**

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

**a)** Clases:

* Circle
* Rectangle
* Triangle
* Canvas

**b)** *Circle*, *Rectangle* y *Triangle* se relacionan con *Canvas*, estas hacen uso de la última para pintarse y visualizarse.

**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? (b) ¿Qué atributos tiene la clase Rectangle? (c) ¿Cuántos métodos ofrece la clase Rectangle? (d) ¿Cuáles métodos ofrece la clase Rectangle para que la figura cambie (incluya sólo el nombre)?

**a)** *Canvas, Circle, Rectangle* and *Triangle.*

**b)** En la documentación no se menciona ningún atributo de *Rectangle*.

**c)** Tiene 12 métodos.

**d)** *changeSize* y *changeColor*

**4.** En el código de cada clase está el detalle de la implementación. Revisen el código de la clase Rectangle. Con respecto a los atributos: (a) ¿Cuántos atributos realmente tiene? (b) ¿Cuáles atributos describen la forma de la figura? Con respecto a los métodos: (c) ¿Cuántos métodos tiene en total? (d) ¿Quiénes usan los métodos privados?

**a)** Tiene 7 atributos.

**b)** *edges, height, width* and *color*.

**c)** Tiene 15 métodos.

**d)** Los métodos privados son *erase* and *draw*, los cuales son usados por *makeVisisble, makeInvisible, moveHorizontal, moveVertical, slowMoveHorizontal, slowMoveVertical, changeSize, changeColor.*

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

**a)** No se ven los atributos y los métodos privados

**b)** Porque solo pueden ser accedidos por la clase que los posee.

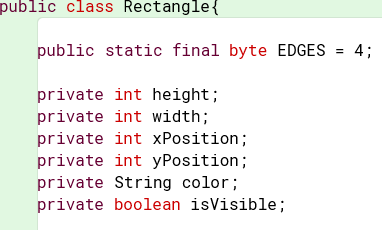
**6.** En el código de la clase Rectangle, revise el atributo EDGES (a) ¿Qué significa que sea public? (b) ¿Qué significa que sea static? (c) ¿Qué harían para que no se pueda cambiar su valor? ¿Por qué es necesario? (d) ¿De qué tipo de datos debería ser? ¿Por qué? Actualícenlo.

**a)** Si un atributo es *public* significa que tendremos acceso a él desde cualquier clase o instancia sin importar el paquete o procedencia de ésta.

**b)** Significa que todos los objetos en este caso de la clase *Rectangle* en el atributo *edges* tendrán el mismo valor, en ese caso la variable es única para todas las instancias de la clase.

**c)** Usaríamos f*inal* para que solo se le puede asignar su valor una vez y no se puede modificar.

**d)** Debería ser de tipo *byte,* ya que sabemos que no se va a modificar ese valor y va a permanecer en 4, el cual es un valor pequeño, así que para ahorrar memoria lo usaríamos en lugar de *Int* que requiere 3 *bytes* más de memoria.



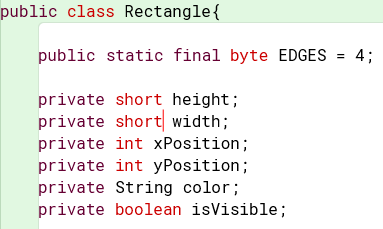
**7.** En el código de la clase Rectangle revisen el detalle del tipo del atributo height (a) ¿Qué se está indicándo al decir que es int?. (b) Si sabemos todos los rectángulos van a ser pequeños (alto y ancho menor 100), ¿de qué tipo deberían ser este atributo? (c) Si son grandes, pero no tanto (alto y ancho menor a 30000), ¿de qué tipo deberían ser este atributo? (d) ¿qué restricción adicional deberían tener estos atributos? Refactoricen el código considerando (c) y expliquen claramente sus respuestas.

**a)** Que se le pueden asignar números enteros de hasta 32 bits entre -2,147,483,648 y 2,147,483,647.

**b)** Como van a ser pequeños se puede usar el tipo *byte* que sirve para número enteros de hasta 8 bits entre -128 y 127

**c)** Se usaría *short* que son enteros de hasta 16 bits que son los que están entre -32.78 y 32.767

**d)** La restricción que deberían tener es que deben ser positivos.



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

Su propósito es demostrar algunas características de objetos y clases como su estructura.

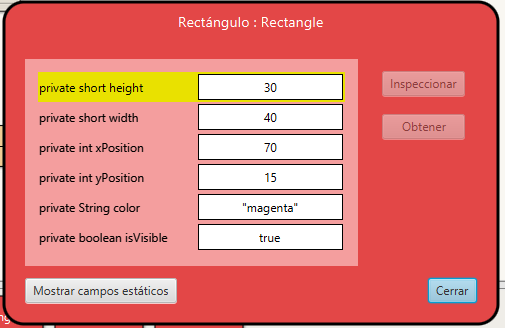
**Manipulando objetos. Usando un objeto.**

**1.** Creen un objeto de cada una de las clases que lo permitan . (a) ¿Cuántas clases hay? ¿Cuántos objetos crearon? (b) ¿Por qué?

**a)** Hay cuatro clases y de cada una creamos un objeto.

**b)** Porque de cada clase se puede crear un objeto, incluyendo al *canvas* que es la interfaz gráfica donde se pueden visualizar los otros objetos.

**2.** Inspeccionen el estado del objeto: Rectangle, ¿Cuáles son los valores de inicio de todos sus atributos? Capture la pantalla.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

**3.** Inspeccionen el comportamiento que ofrece el objeto :Rectangle. (a) Capturen la pantalla. (b) ¿Por qué no aparecen todos los que están en el código?

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Porque draw() y erase() son privados y solo los puede utilizar la clase.

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

**a)** Se necesitan 3 clases: Circle, Rectangle y Canvas

**b)** Se usan en total 7 objetos

**c)**

Forma

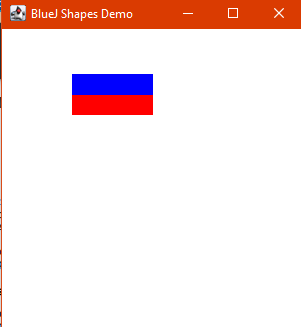
Descripción generada automáticamente

**d)**



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

1. Lean el código anterior. (a) ¿cuál es la figura resultante? (b) Píntenla.
2. Son dos rectángulos de las mismas dimensiones, uno azul que está arriba de uno rojo.



1. Habiliten la ventana de código en línea6 , escriban el código. Para cada punto señalado indiquen: (a) ¿cuántas variables existen? (b) ¿cuántos objetos existen? (c) ¿qué color tiene cada uno de ellos? (d) ¿cuántos objetos se ven? Al final, (e) Expliquen sus respuestas. (f) Capturen la pantalla.

**//1**

1. Variables: 3
2. Objetos: 0
3. No existen los objetos
4. Objetos visibles: 0
5. Hasta ahora se declararon las variables que son *yellow, blue y red.*

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**//2**

1. Variables: 3
2. Objetos: 2
3. Todos tienen el predeterminado que es magenta
4. Objetos visibles: 1
5. Se crean dos objetos blue y *yellow-red* que son el mismo objeto, así que este último solo cuenta por uno.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

**//3**

1. Variables: 3
2. Objetos: 2
3. Hay un objeto magenta y uno amarillo
4. Objetos visibles: 1
5. Se le cambia el color amarillo que tenía *yellow* por magenta.

Interfaz de usuario gráfica

Descripción generada automáticamente

**//4**

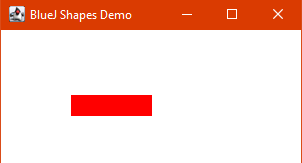
1. Variables: 3
2. Objetos: 2
3. Hay un objeto azul y uno amarillo
4. Objetos visibles: 1
5. A azul se le vuelve asignar el objeto *rectangle,* así que no se crea un nuevo objeto y se le cambia el color magenta por el azul.

Interfaz de usuario gráfica

Descripción generada automáticamente

**//5**

1. Variables: 3
2. Objetos: 2
3. Hay un objeto azul y uno rojo
4. Objetos visibles: 1
5. Se le cambia al objeto red-yellow el color amarillo por rojo y se vuelve visible o través, lo cual esto último no genera ningún cambio.



**//6**

1. Variables: 3
2. Objetos: 2
3. Hay un objeto azul y uno rojo
4. Objetos visibles: 2
5. Se vuelve visible *blue.*

Interfaz de usuario gráfica

Descripción generada automáticamente

1. Compare figura pintada en 1. con la figura capturada en 2. , (a) ¿son iguales? (b) ¿por qué?
2. Sí, ya que al analizar el código en el primer punto nos dimos cuenta que *red y yellow* hacen referencia al mismo objeto, por lo tanto lo que le suceda al objeto *red* le sucederá al objeto *yellow* y esto mismo se vio reflejado en el punto dos.

**Extendiendo una clase. Rectangle.**

**1.** Desarrollen en Rectangle el método perímetro(). ¡Pruébenlo! Capturen una pantalla.

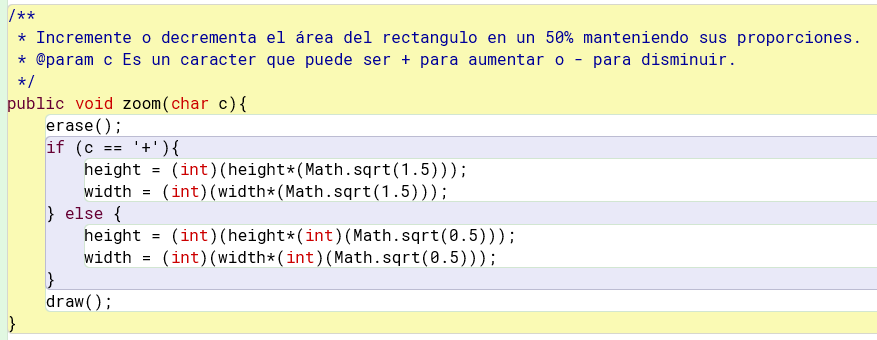
Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamente

**2.** Desarrollen en Rectangle el método zoom(c:char) (c: + o - , incrementa o decrementa su área un 50% manteniendo las proporciones) . ¡Pruébenlo! Capturen dos pantallas.



Forma

Descripción generada automáticamente

Imagen que contiene Forma

Descripción generada automáticamente

**3.** Desarrollen en Rectangle el método blink(times:int) (si times>0, parpadea el número de veces indicado) . ¡Pruébenlo! Capturen dos pantallas.

Texto

Descripción generada automáticamente con confianza media

**4**. Propongan un nuevo método para esta clase.

Forma

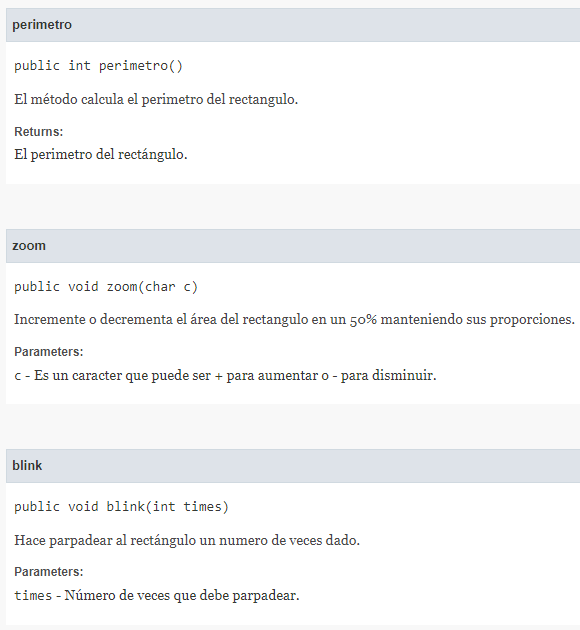
Descripción generada automáticamente con confianza mediaInterfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Imagen que contiene Rectángulo

Descripción generada automáticamente

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



**Codificando una nueva clase. Binary**

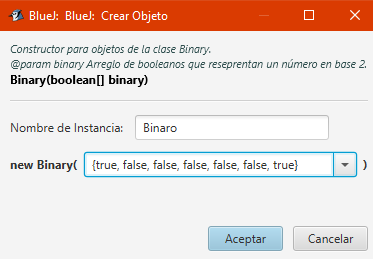
**1.** Revisen el diseño y clasifiquen los métodos en: constructores, analizadores y modificadores.

* Constructores: \_(binary), \_(decimal)
* Analizadores: getDecimal(), getBinary()
* Modificadores: next(), change(decimal), change(), makeVisile(), makeInvisible(), moveTo(x, y), changeColor()

**2.** Desarrollen la clase Binary considerando los 4 mini-ciclos. Al final de cada mini-ciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes.

* **Primer Ciclo**

Se ingresa un número en binario *Binary(boolean[] binary)* representándolo como un arreglo de binarios y se obtiene el valor decimal de este *getDecimal()*.

 Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente

Se ingresa el numero decimal *Binary(int decimal)* y se con obtiene en notación binaria getBinary().

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica

Descripción generada automáticamente

* **Segundo Ciclo**

Se obtiene el número siguiente al actual *next().*

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Sitio web

Descripción generada automáticamente con confianza baja

Se cambia el número por uno nuevo *change(int newDecimal).*

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

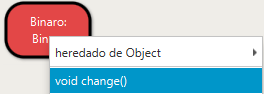
Descripción generada automáticamente

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

**Se cambia el número por uno aleatorio entre 1 y 100 *change().*

*Escala de tiempo

Descripción generada automáticamente con confianza bajaImagen que contiene Gráfico

Descripción generada automáticamente*

* **Tercer Ciclo**

Se hace visible el número en binario representado por círculos y rectángulos *makeVisible()*, y se vuelve a ocultar de la pantalla con *makeInvisible().*



Interfaz de usuario gráfica

Descripción generada automáticamente con confianza mediaForma

Descripción generada automáticamente con confianza baja

* **Cuarto Ciclo**

Mueve la representación del número a la posición dada *moveTo(int x, int y).*

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente**

**Forma

Descripción generada automáticamente**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamenteCambia el color de los rectángulos y círculos que representan al número en binario *changeColor(String newColor).*

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

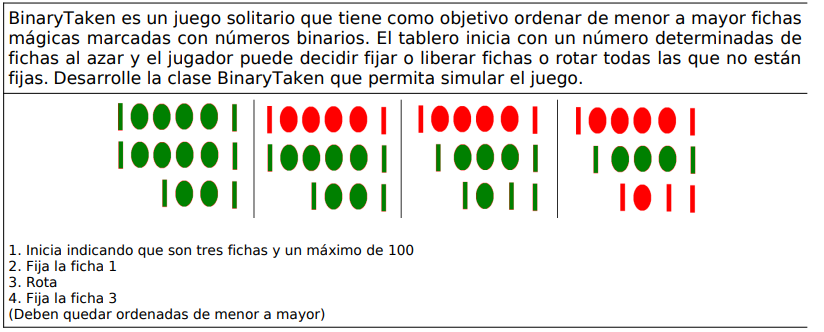
Descripción generada automáticamente

Forma

Descripción generada automáticamente

**Diseñando y codificando una nueva clase. BinaryTaken**

BinaryTaken es un juego solitario que tiene como objetivo ordenar de menor a mayor las fichas mágicas marcadas con números binarios. El tablero inicia con un número determinadas de fichas al azar y el jugador puede decidir fijar o liberar fichas o rotar todas las que no están fijas. Desarrolle la clase BinaryTaken que permita simular el juego.



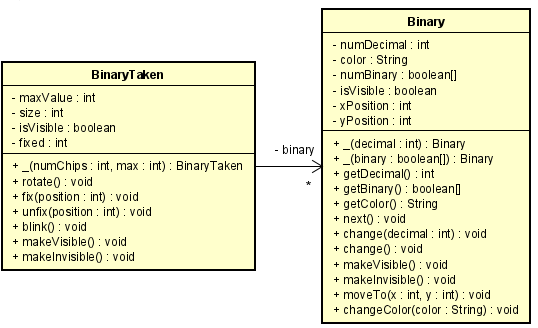
**Requisitos funcionales**

* Permitir iniciar el juego indicando el tamaño y el máximo valor de las fichas. El valor de las fichas inicia al azar.
* Cambiar el modo de una ficha: fija o no fija.
* Modificar al azar el valor de todas las fichas no fijas

**Requisitos de interfaz**

* Las fichas deben estar en fila
* Las fichas identifican por su posición[1..]
* El modo en que está la ficha (fija, no fija) debe indicarse con un color específico.
* En caso de que no sea posible realizar una de las acciones de usuario, el juego debe parpadear.
* Se debe presentar un mensaje de felicitación cuando el jugador gana. Use JoptionPane.

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



* El constructor de BinaryTaken.
* Un método para rotar las fichas que no estén fijas *rotate().*
* Para fijar *fix(int position)* y para desfijar *unfix(int position).*
* Un método para que parpadee el juego.
* Hacer el juego visible *makeVisible()* e invisible *makeInvisible().*

**2.** Planifiquen la construcción definiendo algunos miniciclos. Recuerden que al final de cada miniciclo deben poder probar.

**Miniciclos**

1. \_(numChips, max)
2. makeVisible()

makeInvisible()

blink()

1. fix(position)

unfix(position)

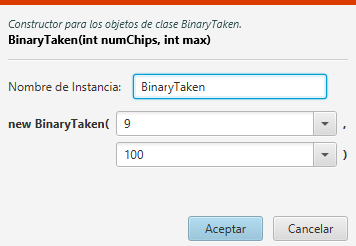
rotate()

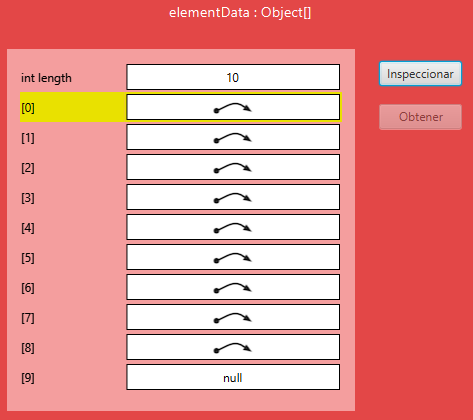
**3.** Implementen la clase . Al final de cada miniciclo realicen una prueba de aceptación. Capturen las pantallas relevantes.

**Primer ciclo**

Para iniciar el juego se debe indicar el tamaño y el máximo valor de las fichas,

 se generarán las fichas aleatoriamente con las especificaciones dadas.

****

 Imagen que contiene Tabla

Descripción generada automáticamente

**Segundo Ciclo**

Para visualizar el juego se ubican las fichas en filas *makeVisible(),* y para borrarlo de la pantalla *makeInvisible().* Para indicar al usuario que no es posible realizar la acción que quiere se hacen parpadear las fichas *blinck().*

****

**Interfaz de usuario gráfica

Descripción generada automáticamente con confianza mediaGráfico de burbujas

Descripción generada automáticamente**

**Tercer Ciclo**

**Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

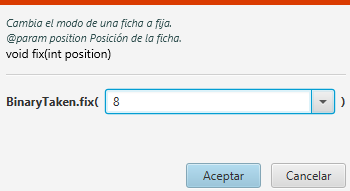
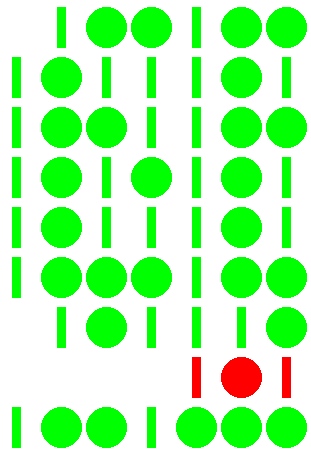
Descripción generada automáticamente**Se puede cambiar el modo de las fichas a fijas *fix(position)* dando la posición de estas, las verdes indican que no están fijas y las rojas que si lo están.

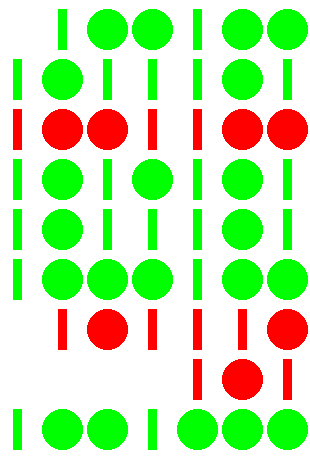
Imagen que contiene Patrón de fondo

Descripción generada automáticamenteDe igual forma se pueden pasar fichas a no fijas (verdes) dando su posición *unfix(position).*

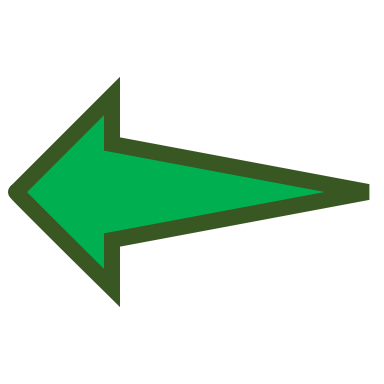
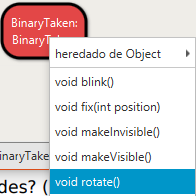
Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

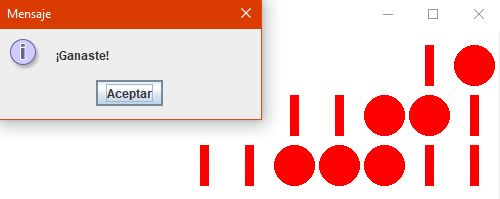
Descripción generada automáticamente

Gráfico

Descripción generada automáticamente con confianza mediaLos usuarios pueden rotar las fichas *rotate()*, se modifica al azar el valor de las fichas que no están fijas.



Si el usuario ordena todas las fichas de menor a mayor y las pone fijas le aparecerá un mensaje de felicitación.



**4.** Indiquen las extensiones necesarias para reutilizar el paquete shapes y la clase Binary. Explique.

En *Binary* se debió crear un nuevo método llamado *getColor()* el cuál nos daba el color de la ficha, para el uso de *Binary* se creo un arreglo de esta clase, en el cual para obtener un objeto en especifico se hizo con *get(position)*. De la misma forma se hizo para utilizar *shapes* en *Binary* se crearon arreglos de *Circles y Rectangles.*

Así quedó Binary dentro de la interfaz de *BlueJ*

*Diagrama

Descripción generada automáticamente*

La clase *Binary* tiene extensiones a las clases *Rectangle* y *Circle,* esto se debe a que para representar los números en la pantalla necesitamos de círculos y rectángulos.

**RETROSPECTIVA**

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

12 horas por cada persona.

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

El laboratorio está totalmente terminado.

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

Nos resultó más útil la programación a pares, gran parte de este laboratorio lo hicimos trabajando de esta manera ya que las ideas que nos surgen las complementamos, podemos evitar posibles errores futuros de mejor manera, etc.

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

Fue representar los números con los rectángulos y círculos.

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

Hacer que las fichas se ubicaran como queríamos y que al cambiarlas de posición quedaran donde se indicara. Para resolverlo buscamos donde se encontraba el error que hacía que se movieran de forma errónea.

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

La comunicación fue clave para la elaboración de este laboratorio. Nos comprometemos a dedicarle más tiempo al laboratorio, elaborándolo con más anticipación y a mejorar el nivel de conocimiento del lenguaje (JAVA).