UNIVERSIDAD ESCUELA COLOMBIANA DE INGENIERÍA
JULIO GARAVITO

LABORATORIO 1

PROGRAMACIÓN ORIENTADA A OBJETOS

NOMBRES:

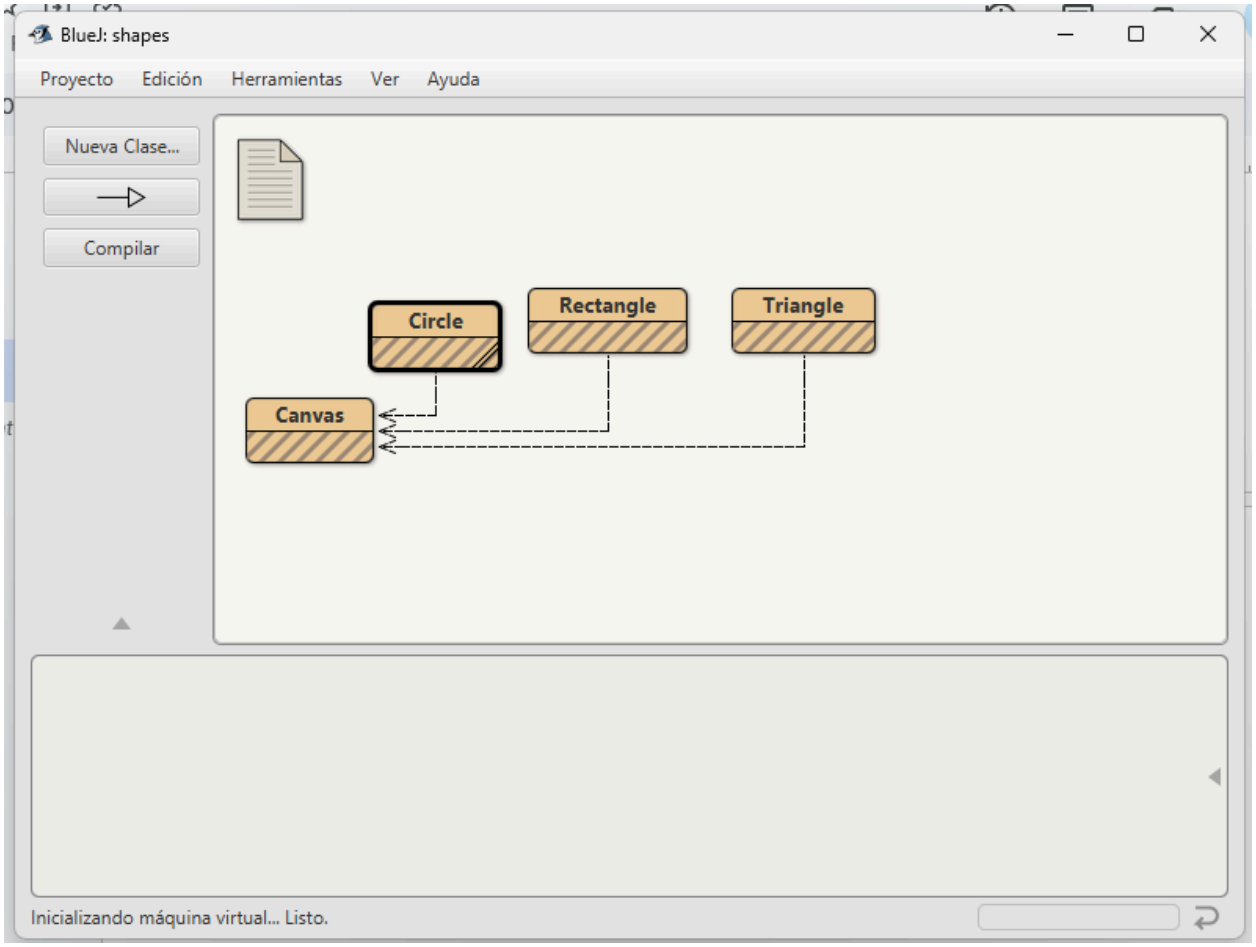
CHRISTIAN ALFONSO ROMERO MARTINEZ
ANDERSON FABIAN GARCIA NIETO

PROFESORA:
MARIA IRMA DIAZ ROZO

30/01/24

PARTE A

1. El proyecto “shapes” es una versión modificada de un recurso ofrecido por BlueJ. Para trabajar con él, bajen shapes.zip y ábranlo en BlueJ2. Capturen la pantalla.



2. El diagrama de clases permite visualizar las clases de un artefacto software y las relaciones entre ellas. Considerando el diagrama de clases de “shapes”

(a) ¿Qué clases ofrece?

Canvas
Circle
Rectangle
Triangle

(b) ¿Qué relaciones existen entre ellas?

La relación entre las clases se resume a la relación de cada una de ellas a Canvas, pues en esta es donde se representan.

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?

El paquete shapes contiene las siguientes clases:

Canvas
Circle
Rectangle
Triangle

(b) ¿Qué atributos tiene la clase Triangle?

De acuerdo al constructor de la clase Triangle, se observan varios atributos, pero en este caso como se refiere a la documentación y solo se muestran los atributos que estén en público, por eso solo aparece el atributo VÉRTICES.

(c) ¿Cuántos métodos ofrece la clase Triangle?

12 metodos

Change the color.
Change the size to the new size
Make this triangle invisible.
Make this triangle visible.
Move the triangle a few pixels down.
Move the triangle horizontally.
Move the triangle a few pixels to the left.
Move the triangle a few pixels to the right.
Move the triangle a few pixels up.
Move the triangle vertically.
Slowly move the triangle horizontally.
Slowly move the triangle vertically.

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

No se pueden conocer los atributos que determinan el tamaño del triángulo, debido a que en la vista de la documentación solo aparece el atributo VÉRTICES, siendo un caso diferente desde la vista del código.

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

changeSize

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

(a) ¿Cuántos atributos realmente tiene?

Cuenta con 7 atributos:

VERTICES
height
width
xPosition
yPosition
color
isVisible

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

Todos pueden hacer uso de los atributos públicos, refiriéndose a que tanto los métodos dentro de la clase Triangle o los métodos que están fuera de ella

(c) ¿Cuántos métodos tiene en total?

En total hay 15 métodos, siendo estos:

Triangle()
makeVisible()
makeInvisible()
moveRight()
moveLeft()
moveUp()
moveDown()
moveHorizontal()
moveVertical()
slowMoveHorizontal()
slowMoveVertical()
changeSize()
changeColor()
draw()
erase()

(d) ¿Quiénes usan los métodos privados?

Los métodos privados son usados por por otros métodos que pertenezcan a la clase Triangle:

draw()es usado por:
makeVisible

moveHorizontal
moveVertical
slowMoveHorizontal
slowMoveVertical
changeSize
changeColor

erase() es usado por:
makeInvisible
moveHorizontal
moveVertical
changeSize

5. Comparando la documentación con el código

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

En la documentación no se presentan los métodos privados.

(b) ¿Por qué debe ser así?

La documentación tiene como propósito guiar a los usuarios, y al colocar cosas que no se encuentran a su alcance puede llegar a generar confusiones.

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

(a) ¿Qué significa que sea public?

El que sea public significa que se puede acceder a ellos directamente,desde cualquier lugar como otra clase o desde la misma.

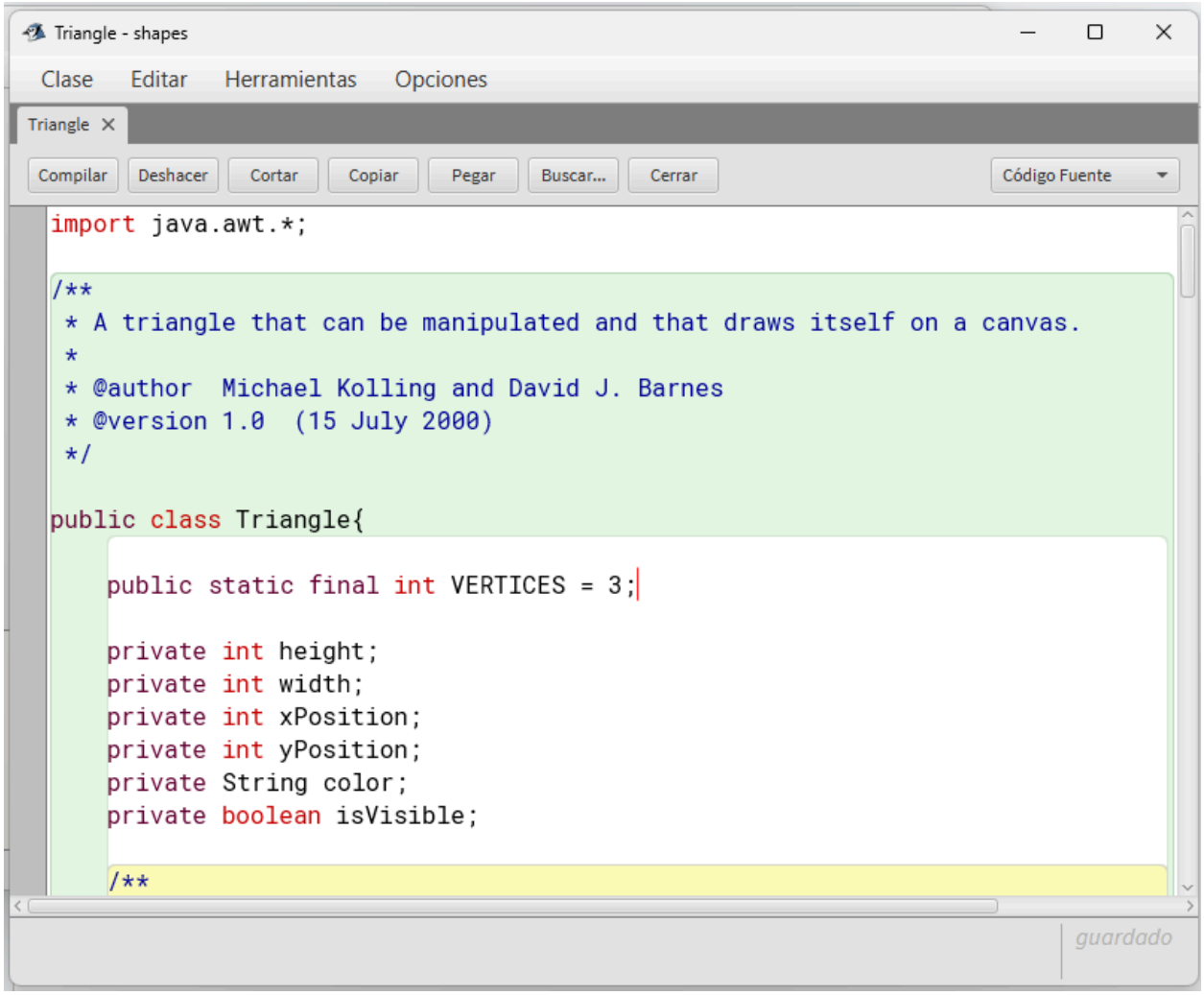
(b) ¿Qué significa que sea static?

Static significa que es un valor propio de la clase y nos puede dar una ventaja, y es que se puede acceder a ella directamente.
La modificación del atributo que contenga static, provoca que todos los objetos que dependan de este, se modifiquen

(c) ¿Qué significa que fuera final?

El que sea final significa que no se puede cambiar una vez se inicia
¿Debe serlo?
Si debe ser final, pues los vértices de un triángulo si o si deben ser 3 y cualquier modificación de esto haría la clase ilógica.

(d) Actualícenlo



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

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

Al decir int se indica que los valores que puede tomar son enteros sin signo

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

El rango que se tiene es de -2^7 a 2^7-1, pero como se trata de longitudes no se incluyen los números negativos, ni tampoco el cero, quedando de 1 a 2^7-1, por lo que si la fórmula del área de un triángulo la comprendemos como $A = \frac{(b*h)}{2}$ su máximo será cuando b y h sean su máximo valor

$A = \frac{(2^7-1)*(2^7-1)}{2} = 8192\ unidades^2$

(c) Y ¿Si fuera long?

El rango que podemos tener es de -2^{63} a $2^{63}-1$ por que si tomamos el mayor número y reemplazamos en b y h por el máximo valor tenemos

$A = \frac{(2^{63})*(2^{63}-1)}{2} = 4.253529586511731\times\times10^{37}$

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

De acuerdo a la definición de int, este acepta números enteros, siendo positivos y negativos, por este motivo la restricción adicional que debería tener que la altura (height), debe ser mayor a cero

(e) Refactorizar el código considerando

Se añadió al código que la altura deba ser obligatoriamente un número positivo, de lo contrario no ocurrirá nada.

```
/**
 * Change the size to the new size
 * @param newHeight the new height in pixels. newHeight must be >=0.
 * @param newWidht the new width in pixels. newWidht must be >=0.
 */
public void changeSize(int newHeight, int newWidth) {
    int oldHeight = height;
    if (newHeight > 0){
        erase();
        height = newHeight;
        width = newWidth;
        draw();
    } else{
        System.out.println("El tamaño no se ha cambiado");
    }
}
```

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

El propósito shapes teniendo en cuenta su autor Michael Kolling es la introducción a BlueJ, se puede decir que también a la programación orientada a objetos y el modelado en java, pues menciona clases, objetos, entre otros.

PARTE B

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

(a) ¿Cuántas clases hay?

4 Clases.

(b) ¿Cuántos objetos crearon?

Se crearon los 4 objetos cada uno perteneciente a cada una de las clases.

(c) ¿Quién se crea de forma diferente?

El lienzo es quien se crea de una distinta manera

¿Por qué?

Se crea de forma distinta debido a su propósito pues es en este donde se ilustra, se dibuja y se integra AWT/Swing

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

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

Entre las clase de Circle, Rectangle, Triangle los constructores no tiene mayor diferencia además de los distintos parámetros de construcción, mientras que con el caso de Canvas se diferencia bastante, allí destacó que no se marcan los parámetros para su construcción, solo se llama al constructor sin parámetros y se crea en default

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

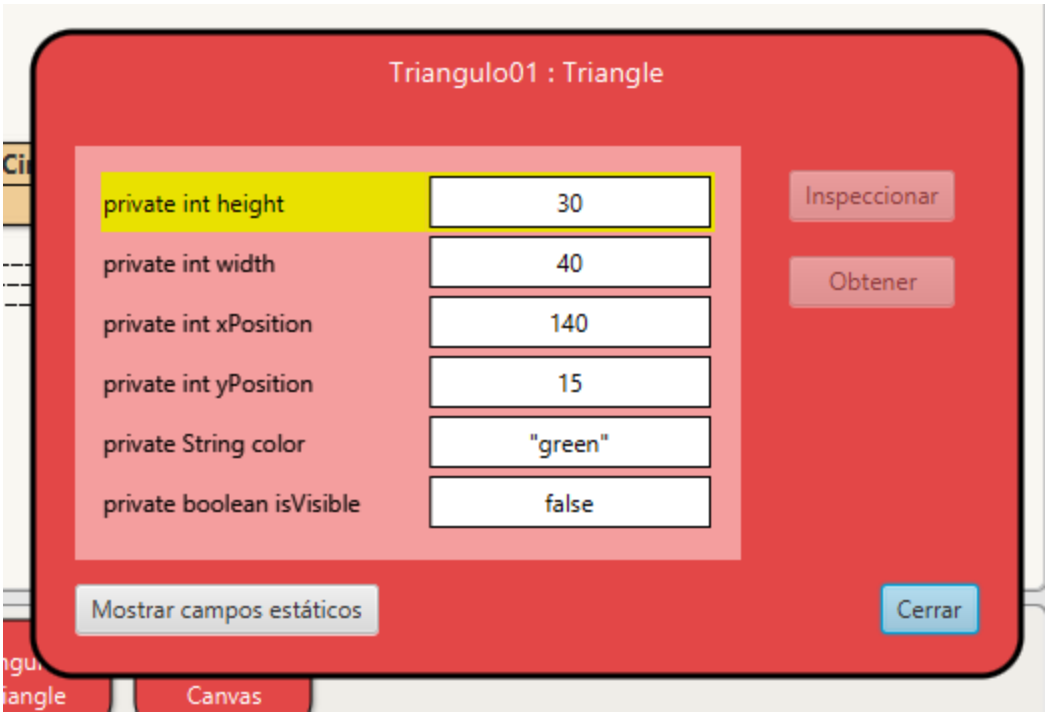
Crear el tablero donde se van a materializar las otras clases, es importante decir que en esta también se reciben los cambios hechos en las otras clases.

3. Inspeccionen el estado del objeto :Triangle,

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

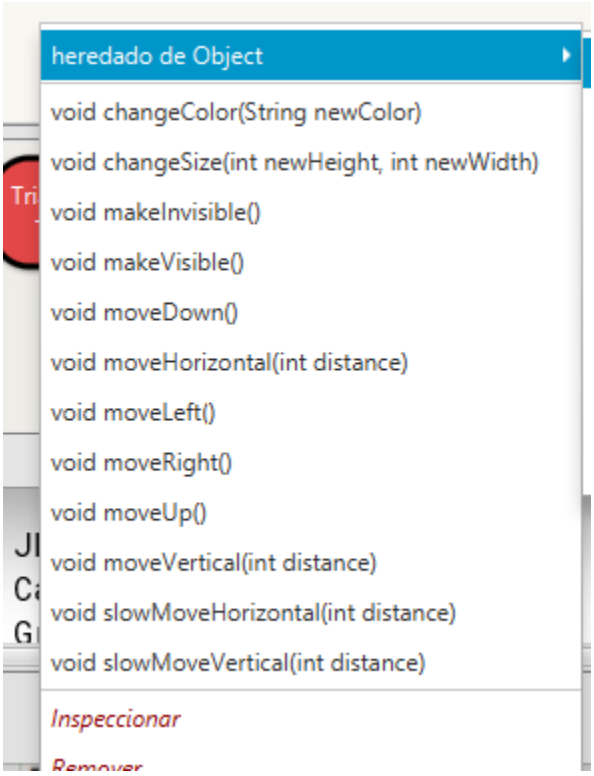
Altura: 30
Ancho: 40
Posicion en X: 140
Posicion en Y: 15
Color: Verde
Visibilidad: Falso

(b) Capturar la pantalla.



4. Inspeccionen el comportamiento que ofrece el objeto :Triangle.

(a) Capturen la pantalla.



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

En este punto aplica lo mismo que al momento de la documentación, solo se muestran los públicos, pero además tampoco se muestran los atributos estáticos, ya que estos hacen parte de la clase y no del objeto.

5. Construyan, con "shapes" sin escribir código, una propuesta de la imagen del logo de de su chatbot IA favorito.

(a) ¿Cuántas y cuáles clases se necesitan?

Se requirieron las 4 clases para cumplir con nuestro objetivo.

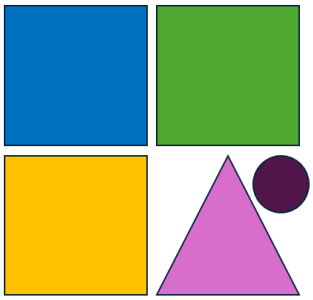
Las clases usadas son:

Canvas
Circle
Rectangle
Triangle

(b) ¿Cuántos objetos se usan en total?

6 objetos son usados

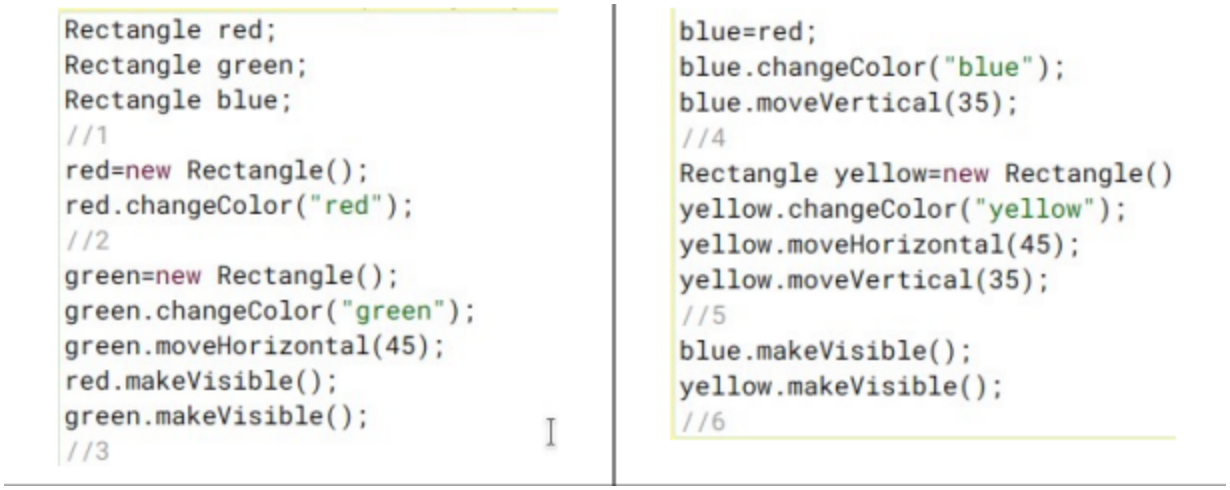
(c) Capturen la pantalla.



(d) Incluyan el logo original.

PARTE C

1. **Lean el código anterior. (a) ¿Cuál creen que es la figura resultante?**
   NOTA: Para este ejercicio se asumirá que en la primera linea de codigo de la parte 4, se cerró con ";", ya que de lo contrario de ahí para abajo ocurrirán errores



```
Rectangle red;
Rectangle green;
Rectangle blue;
//1
red=new Rectangle();
red.changeColor("red");
//2
green=new Rectangle();
green.changeColor("green");
green.moveHorizontal(45);
red.makeVisible();
green.makeVisible();
//3
blue=red;
blue.changeColor("blue");
blue.moveVertical(35);
//4
Rectangle yellow=new Rectangle()
yellow.changeColor("yellow");
yellow.moveHorizontal(45);
yellow.moveVertical(35);
//5
blue.makeVisible();
yellow.makeVisible();
//6
```

De acuerdo al código, proporcionado, se explicará detalladamente qué es lo que ocurre, y en el siguiente punto se dibujara usando el programa paint, que es lo que ocurre:

```
Rectangle red;
Rectangle green;
Rectangle blue;
```

Las 3 líneas de código cumplen con la función de declarar las variables que se van a usar, en este caso asignándoles la función de que reciban un objeto de la clase rectangulo.

Parte 1

```
//1
red=new Rectangle();
red.changeColor("red");
```

Le asigna a la variable rojo un nuevo objeto de la clase rectangulo, y a su vez a este rectangulo lo colorea de color rojo.

Parte 2

```
//2
green=new Rectangle();
green.changeColor("green");
green.moveHorizontal(45);
red.makeVisible();
green.makeVisible();
```

Para esta parte se le asigna a la variable verde, un nuevo objeto de la clase rectangulo, seguido de eso, se le cambia el color del rectángulo a verde, se mueve a la derecha 45 píxeles, se hace visible el rectángulo rojo y el rectángulo verde.

Parte 3

```
blue=red;
blue.changeColor("blue");
blue.moveVertical(35);
```

Para esta sección en la primera línea de código, cuando se iguala azul con rojo, se está haciendo que azul referencie al mismo objeto al que también referencia el rojo, inmediatamente de esto, se le cambia el color al rectángulo referenciado por la variable azul, haciendo que el rectángulo que antes era rojo, ahora sea azul y por último lo mueve 35 píxeles hacia abajo

Parte 4

```
//4
Rectangle yellow=new Rectangle()
yellow.changeColor("yellow");
yellow.moveHorizontal(45);
yellow.moveVertical(35);
```

Para esta parte, se declara una nueva variable e inmediatamente se le asigna un nuevo objeto a esta variable, siendo la variable amarilla, despues de esto, se le cambia el color al rectangulo, ahora siendo de color amarillo, se mueve 45 pixeles hacia la derecha y 35 pixeles hacia abajo.

Parte 5

```
//5
blue.makeVisible();
yellow.makeVisible();
```

Para esta última parte, se hacen visibles los rectangulos azul y amarillo

(b) Píntenla.

Se considera que la figura resultante es testa, debido a que el desplazamiento con los pixeles es un poco confuso

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

(a) ¿Cuántas variables existen?

`Rectangle red;`
`Rectangle green;`
`Rectangle blue;`
`//1`
`red=new Rectangle();`
`red.changeColor("red");`
`//2`
`green=new Rectangle();`
`green.changeColor("green");`
`green.moveHorizontal(45);`
`red.makeVisible();`
`green.makeVisible();`
`//3`
 	
`blue=red;`
`blue.changeColor("blue");`
`blue.moveVertical(35);`
`//4`
`Rectangle yellow=new Rectangle()`
`yellow.changeColor("yellow");`
`yellow.moveHorizontal(45);`
`yellow.moveVertical(35);`
`//5`
`blue.makeVisible();`
`yellow.makeVisible();`
`//6`
	En el código proporcionado existen 4 variables, aunque dos referencian el mismo objeto (Llegan al mismo punto de memoria), esto no significa que se elimine una.

(b) ¿Cuántos objetos existen? (No cuentan ni los objetos String ni el objeto Canvas)

Existen 3 objetos creados en el código proporcionado, estos son:

red = new Rectangle();

green = new Rectangle();

yellow = new Rectangle();

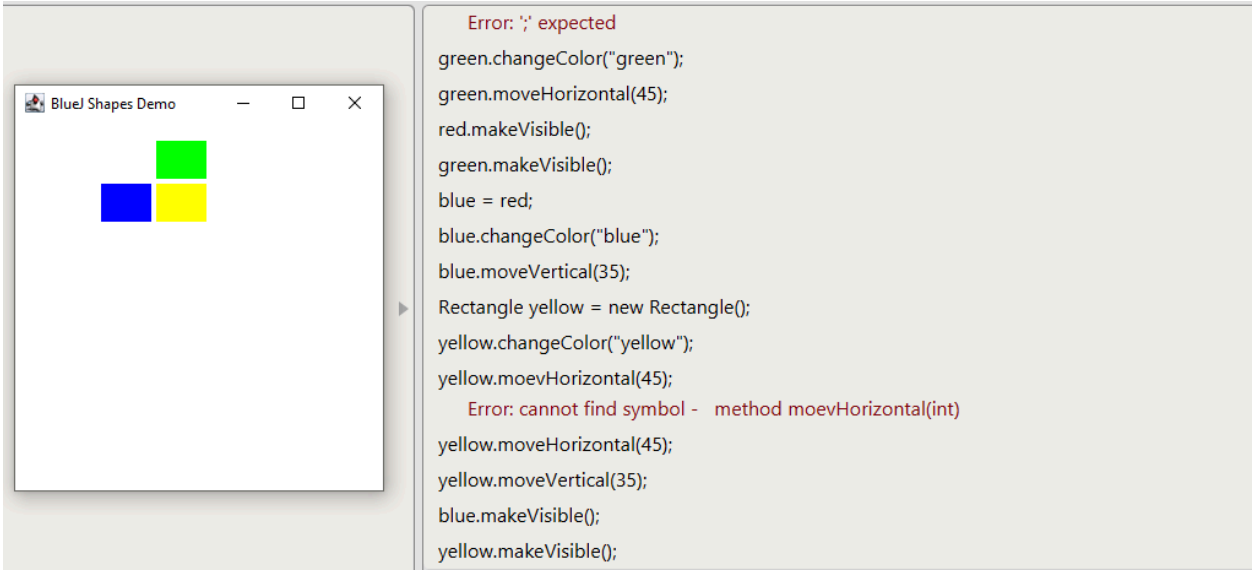
Son 3 y no 4, debido a que la variable azul y rojo referencian el mismo objeto, entonces cuentan como el mismo objeto, por lo que a diferencia de las 4 variables, en este caso hay 3 objetos.

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

Estan los 3 objetos, el objeto que esta asignado a la variable amarillo, es de color amarillo, lo mismo sucede con el que esta asignado a la variable verde, y el ultimo rectangulo, el que esta referenciado por dos variables, este es de color azul.

(d) ¿Cuántos objetos se ven?
Sin contar al lienzo (Canvas), se observan 3 objetos, siendo estos 3 rectángulos diferentes.

3. Al final,
(e) Expliquen sus respuestas.
Todo esto se logró gracias a las debidas declaraciones de las variables, y a sus respectivas referencias a los rectángulos, permitiendo cambiar los atributos de cada uno de estos, siendo por ejemplo la posición en la que se encuentra y el color que posee.
(f) Capturen la pantalla.



4. Compare la figura pintada en 1. con la figura capturada en 2.
(a) ¿Son iguales?
Tal vez no son iguales pero son semejantes, ya que el color y la distancia entre estos pueden variar, así sea muy poco.

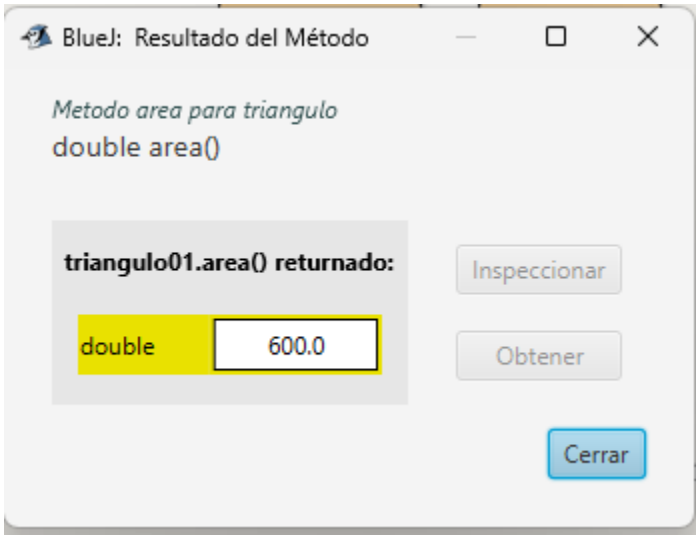
(b) ¿Por qué?
No son iguales porque influyen varios factores, siendo principalmente el tamaño de los rectángulos que se representaron por nuestra propia cuenta, por otro lado los colores que se eligieron para los rectángulos, y las distancias que podían variar unas respecto a otras, así se hayan desplazado la misma distancia en píxeles unos de otros. Ya que aún no se asimila la distancia en milímetros en cuanto a la longitud de un pixel.

Parte D.

1. Desarrollen en Triangle el método area().

```
/**
 * Metodo area para triangulo
 */
public double  area(){
    return (height*width/2);
}
```

¡Pruébenlo! Capturen una pantalla



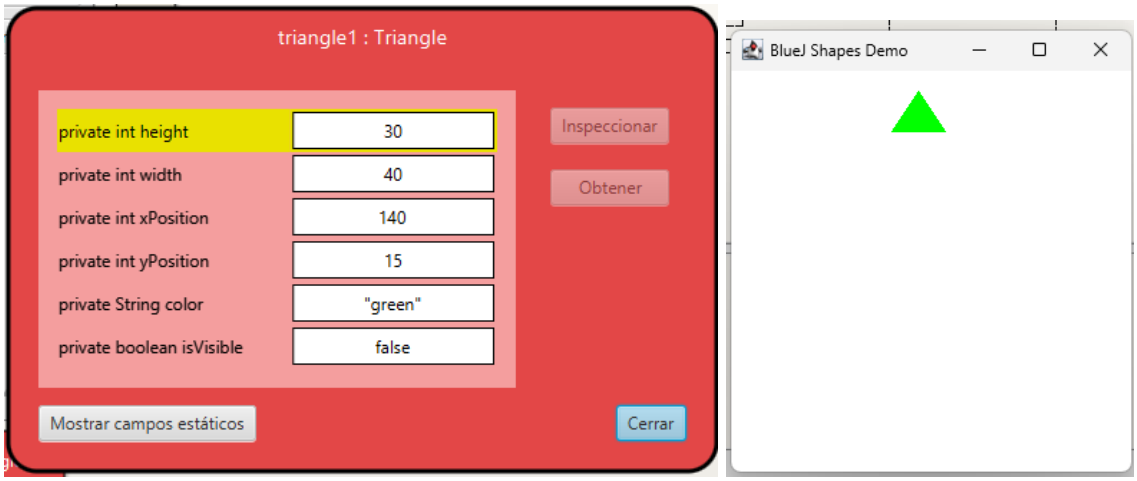
2. Desarrollen en Triangle el método equilateral()
(Transforma el triangulo en un triangulo equilatero de área equivalente) .

```
/**
 * Metodo modificar a triangulos equilateros
 */
public void equilateral(){
    double cArea = area();
    double lado = Math.sqrt((4 * cArea) / Math.sqrt(3));
    this.width = (int) lado;
    this.height = (int) (Math.sqrt(3) / 2 * lado);
}
```

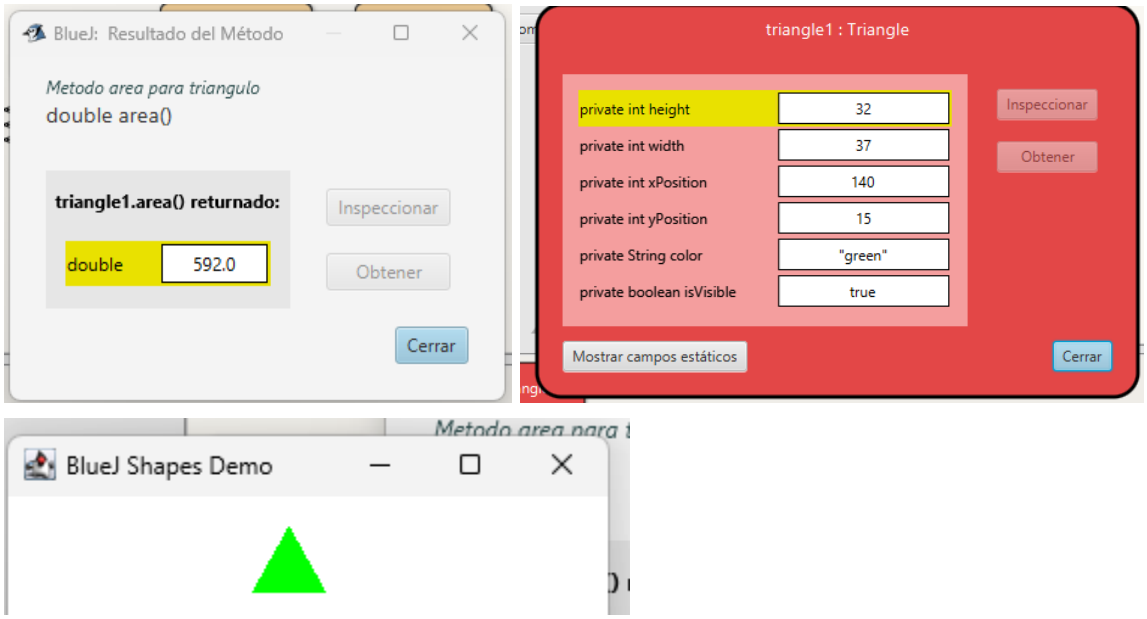
Se basó en la formula del area de un triángulo que es $Ae = \frac{\sqrt{3}}{4} lado^2$ y se despejó el lado tomando en cuenta que se sabe el área que debe tomar.

Su base es igual a un lado y luego se toma su altura con la fórmula $h = \frac{\sqrt{3}}{a} lado^2$

¡Pruébenlo! Capturen dos pantallas.



Valores iniciales del triángulo



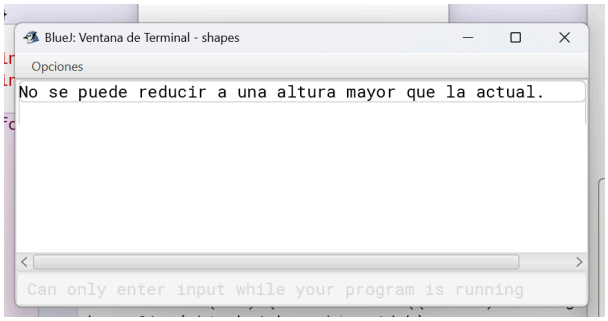
El valor del área es distinta al inicial debido al tipo de dato, al tomar solo datos enteros, se toma solo la parte entera.

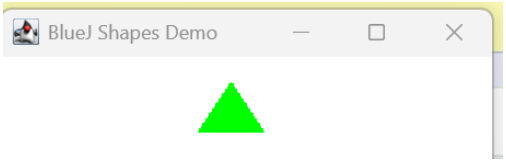
3. Desarrollen en Triangle el método shrink(times:int, height: int) (disminuye su tamaño times veces. Hasta llegar a una altura de height.)

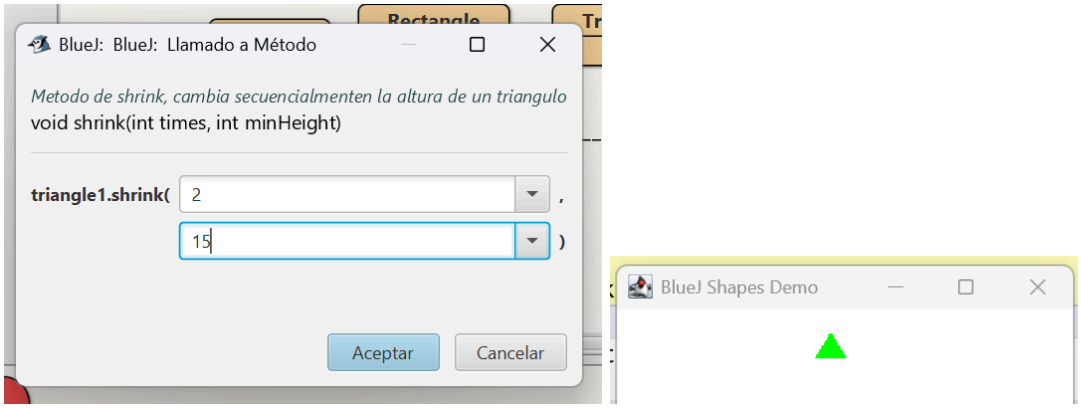
```java
/**
 * Metodo de shrink, cambia secuencialmenten la altura de un triangulo
 */
public void shrink(int times, int minHeight) {
    if (minHeight > this.height) {
        System.out.println("No se puede reducir a una altura mayor que la actual.");
        return;
    }

    int diferencia = this.height - minHeight;
    int decremento = diferencia / times;

    for (int i = 0; i < times; i++) {
        if (this.height - decremento >= minHeight) {
            this.height -= decremento;
            this.width = (int) (this.width * ((double) this.height / (this.height + decremento)));
            changeSize(this.height, this.width);
        }
        else {
            this.height = minHeight;
            this.width = (int) (this.width * ((double) minHeight / (this.height + decremento)));
            changeSize(this.height, this.width);
            break;
        }
    }
}
```

El desarrollo inició con una validación de objetivos, pues se comparó la altura nueva con la del triángulo, pues el objetivos es reducir y en caso que no se reduzca no debería ser parte del método, luego se definieron términos importantes para la estrategia de solución elegida, que fue tomar la diferencia entre las alturas y dividirlo por los times, para encontrar la reducción que se debe hacer en cada time, luego usando un for se validó que se pueda seguir reduciendo para luego hacerlo, y después se definió cuanto debe de disminuir la base para para seguir con la proporcionalidad del triángulo, en caso que por temas de enteros no se toma el número exacto, se cuenta con un else, que cuando se resta la diferencia a la altura de un número inferior a la altura deseada, la cambie automáticamente por la altura deseada

¡Pruébenlo! Capturan tres pantallas.





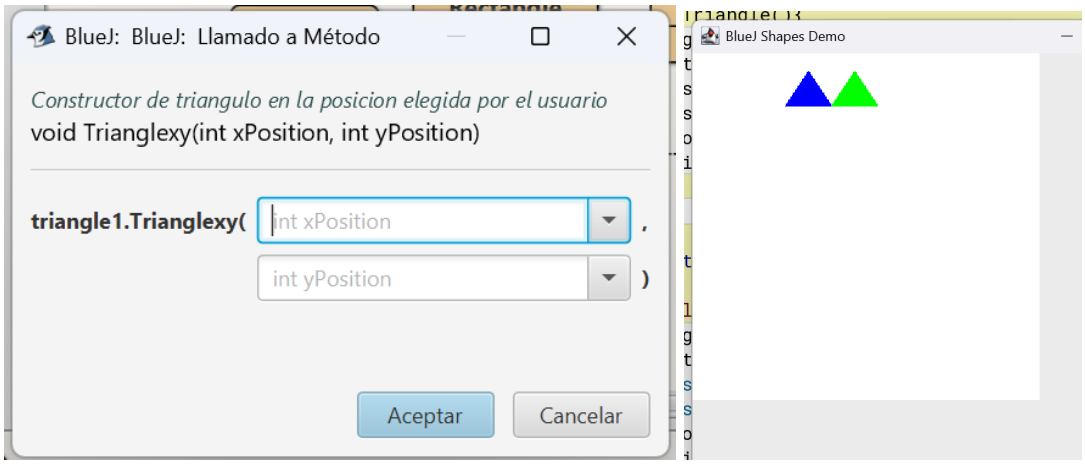


4. Desarrollen en Triangle un nuevo creador que permita crear un triángulo en una posición específica.

```
/**
 * Constructor de triangulo en la posicion elegida por el usuario
 */
    public void Trianglexy(int xPosition, int yPosition){
    height = 30;
    width = 40;
    this.xPosition = xPosition;
    this.yPosition = yPosition;
    color = "blue";
    isVisible = true;
}
```

Para este desarrollo se tomó como valor de entrada la posición para reemplazarla en los atributos necesarios para poder ilustrar el triángulo

¡Pruébenlo! Capturen una pantalla.

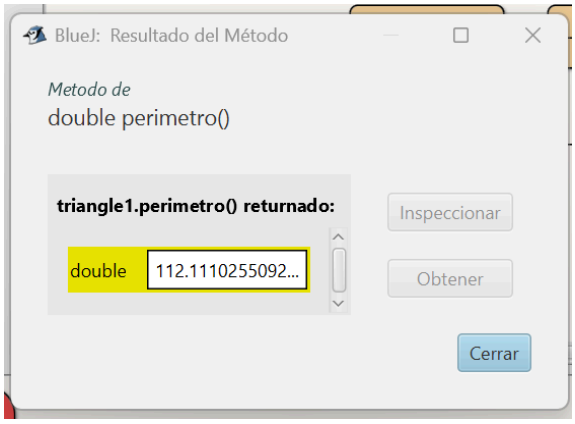


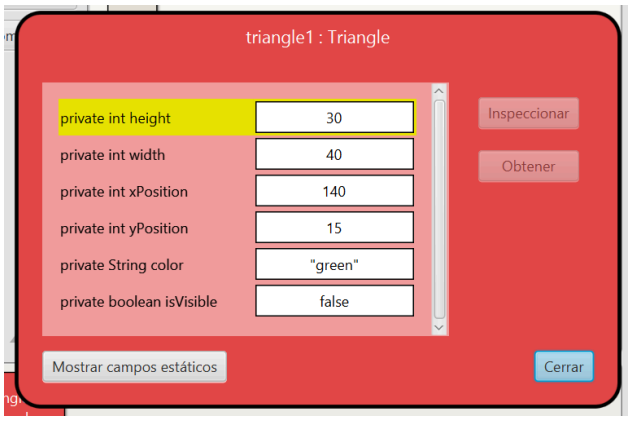
5. Propongan un nuevo método para esta clase. Desarrollen y prueban el método.

```java
/**
 * Metodo de perimetro para triangulos isosceles
 */
   public double perimetro(){
       double lado = Math.sqrt((height * height)+((width/2)*(width/2)));
       return ((lado*2)+width);
   }
```

Se desarrolló el método de perímetro para triángulos isósceles como los representados por esta clase, primero se halló la longitud de los lados congruentes y luego el perímetro con la ecuación de perímetro para triángulos isósceles.

Prueba





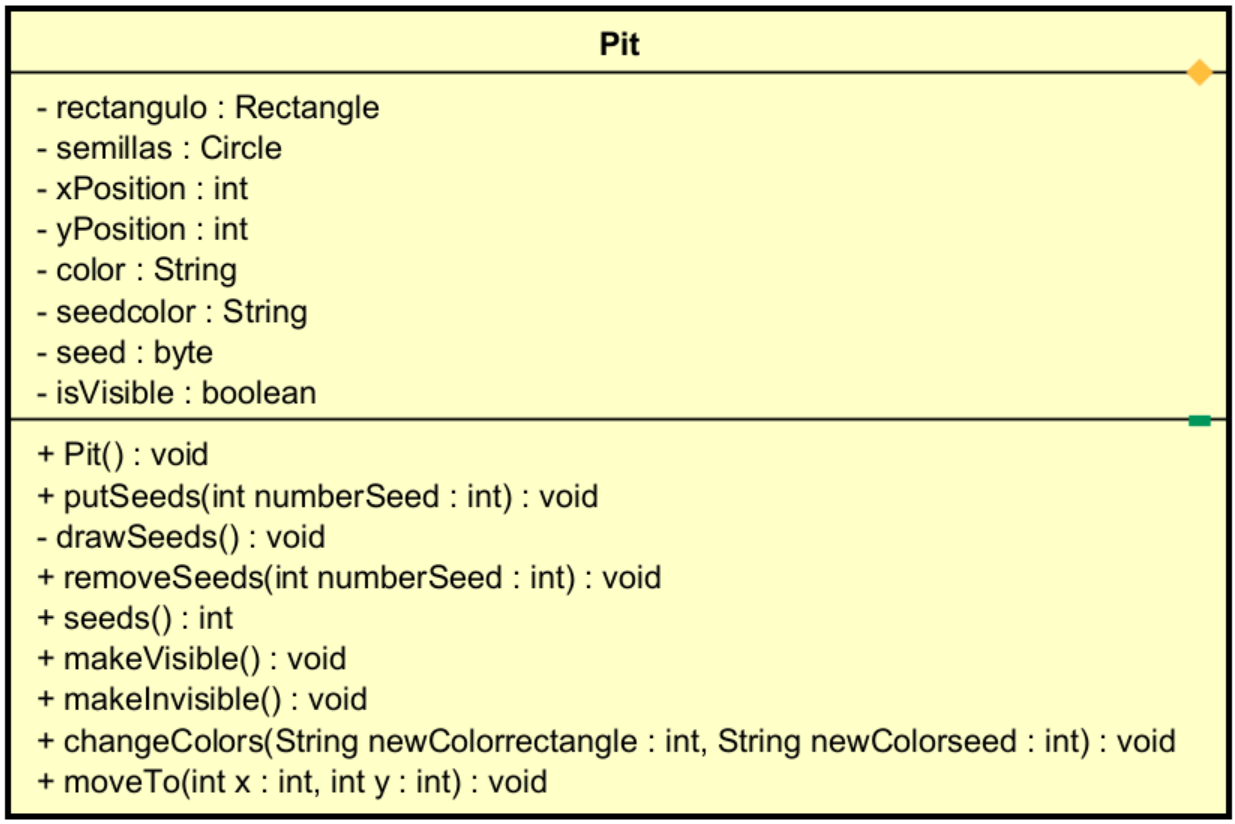
6. Generen nuevamente la documentación y revisen la información de estos nuevos métodos. Capturen la pantalla.

Modifier and Type	Method	Description
double	**area**()	Metodo area para triangulo
void	**changeColor**(String newColor)	Change the color.
void	**changeSize**(int newHeight, int newWidth)	Change the size to the new size
void	**equilateral**()	Metodo modificar a triangulos equilateros
void	**makeInvisible**()	Make this triangle invisible.
void	**makeVisible**()	Make this triangle visible.
void	**moveDown**()	Move the triangle a few pixels down.
void	**moveHorizontal**(int distance)	Move the triangle horizontally.
void	**moveLeft**()	Move the triangle a few pixels to the left.
void	**moveRight**()	Move the triangle a few pixels to the right.
void	**moveUp**()	Move the triangle a few pixels up.
void	**moveVertical**(int distance)	Move the triangle vertically.
double	**perimetro**()	Metodo de perimetro para triangulos isosceles
void	**shrink**(int times, int minHeight)	Metodo de shrink, cambia secuencialmenten la altura de un triangulo
void	**slowMoveHorizontal**(int distance)	Slowly move the triangle horizontally.
void	**slowMoveVertical**(int distance)	Slowly move the triangle vertically.
void	**Trianglexy**(int xPosition, int yPosition)	Constructor de triangulo en la posicion elegida por el usuario

Parte E.

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

Se eligieron estos atributos debido a que son los que se necesitarán en los métodos, estos son:



- **rectangulo**: Esta variable contiene al rectángulo, que será el objeto que usaremos como hueco
- **semillas:** Esta variable contiene una lista llena con todas las semillas que el usuario ha generado, se escogió una lista para poder eliminar, o desaparecer todas las semillas al tiempo
- **xPosition:** Está atributo contiene el valor actual de la posición en el eje x del rectángulo
- **yPosition:** Está atributo contiene el valor actual de la posición en el eje y del rectángulo
- **color:** Este atributo contiene el color actual del rectángulo
- **seedColor:** Este atributo contiene el color actual de las semillas
- **seed:** Este atributo contiene la cantidad de semillas en el hueco
- **isVisible:** Este atributo contiene un valor booleano que permite observar si el rectángulo es visible o no

```java
 * A well where to put seeds according to the number that a person foresees.
 *
 * @authors Anderson Fabian Garcia Nieto y Cristian Alfonso Romero Martinez
 * @version (January 31 2025)
 */
public class Pit
{
    // instance variables - replace the example below with your own
    private Rectangle rectangulo;
    private Circle[] semillas;
    private int xPosition;
    private int yPosition;
    private String color;
    private String seedcolor;
    private byte seed;
    private boolean isVisible;
```

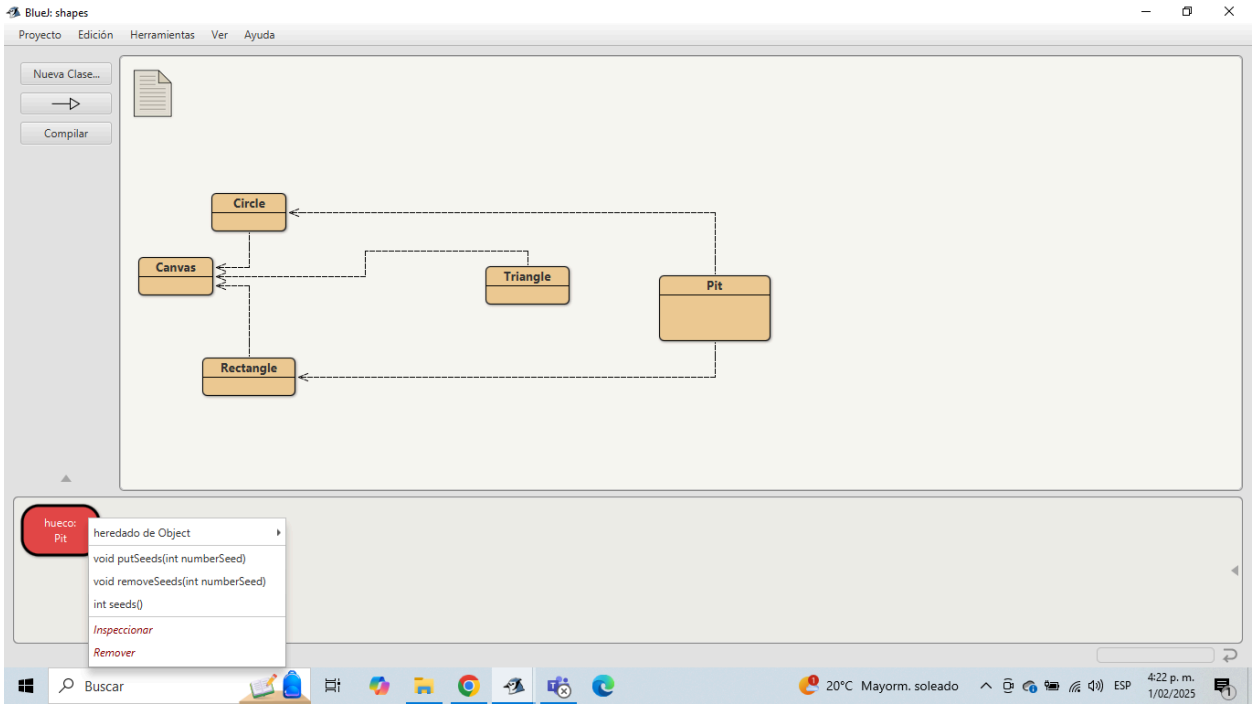
2. **Desarrollen la clase considerando los 3 mini-ciclos. Al final de cada mini-ciclo realicen dos pruebas indicando su propósito. Capturen las pantallas relevantes.**

Mini Ciclo 1

```java
//Mini Ciclo 1
   public Pit(boolean big)
   {
      if (big != true){
         this.seed = 0;
         this.color = "green";
         this.seedcolor = "yellow";
         this.xPosition = -10;
         this.yPosition = -90;
         this.semillas = new Circle[100];
         this.big = big;
         this.rectangulo = new Rectangle();
         this.rectangulo.changeColor(this.color);
      } else{
         this.seed = 0;
         this.color = "green";
         this.seedcolor = "yellow";
         this.xPosition = -10;
         this.yPosition = -90;
         this.semillas = new Circle[400];
         this.big = big;
         this.rectangulo = new Rectangle();
         this.rectangulo.height = this.rectangulo.height * 2;
         this.rectangulo.width = this.rectangulo.width * 2;
         this.rectangulo.changeColor(this.color);
      }


   }
```

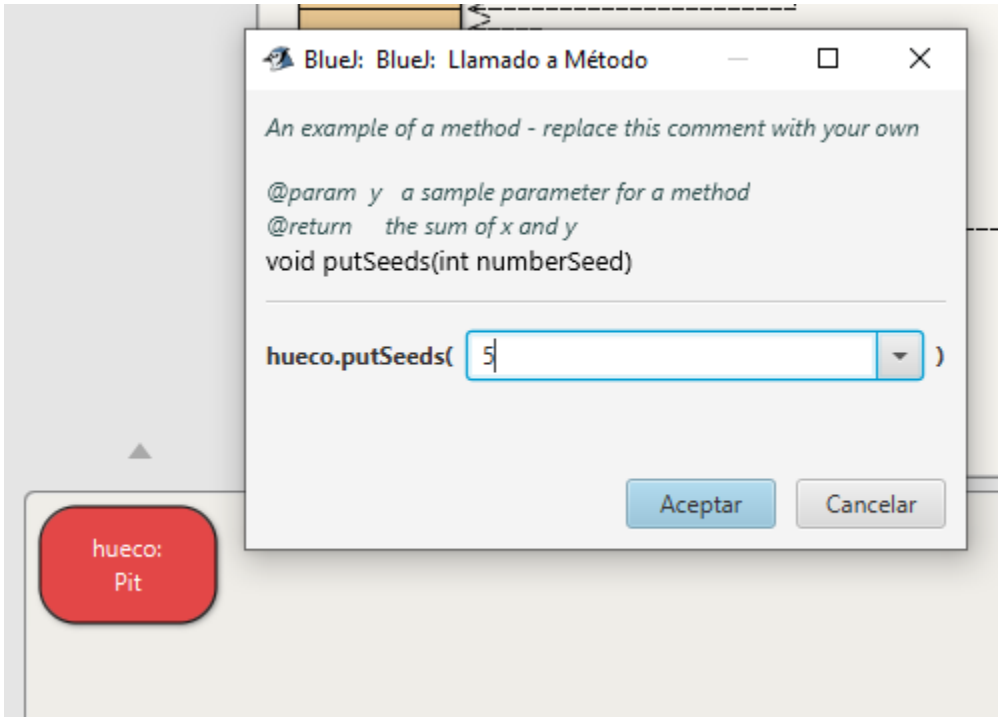
```
   /**
    * Increases the number of seeds for which the user logs in and draws them
    */
   public void putSeeds(int numberSeed)
   {   if (this.big){
       if (numberSeed > 0){
           if ((numberSeed + this.seed) <= 400){
               this.seed += numberSeed;
               drawSeeds();
           }
       }

   } else {
       if (numberSeed > 0){
           if ((numberSeed + this.seed) <= 100){
               this.seed += numberSeed;
               drawSeeds();
           }
       }
   }

   }

   /**
    * Draw the seeds since the number of seeds is greater than 0, also drawing them one
after the other.
    */
   private void drawSeeds() {
   if (this.big) {
   for (int i = 0; i < seed; i++) {
       if (semillas[i] == null) {
           semillas[i] = new Circle();
           semillas[i].changeColor("black");
           int seedx = xPosition + (i % 20) * 10 + 50;
           int seedy = yPosition + (i / 20) * 10 + 50;
           semillas[i].moveHorizontal(seedx);
           semillas[i].moveVertical(seedy);
           if (isVisible) {
               semillas[i].makeVisible();
           }
       }
   }
} else {
   for (int i = 0; i < seed; i++) {
       if (semillas[i] == null) {
           semillas[i] = new Circle();
```

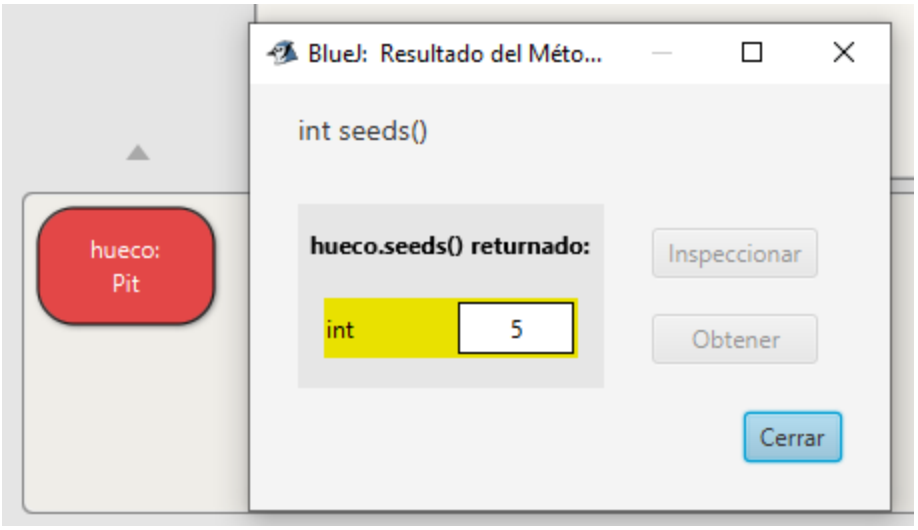
```
            semillas[i].changeColor("black");
            int seedx = xPosition + (i % 10) * 10 + 50;
            int seedy = yPosition + (i / 10) * 10 + 50;
            semillas[i].moveHorizontal(seedx);
            semillas[i].moveVertical(seedy);
            if (isVisible) {
               semillas[i].makeVisible();
            }
         }
      }
   }
}

}
   /**
    * Removes the seeds that the user removes, in turn deleting them.
    */
   public void removeSeeds(int numberSeed)
   {
    if (numberSeed > 0 && this.seed >= numberSeed) {
         this.seed -= numberSeed;
         for (int i = seed; i < seed + numberSeed; i++) {
            if (semillas[i] != null) {
               semillas[i].makeInvisible();
               semillas[i] = null;
            }
         }
      } else {
         System.out.println("La cantidad de semillas a remover sobrepasa las existentes");
      }
   }


   /**
    * Returns the number of seeds in the hole.
    */
   public int seeds()
   {
      return this.seed;
   }
```



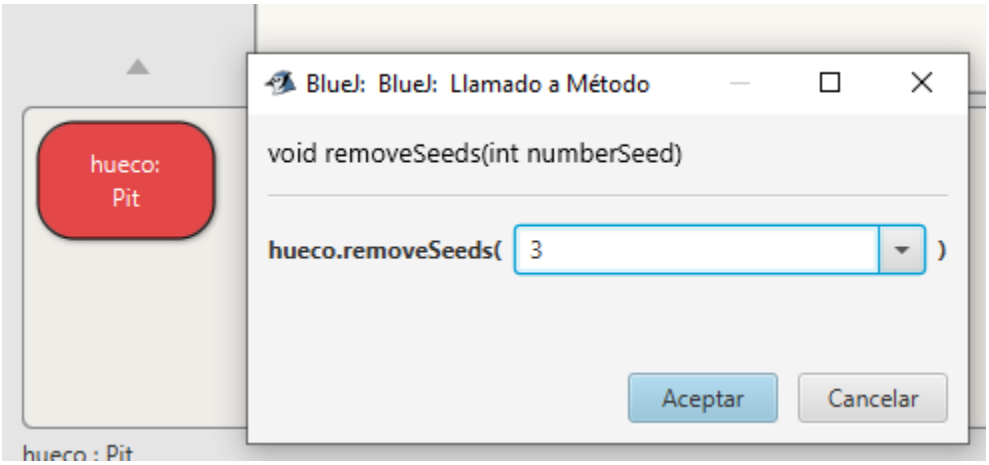
Pruebas

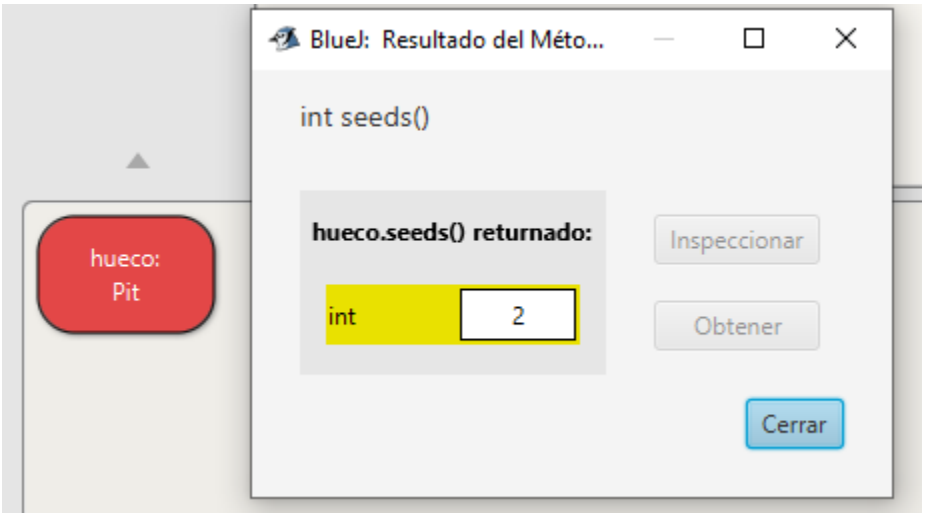
a) Agregar semillas y contar cuántas hay



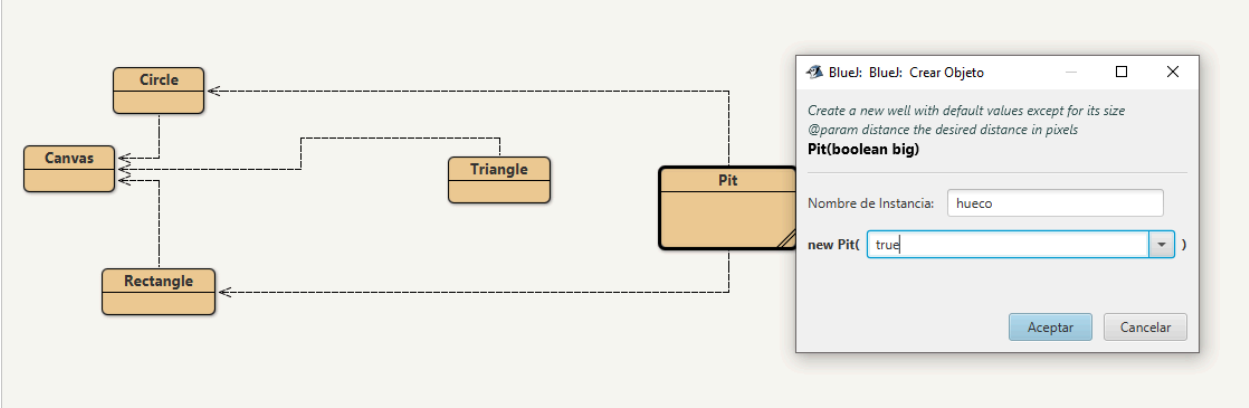


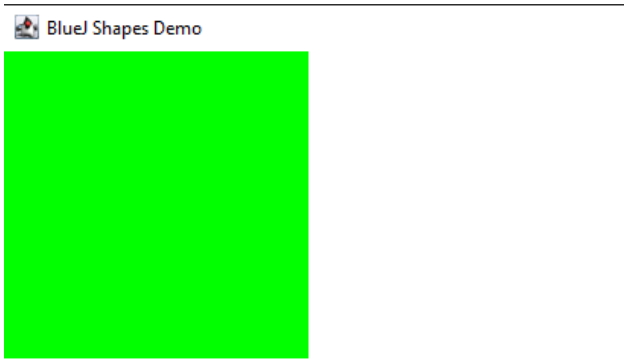
b) Remover semillas y contar



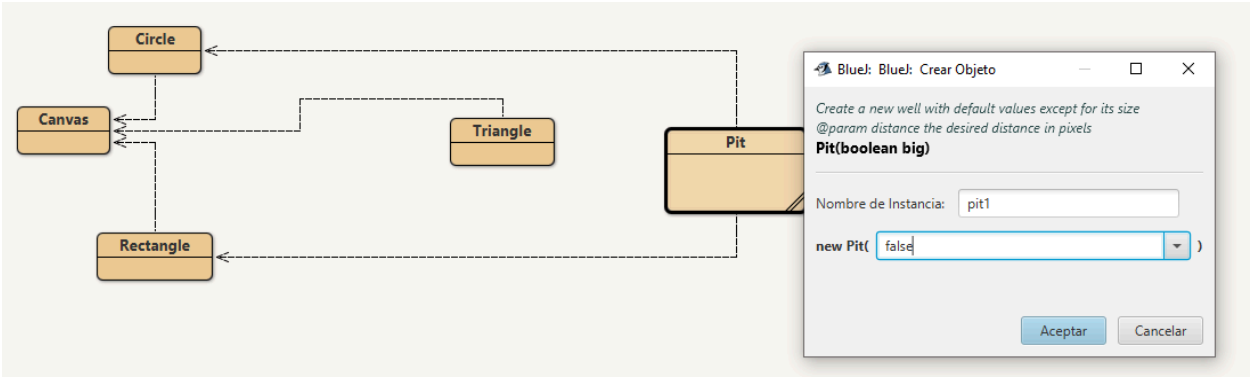


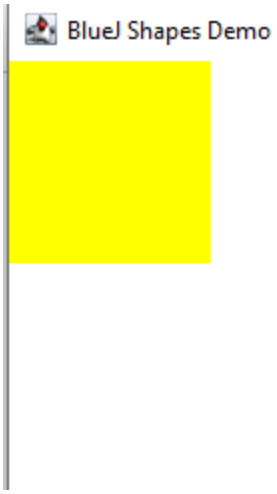
c) Creando el pit con la condición true





d) Creando el pit con la condición false

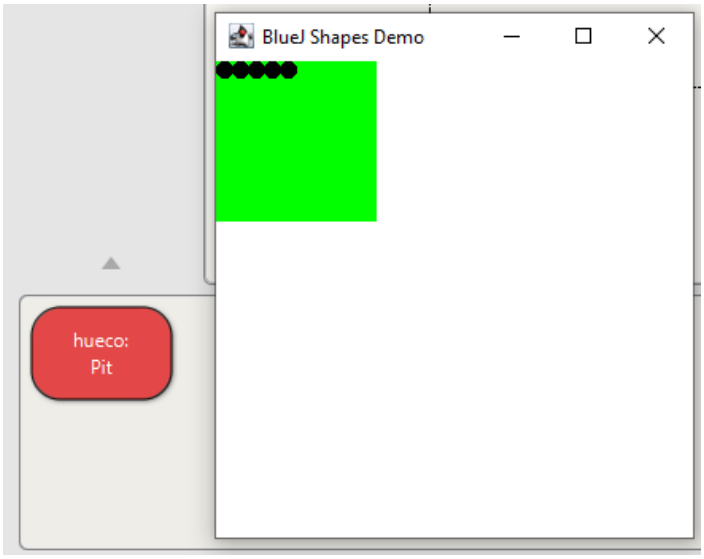




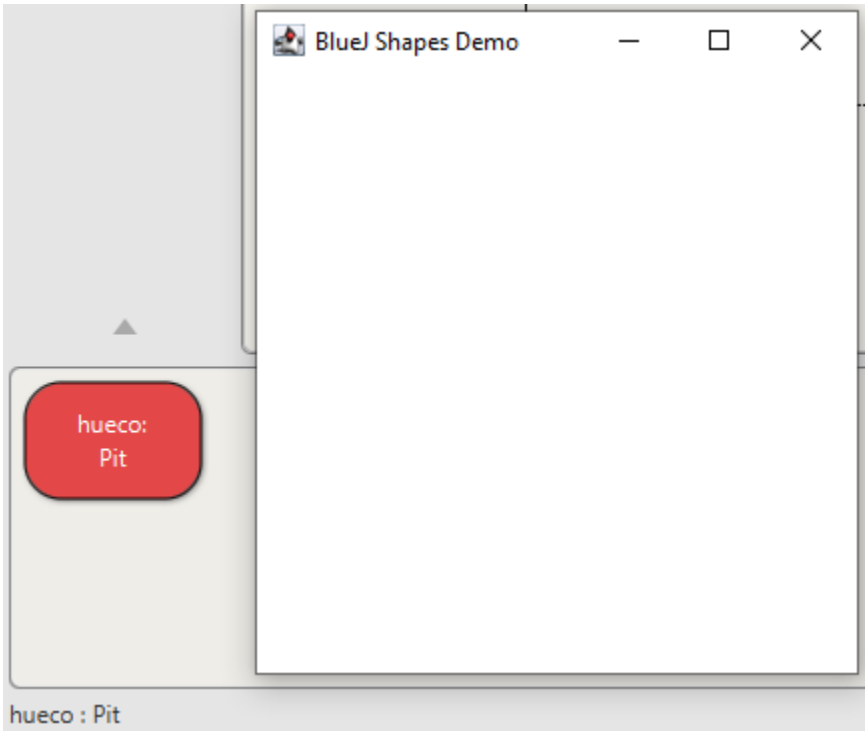
Mini Ciclo 2

```
//Mini Ciclo 2
   /**
    * Makes all elements of the canvas invisible
    */
   public void makeVisible() {
      rectangulo.makeVisible();
      isVisible = true;
      for (int i = 0; i < seed; i++) {
         if (semillas[i] != null) {
            semillas[i].makeVisible();
         }
      }
   }

   /**
    * Makes all elements of the canvas visible
    */
    public void makeInvisible() {
      rectangulo.makeInvisible();
      isVisible = false;
      for (int i = 0; i < semillas.length; i++) {
         if (semillas[i] != null) {
            semillas[i].makeInvisible();
         }
      }
   }
```

a) **Volviendo visibles los elementos**



b) Volviendo invisibles los elementos

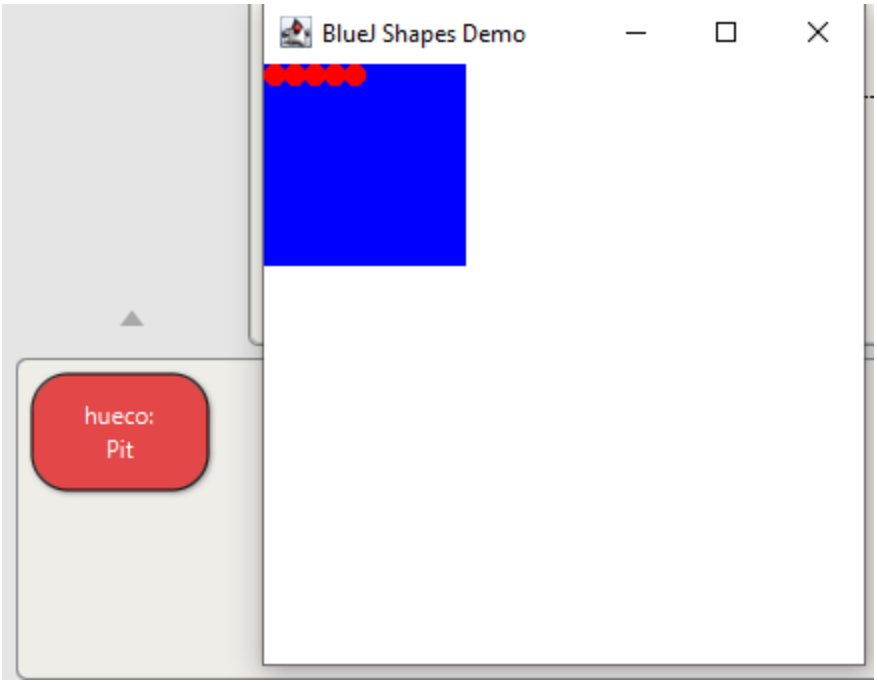


Mini ciclo 3

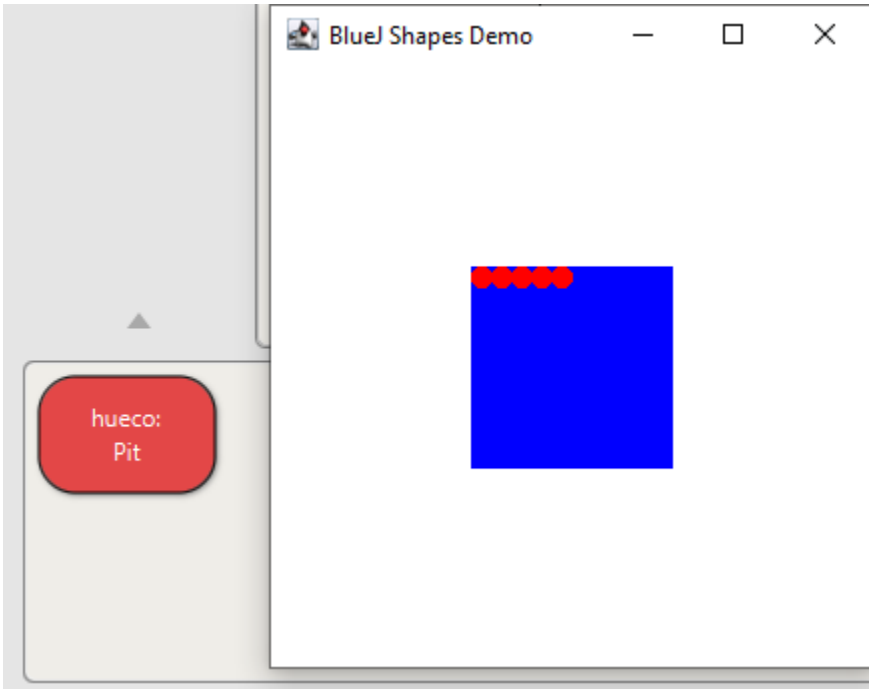
```
//Mini Ciclo 3
   /**
    * Changes the colors of the hole and of the seeds
    */
    public void changeColors(String newColorrectangle,String newColorseed) {
      this.rectangulo.changeColor(newColorrectangle);
      for (int i = 0; i < this.seed; i++) {
```

```
        this.color = newColorrectangle;
        this.seedcolor = newColorseed;
        semillas[i].changeColor(newColorseed);
      }
   }

   /**
    * Changes the location of the hole and the seeds
    */
   public void moveTo(int x ,int y ){
      rectangulo.moveVertical(y);
      rectangulo.moveHorizontal(x);
      this.xPosition = this.xPosition + x;
      this.yPosition = this.yPosition + y;
      makeInvisible();
      for (int i =0; i<this.seed; i++){
         semillas[i].moveVertical(y);
         semillas[i].moveHorizontal(x);
      }
      makeVisible();

   }

}


   }

}
```

a) Cambiar el color del pit y de las semillas

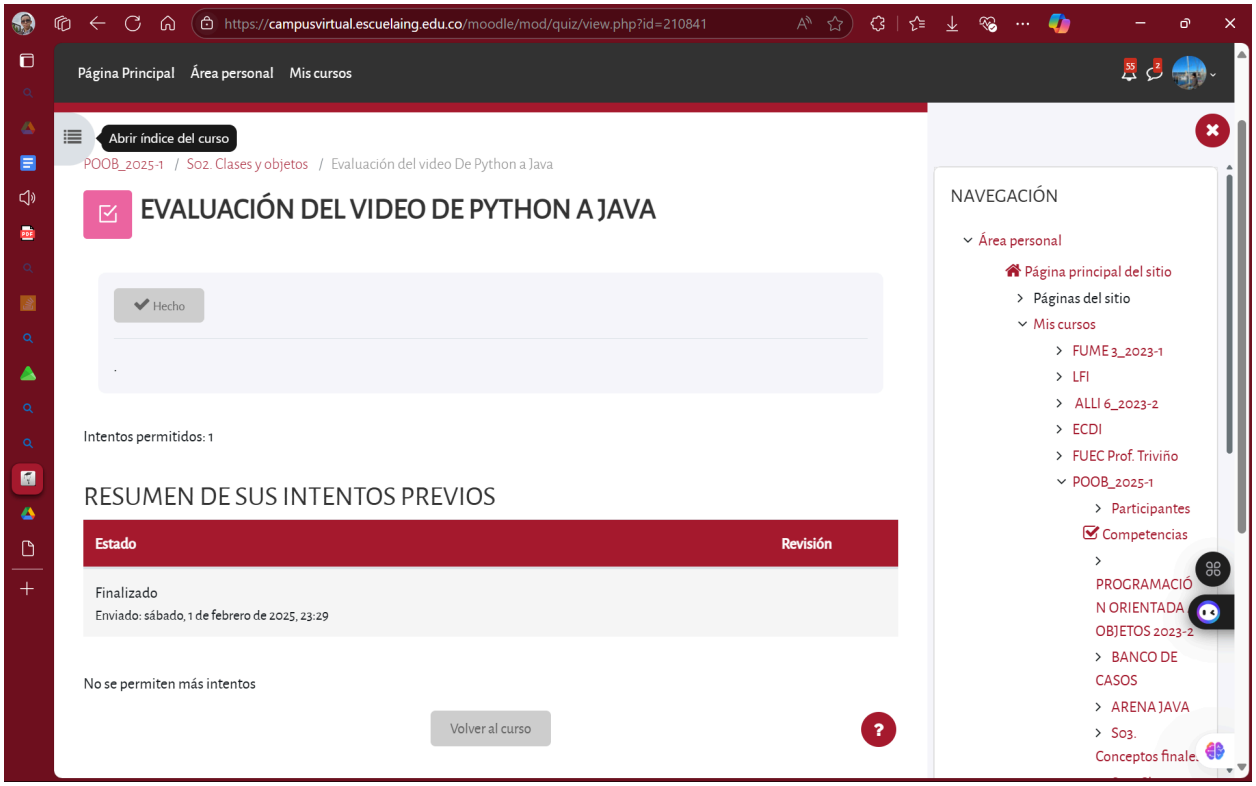


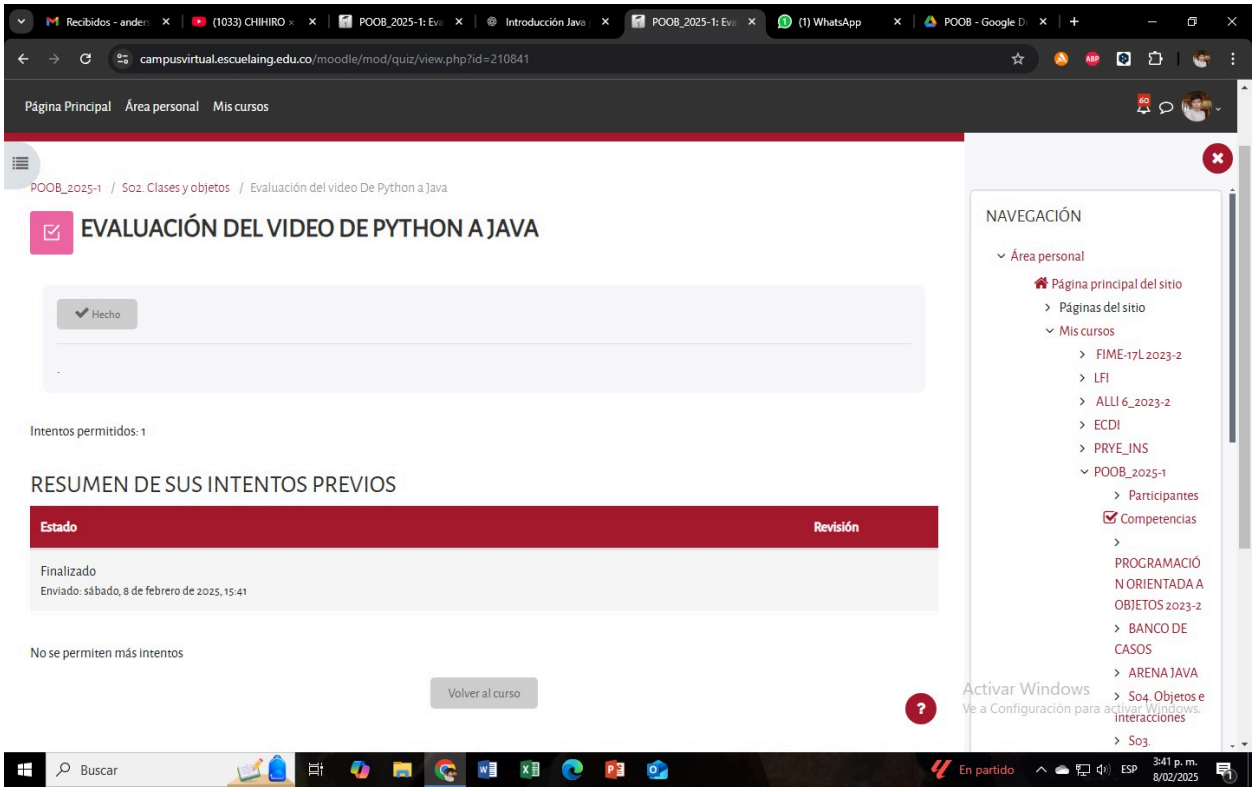
b) Desplazar el pit y las semillas según lo indicado



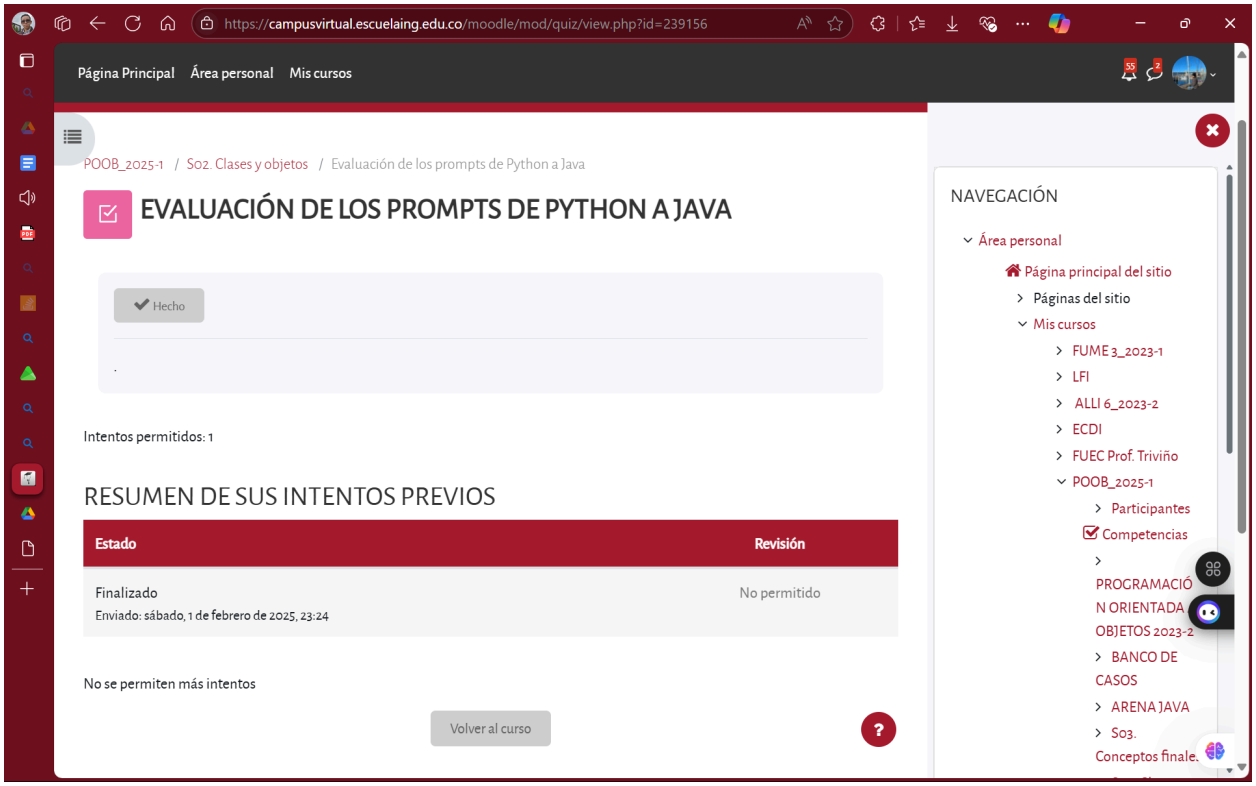
PARTE G

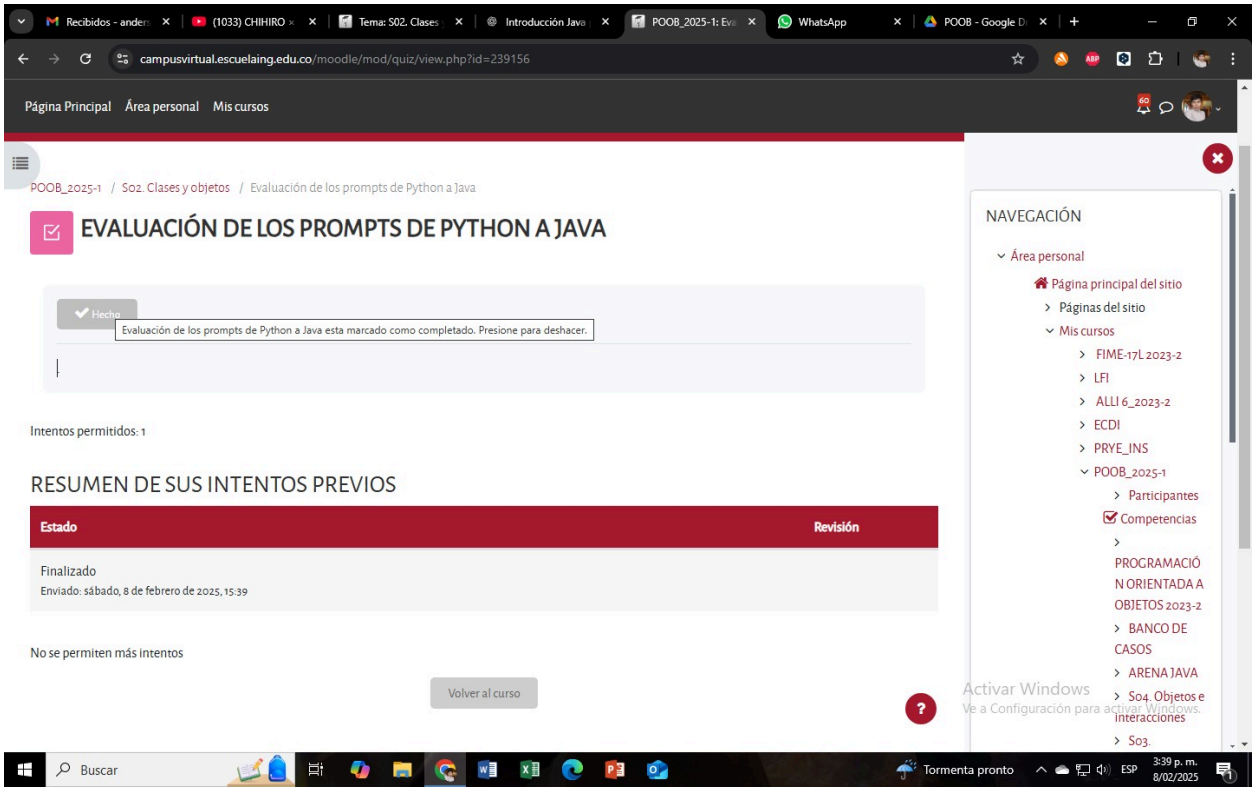
1. El video.





2. Los prompts.





PARTE F

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

Requisitos funcionales

Crear el estado inicial

Realizar los movimientos

Consultar el número de movimientos

Requisitos de interfaz

En caso que alguien gane, debe generar un mensaje de felicitación.

En caso que no sea posible realizar una de las acciones, se debe generar un mensaje de error. Use JoptionPane**.**

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

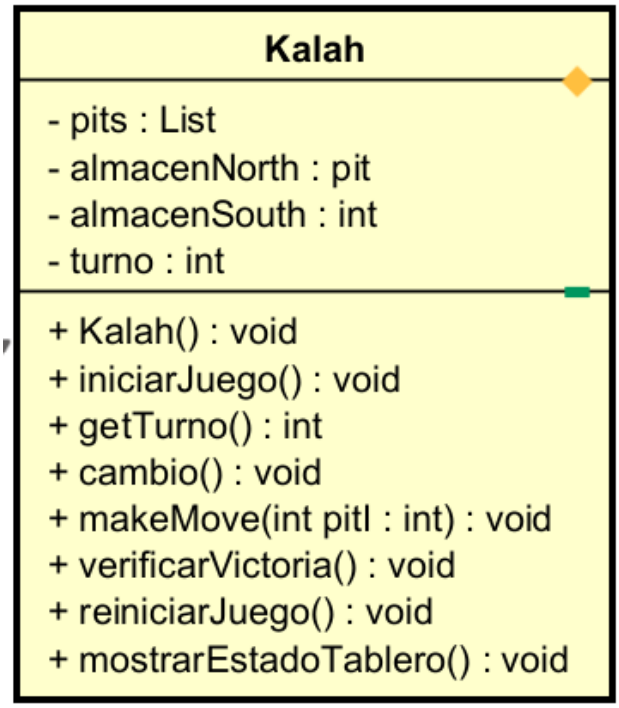
La clase Kalah debe ofrecer

Tablero default, para este caso debe contar con 12 cuadrados colocados en 2 filas, y cada una con 6, y en cada uno de sus extremos 2 rectángulos

Distribución inicial de las semillas, para este caso se debe colocar 4 semillas en cada uno de los cuadrados.

Partida, se debe tener un método donde el jugador elija la casa o cuadrado el cual va a desocupar, y seguido a esto, reparta las semillas en los cuadros siguientes, tomando la particularidad de los rectángulos de los extremos que es donde solo se guardan las semillas, así mismo debemos tomar una instancia que verifique si la última semilla se colocó en una casa vacía, pues esto indica que puede tomar las semillas de la caja del frente.

En caso que alguno de los jugadores no tenga movimientos proceder al conteo de las semillas de cada jugador, tanto las guardadas como las de las casas.



2. Planifiquen la construcción considerando algunos mini-ciclos.

Mini ciclo 1.

Creación del tablero y disposición inicial de las semillas.

Teniendo en cuenta los requerimientos iniciamos la construcción del tablero como un arraylist para facilitar la ejecución del mismo y la asignación de semillas default.

Mini ciclo 2.

Turnos y reglas de juego.

Se definen los turnos y el cambio de ellos uno a uno, luego definimos los cambios de forma numérica como jugador 1 y 2, para que sea más fácil llevar el control, luego definimos la que consideramos el método más complicado y vital para el desarrollo del proyecto, que es la de makemove, pues en esta se materializa la interacción de los jugadores, donde se destaca la validación de la elección del hoyo, desde si existe o si cuenta con semillas para repartir, luego de esto reparte las semillas y revisa generalidades de reglas como el robo de semillas o repetición de turno, sin dejar de lado la validación si hubo un ganador.

Mini ciclo 3.

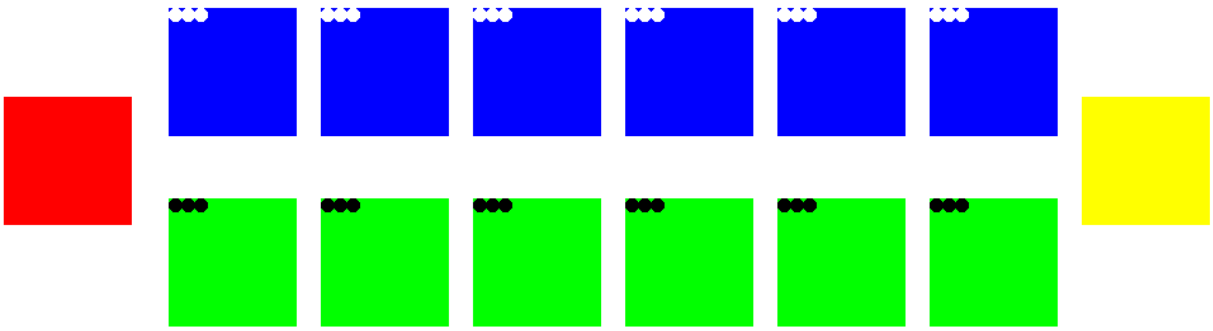
Muestra del tablero.

Segmentamos el tablero en 2 partes, del lado sur y del lado norte, para hacer un conteo real del estado de cada uno de los jugadores, es importante mencionar que nos apalancamos del pit para conocer el número de semillas asignadas a cada hoyo y almacén.

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

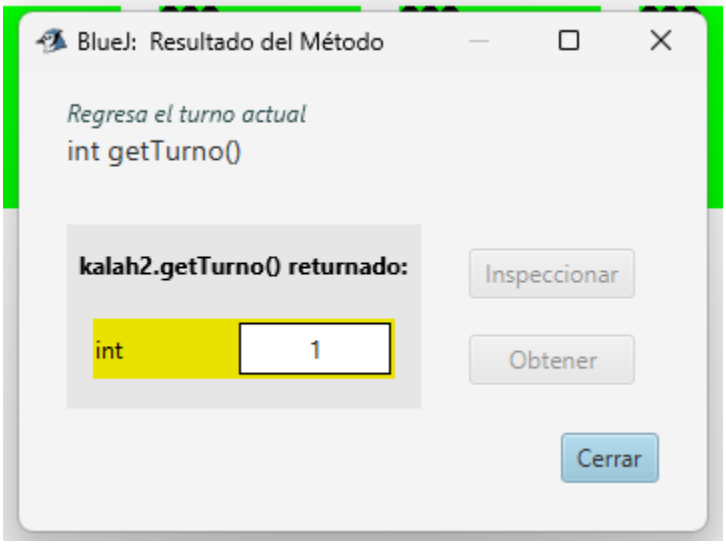
Mini ciclo 1.

Creación del tablero y disposición inicial de las semillas.



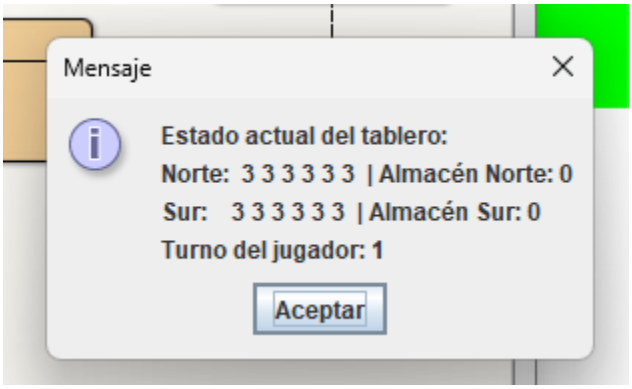
Mini ciclo 2

Turnos y reglas de juego.



Mini ciclo 3

Muestra del tablero.



4. Indiquen las extensiones necesarias para reutilizar la clase de los huecos y el paquete shapes. Expliquen.

Las extensiones usadas java.util.ArrayList y java.util.List; las usamos para poder ejecutar nuestra idea de almacenar los hoyos y almacenes como listas y poder navegar de la forma propuesta en el problema, y de esta forma podemos usar cada una de las clases sin cambios grandes dentro de ellos.

RETROSPECTIVA

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

Anderson Fabian Garcia Nieto 14 horas de trabajo

Christian Alfonso Romero Martinez 14 horas de trabajo

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

El estado del laboratorio terminado, consideramos que se debe al compromiso que tuvimos como equipo, al igual que el apoyo mutuo en el desarrollo.

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

La más útil fue la de la programación en pareja puesto que logramos percatarnos de errores así mismo me gustaría destacar las ideas para poder enfrentar los retos propuestos.

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

Consideramos que nuestro mayor logro fue la comunicación que nos permitió un desarrollo asertivo del laboratorio, al igual que la sumatoria de ideas para lograr estrategias que cumplieran con los requerimientos

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

Nuestro mayor problema técnico fue el cambio constante de enunciados, ya que al intentar avanzar y culminar el laboratorio con anticipación, el cambio de enunciados hizo perder trabajo ya hecho, además que no nos percatamos al instante del cambio, para solucionar las dificultades, fue crucial el trabajo en equipo la comunicación para compensar esto.

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

Como equipo consideramos que hicimos un buen proceso de desarrollo y de comunicación, y como compromiso para mejorar, nos comprometemos a continuar fortaleciendo la comunicación y búsqueda de herramientas de trabajo conjunto.

7. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares Adecuados.

Barker, J. (2005). *Beginning Java Objects: From Concepts to Code* (2nd ed.). Apress.
Oracle. (n.d.). *Java Platform, Standard Edition Documentation*. Oracle
Copilot. (2025). Respuesta a consulta sobre implementación de arrays en Java. Microsoft.

La referencia más útil fue la API de Java pues nos permitió conocer documentación para aplicarla en el desarrollo del proyecto.
También copilot fue de bastante ayuda ya que nos permitió ejemplos y estrategias para implementar los arrays