

PROGRAMACIÓN ORIENTADA A OBJETOS

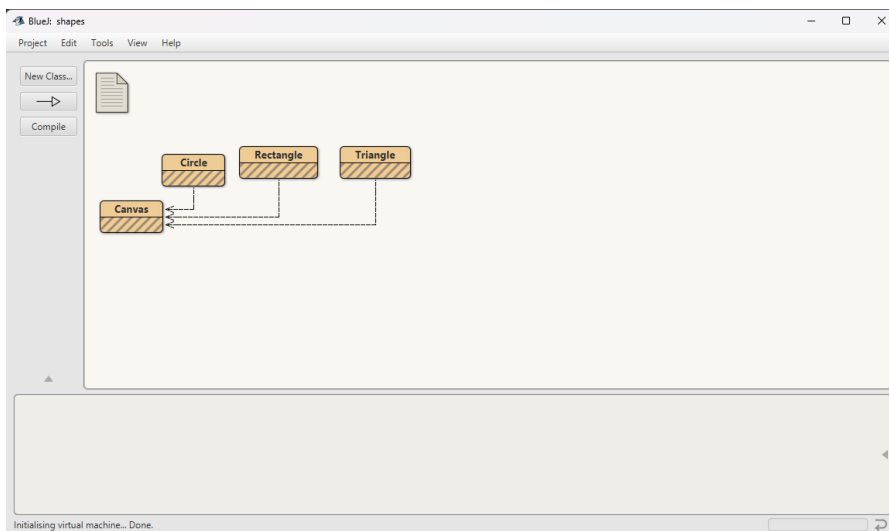
Construcción. Clases y objetos.

2025-1

Laboratorio 1/6

1. El proyecto “shapes” es una versión modificada de un recurso ofrecido por BlueJ.

Para trabajar con él, bajen shapes.zip y ábralo 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?

- Circle
- Rectangle
- Triangle
- Canvas

(b) ¿Qué relaciones existen entre ellas?

Todas son figuras geometricas

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?

All Classes

[Canvas](#)

[Circle](#)

[Rectangle](#)

[Triangle](#)

(b) ¿Qué atributos tiene la clase Triangle?

Ninguno publico

(c) ¿Cuántos métodos ofrece la clase Triangle?

Method Summary	
void	changeColor (String newColor) Change the color.
void	changeSize (int newHeight, int newWidth) Change the size to the new size
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.
void	slowMoveHorizontal (int distance) Slowly move the triangle horizontally.
void	slowMoveVertical (int distance) Slowly move the triangle vertically.

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

```
private int height;  
private int width;
```

(e) ¿Cuáles métodos ofrece la clase Triangle para cambiar su tamaño?

void	<code>changeSize(int newHeight, int newWidth)</code> Change the size to the new size
------	---

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?

```
public static int VERTICES=3;

private int height;
private int width;
private int xPosition;
private int yPosition;
private String color;
private boolean isVisible;
```

Tiene 7 atributos.

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

Todos los usuarios pueden usar los atributos públicos ya que no tienen restricciones.

(c) ¿Cuántos métodos tiene en total?

Tiene 14 métodos en total.

(d) ¿Quiénes usan los métodos privados?

Los usuarios con permisos de editar el código pueden acceder a los métodos privados.

(e) Desde el editor, consulte la documentación. Capture la pantalla.

```
/**
 * A triangle that can be manipulated and that draws itself on a canvas.
 *
 * @author Michael Kolling and David J. Barnes
 * @version 1.0 (15 July 2000)
 */
```

```
/**
 * Create a new triangle at default position with default color.
 */
```

```
/**
 * Make this triangle visible. If it was already visible, do nothing.
 */
```

```
/**
 * Make this triangle invisible. If it was already invisible, do nothing.
 */
```

```
/**
 * Move the triangle a few pixels to the right.
 */
```

```
/**
 * Move the triangle a few pixels to the left.
 */
```

```
/**
 * Move the triangle a few pixels up.
 */
```

```
/**
 * Move the triangle a few pixels down.
 */
```

```
/**
 * Move the triangle horizontally.
 * @param distance the desired distance in pixels
 */
```

```
/**
 * Move the triangle vertically.
 * @param distance the desired distance in pixels
 */
```

```
/**
 * Slowly move the triangle horizontally.
 * @param distance the desired distance in pixels
 */
```

```
/**
 * Slowly move the triangle vertically.
 * @param distance the desired distance in pixels
 */
```

```
/**
 * 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.
 */
```

```
/**
 * Change the color.
 * @param color the new color. Valid colors are "red", "yellow", "blue", "green",
 * "magenta" and "black".
 */
```

5. Comparando la documentación con el código

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

En la documentación no se ve todo lo que tenga como referencia el estado Privado, ejemplo los metodos.

(b) ¿por qué debe ser así?

Por razones de seguridad y diseño, puesto que hay propiedades que simplemente no deberían manipularse, por fines de seguridad.

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

(a) ¿Qué significa que sea public?

Significa que cualquier usuario puede visualizar el atributo, clase o método.

(b) ¿Qué significa que sea static?

Static significa que el valor el cual se le asigno a una variable no se podrá cambiar una vez se ejecute el programa

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

Final significa que el valor el cual se le asigno a una variable no se podrá cambiar una vez se ejecute el programa. Si debe serlo debido a que hablamos de un triángulo, no tendría sentido de que los vértices del triángulo variaran.

(d) Actualícelo.

```
public class Triangle{  
    public final int VERTICES=3;  
  
    private int height;  
    private int width;  
    private int xPosition;  
    private int yPosition;  
    private String color;  
    private boolean isVisible;
```

Class compiled - no syntax errors

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?.

Que es una variable de tipo entero, que almacena enteros, además del rango que puede llegar a alcanzar.

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

El área del rectángulo más grande posible seria de 16 129 unidades cuadrados.

(c) y ¿si fuera long?

El area del rectangulo maximo es 85070591730234615847396907784232501249 unidades cuadradas.

d) ¿qué restricción adicional deberían tener este atributo?

Que height tiene que ser mayor o igual a cero.

(e) Refactoricen el código considerando (d).

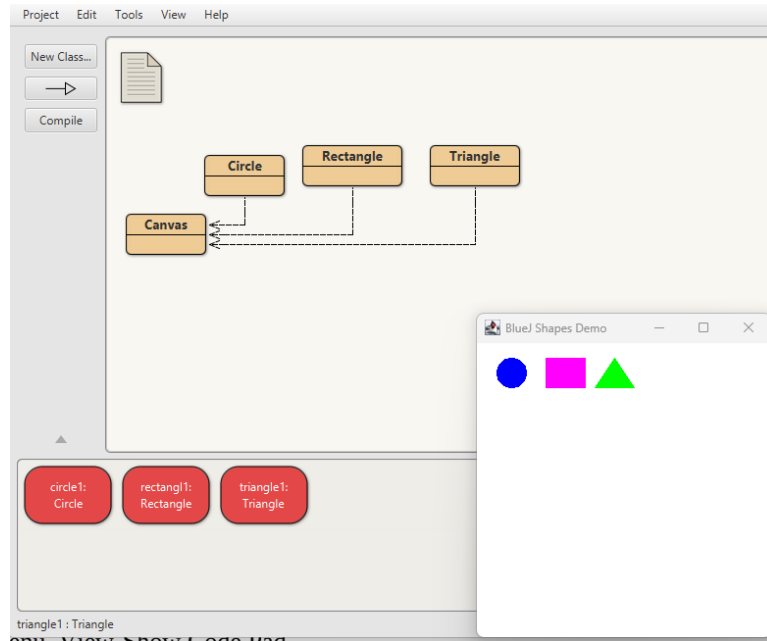
```
public Triangle(){  
    height = 30;  
    if(height < 0){  
        height = 0;  
    }  
  
    width = 40;  
    xPosition = 140;  
    yPosition = 15;  
    color = "green";  
    isVisible = false;  
}
```

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

La representacion de figuras geometricas.

B. Manipulando objetos. Usando un objeto.

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



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

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

Que cada uno tiene propiedades de las figuras a las que representa

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

Que el canvas tiene diferentes propiedades, imaginamos que porque es el encargado de dejar ver al resto de figuras.

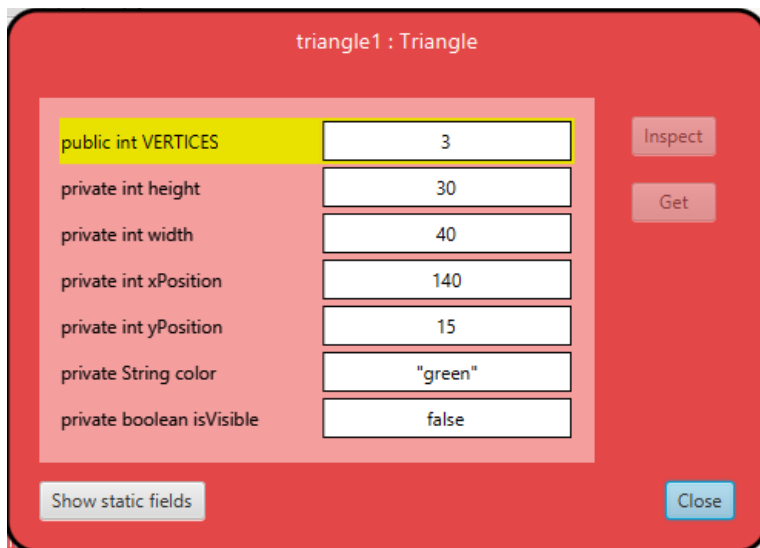
3. Inspeccionen el estado del objeto :Triangle.

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

- Vertices: 3
- Height : 30

- Width: 40
- xPosition: 140
- yPosition: 15
- color: Green
- isVisible: false

(b) Capturen las pantallas.



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

(a) Capturen la pantalla.

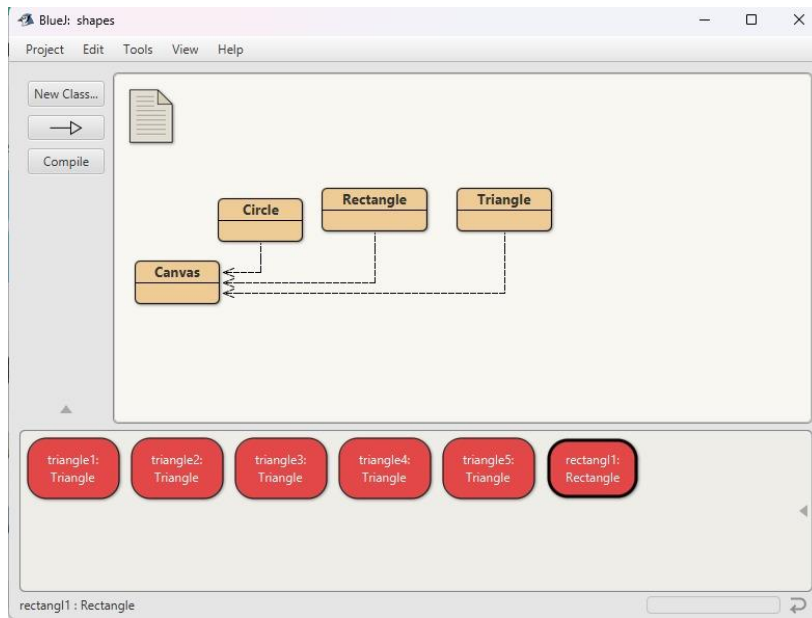
```
void changeColor(String newColor)
void changeSize(int newHeight, int newWidth)
void makeInvisible()
void makeVisible()
void moveDown()
void moveHorizontal(int distance)
void moveLeft()
void moveRight()
void moveUp()
void moveVertical(int distance)
void slowMoveHorizontal(int distance)
void slowMoveVertical(int distance)
```

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

No aparecen todos puesto que estos son los que están establecidos como públicos si no se encuentran allí, significa que los comportamientos son privados.

5. Construyan, con “shapes” sin escribir código, una propuesta de la imagen del logode su chatbot IA favorito.

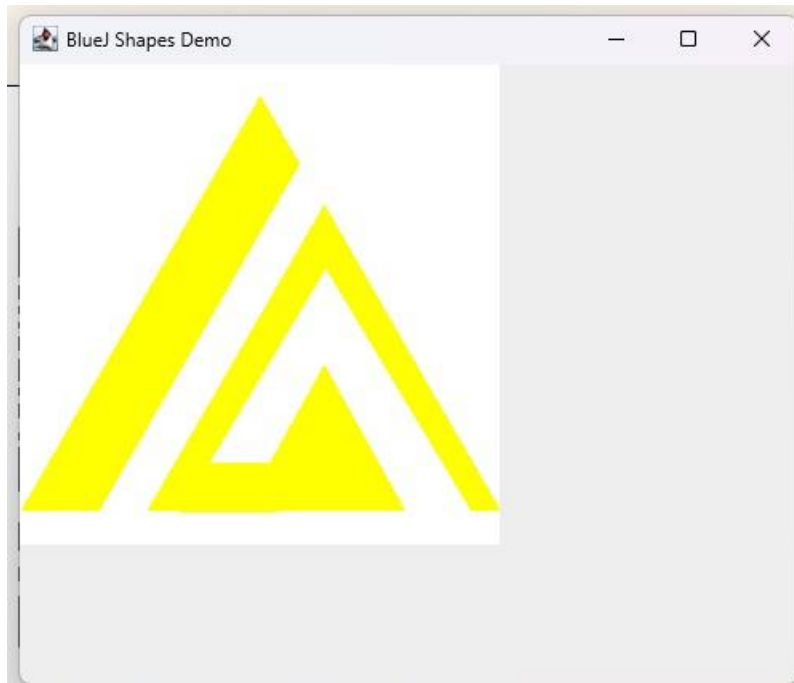
(a) ¿Cuántas y cuáles clases se necesitan



(b) ¿Cuántos objetos se usan en total?

En la figura se utilizan un total de 2 objetos, el triángulo y el rectángulo.

(c) Capturen la pantalla



(d) Incluyan el logo original.



C. Manipulando objetos. Analizando y escribiendo código.

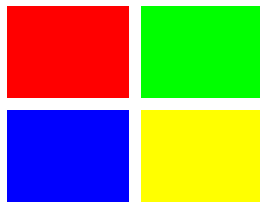
1. Lean el código anterior.

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

Un cuadrado conformado por cuadrados.

(b) Píntenla.

Es el logo de microsoft =)



2.

```
yellow.changeColor("yellow");  
yellow.moveHorizontal(45);  
yellow.moveVertical(35);  
yellow.makeVisible();
```

Para cada punto señalado indiquen:

(a) ¿cuántas variables existen?

4 variables existentes

Amarillo, verde, rojo y azul

(b) ¿cuántos objetos existen?

Existen 3 objetos, Circulo, Triangulo, Rectangulo

(no cuenten ni los objetos String ni el objeto Canvas)

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

Amarillo, verde, azul, rojo

(d) ¿cuántos objetos se ven?

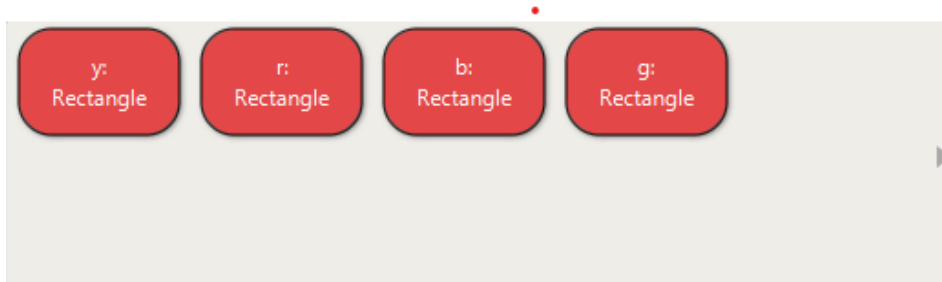
Se ve un objeto, el rectangulo.

(e) Expliquen sus respuestas.

el objeto es el rectangulo y hay 4 variables de tipo Rectángulo, que es el

objeto

(f) Capturen la pantalla.



4. Compare figura pintada en 1. con la figura capturada en 2. ,

(a) ¿son iguales?

Sí, son exactamente iguales, tanto en tamaño como en posición o en color.

(b) ¿por qué?

son las mismas instrucciones ingresadas de diferente manera.

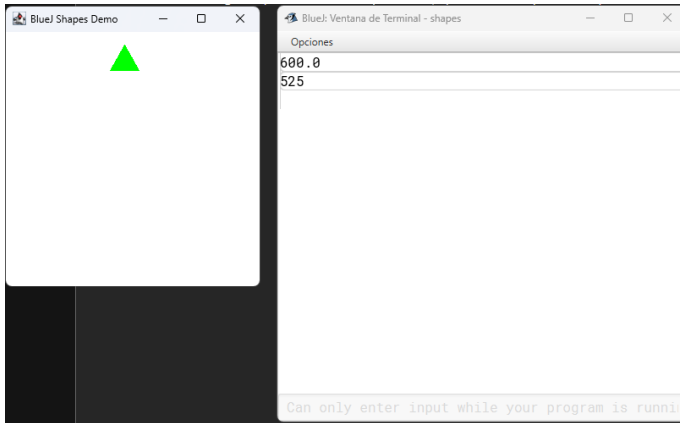
D. Extendiendo una clase. Triangle.

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

A screenshot of a Java IDE. The top part shows a terminal window titled 'BlueJ: Ventana de Terminal - shapes' with a button 'Opciones' and the output '600.0'. Below the terminal is a code editor with a yellow background, showing the following Java code:

```
/*
 * Desarrollamos el metodo area
 */
public void area(){
    float area;
    area= (height*width)/2;
    System.out.println(area);
}
```

2. Desarrollen en Triangle el método equilateral() (transforma el triángulo en un triángulo equilateral de área equivalente) . ¡Pruébenlo! Capturen dos pantallas.



Como se puede ver en la ventana de la izquierda el primer dato que aparece es el área del triángulo no equilátero, después de eso el triángulo se transforma en equilátero y se vuelve a hallar el área para confirmar que el triángulo si es equilátero.

```
/**
 *
 * Desarrollamos el metodo equilateral()
 */
private double sideEquilateral(){
    int areaequilateral;
    areaequilateral = (height*width)/2;

    double lado;
    lado=Math.sqrt(4*areaequilateral/(Math.sqrt(3)));

    return lado;
}

public void equilateral(){
    int newWidth = (int) sideEquilateral();
    int newHeight = (int) (Math.sqrt(3)* newWidth)/2;

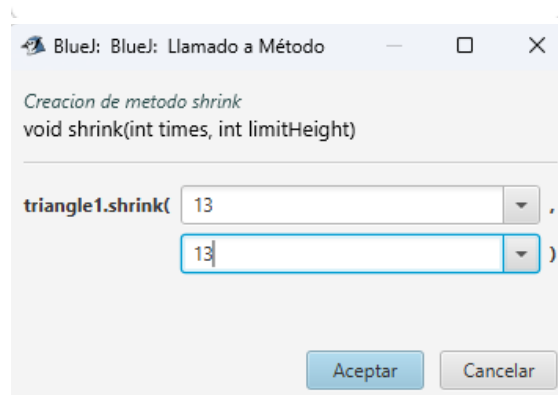
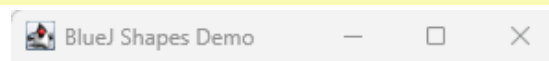
    changeSize(newHeight, newWidth);
}
```

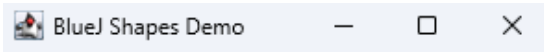
3. Desarrollen en Triangle el método shrink(times:int, height: int) (disminuye su tamaño times veces. Hasta llegar a una altura de height.) ¡Pruébenlo! Capturen tres pantallas.

```

/**
 * Creacion de metodo shrink
 */
public void shrink(int times, int limitHeight){
    if (limitHeight<0){
        limitHeight=0;
    }
    double timesReduction= (height-limitHeight)/times;
    for(int i=0; i< times; i++){
        int newHeight= (int) (height - timesReduction);
        changeSize(newHeight, width);
    }
}

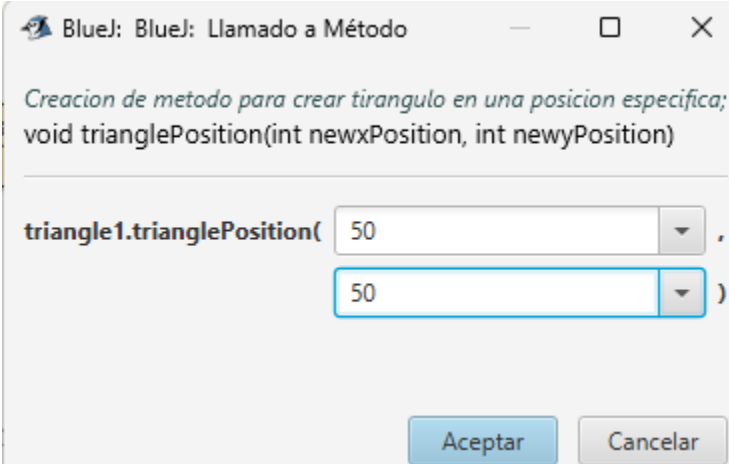
```

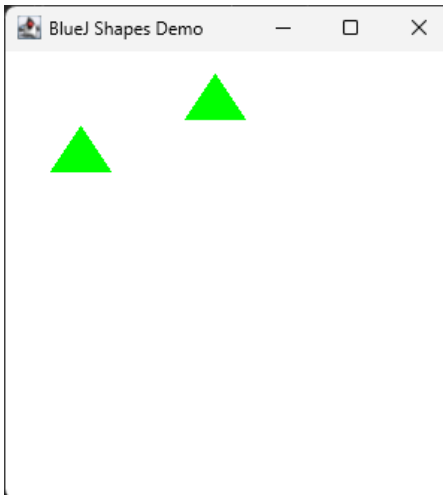




4. Desarrollen en Triangle un nuevo creador que permita crear un triangulo en una posición específica.¡Pruébenlo! Capturen una pantalla.

```
/**
 * Creacion de metodo para crear tirangulo en una posicion especifica;
 */
public void trianglePosition(int newXPosition, int newYPosition){
    Triangle newTriangle = new Triangle();
    newTriangle.xPosition = newXPosition;
    newTriangle.yPosition = newYPosition;
    newTriangle.makeVisible();
}
```



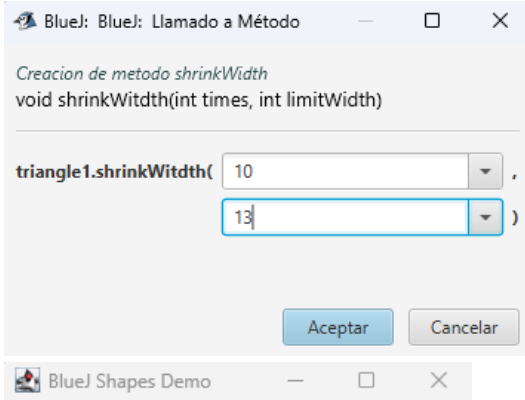
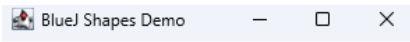


Como se puede ver en la imagen, se crea un nuevo triángulo en la posición que se desee.

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

```
/**
 * Creacion de metodo shrinkWidth
 */
public void shrinkWitdth(int times, int limitWidth){
    if (limitWidth<0){
        limitWidth=0;
    }
    double timesReduction= (width-limitWidth)/times;

    for(int i=0; i< times; i++){
        int NewWidth= (int) (width - timesReduction);
        changeSize(height, NewWidth);
    }
}
```



E. Creando una nueva clase. Usando un paquete. Shapes

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

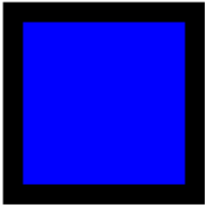
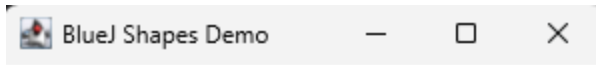
Para la construcción de the pit se hicieron uso de los siguientes atributos:

```
private Rectangle rect1, rect2;  
private ArrayList<Rectangle> seeds;  
private int x, y;  
private String pitColor, seedColor;  
private boolean isBig;
```

Estos tienen la función de darle vida a los objetos siendo rect1 y rect2 los rectángulos, seeds un arreglo para guardar las semillas, la posición de los cuadrados, el color del pit y el color de las semillas y un booleano el cual definirá si el cuadrado es el grande o es el pequeño.

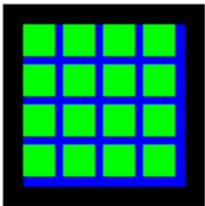
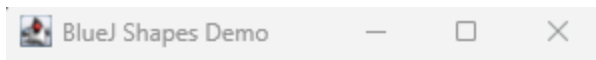
2. **Desarrollen la clase considerando los 3 mini-ciclos. Al final de cada mini- ciclo realicen dos pruebas indicando su propósito. Capturen las pantallas relevantes.**

Mini – ciclo: 1

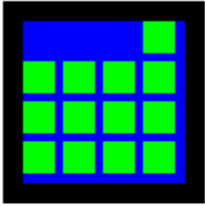
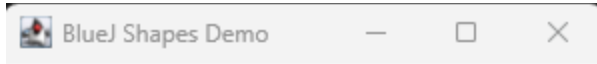


Inicia el juego con el tablero vacío.

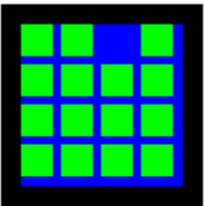
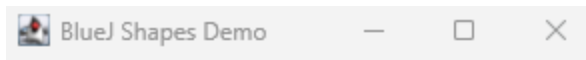
Se decide llenar todo el tablero por lo que se colocan 16 semillas:



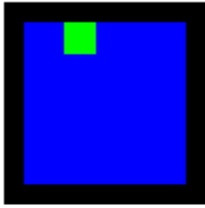
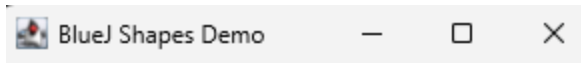
Se deciden eliminar 3 semillas:



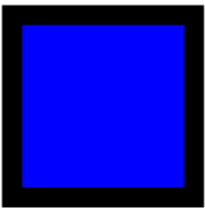
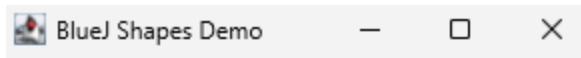
Luego se agregan 2:



Se deciden eliminar 14:

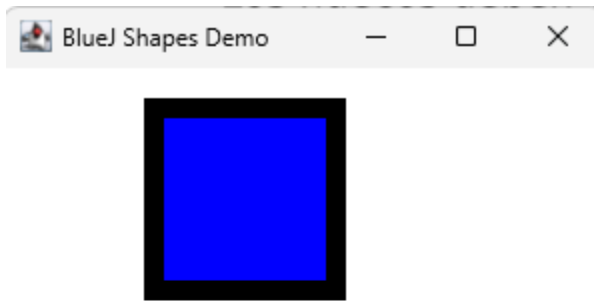


Se elimina la ultima semilla:

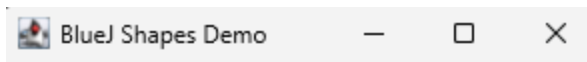


Mini – Ciclo: 2

Se usa el metodo makeVisible():

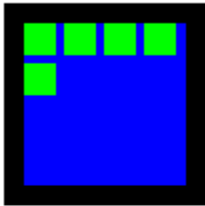
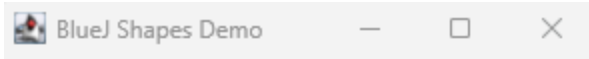


Se usa el metodo makeInvisible():

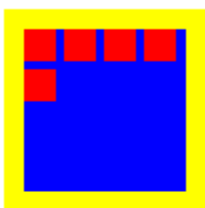
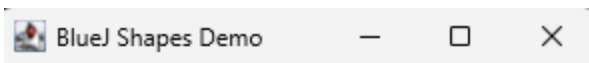


Mini - ciclo: 3

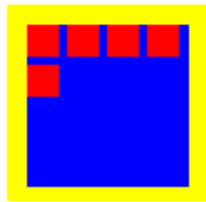
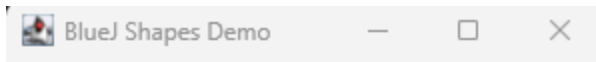
Se inicia creando el tablero con 5 semillas:



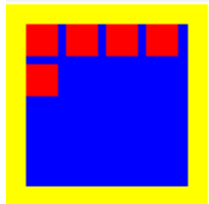
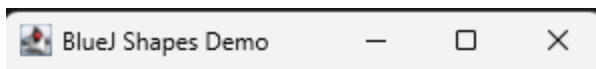
Se decide cambiar los colores de las semillas y del background del tablero, se quiere colocar el tablero de color amarillo y las semillas color rojo:



Con el mismo tablero se decide llevarlo a la posición x: 150, y:200:

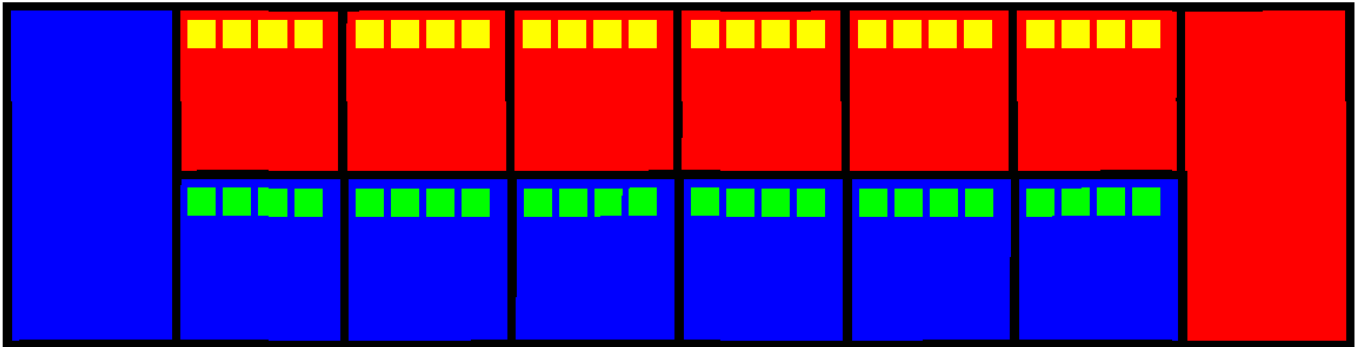


Al tenerlo ahí se decide llevarlo a la posición x:1,y:1:



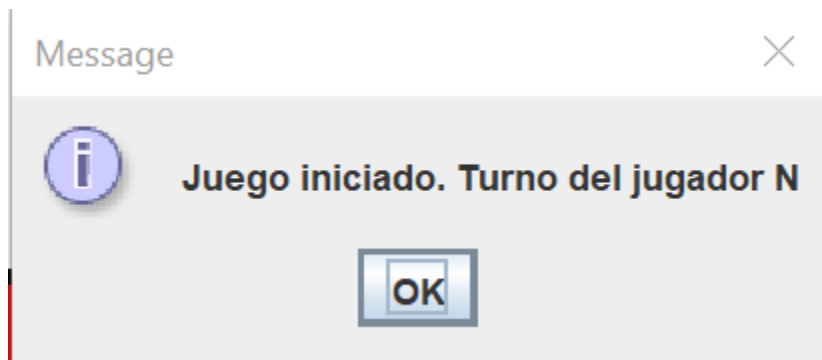
F. Definiendo y creando una nueva clase. Kalah.

Definimos el “KalahGame” utilizando el Pit, que a su vez usa Rectangle para generar cada elemento, desde las seeds.



A fin de reutilizar los objetos anteriormente definidos utilizamos Pit, con sus argumentos true y false para hacer los costados del tablero, y las secciones de en medio, dentro del Pit ya definimos con anterioridad la forma de agregar las seeds, de esta manera, agregamos el estado inicial.

Como se indicaba anteriormente usamos “import javax.swing.JOptionPane;” para agregar los mensajes Tales como los de inicio del juego.



Agregamos atributos que creemos necesarios para hacer cosas tales como:

- Los turnos de los jugadores
- El tablero del juego

```
import javax.swing.JOptionPane;
import java.util.ArrayList;

public class KalahGame {
    private Pit[] board;
    private boolean playerTurn;
```

Creamos el estado de inicio:

```
public KalahGame() {
    initializeGame();
}

public void initializeGame() {
    board = new Pit[14];
    for (int i = 0; i < 14; i++) {
        if (i == 6 || i == 13) {
            board[i] = new Pit(true, i < 7);
        } else {
            board[i] = new Pit(false, i < 7);
            board[i].putSeeds(4);
        }
    }
    organizeBoard();
    playerTurn = true;
    makeVisible();
    JOptionPane.showMessageDialog(null, "Juego iniciado. Turno del jugador N");
}
```

el cual se encarga de la creacion de elementos del board, de su visibilidad y el mensaje de inicio, aunque dentro de este usamos organizeBoard, para organizar el tablero de forma optima y coherente

El proceso es un recorrido en el cual modificamos las posiciones con moveTo, el cual definimos anteriormente en Pit

```
private void organizeBoard() {  
    int xStart = 200, yStart = 200, spacing = 95;  
    for (int i = 0; i < 6; i++) {  
        board[i].moveTo(xStart + (i * spacing), yStart);  
    }  
    board[6].moveTo(xStart + (6 * spacing), yStart);  
    for (int i = 7; i < 13; i++) {  
        board[i].moveTo(xStart + ((12 - i) * spacing), yStart + spacing);  
    }  
    board[13].moveTo(xStart - spacing, yStart);  
}
```

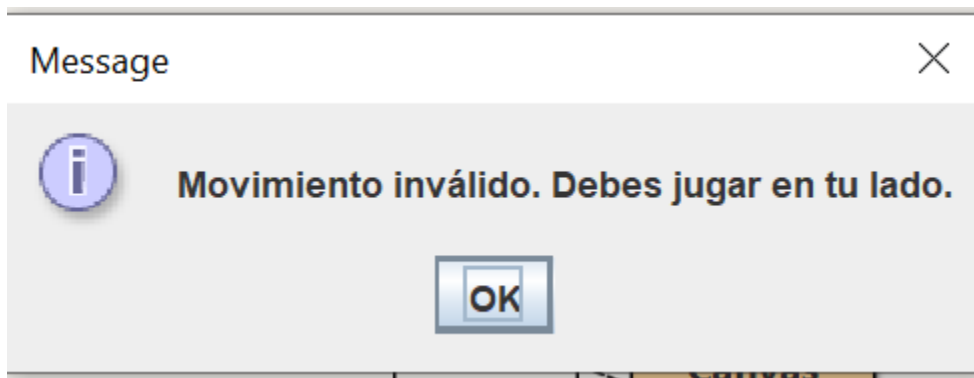
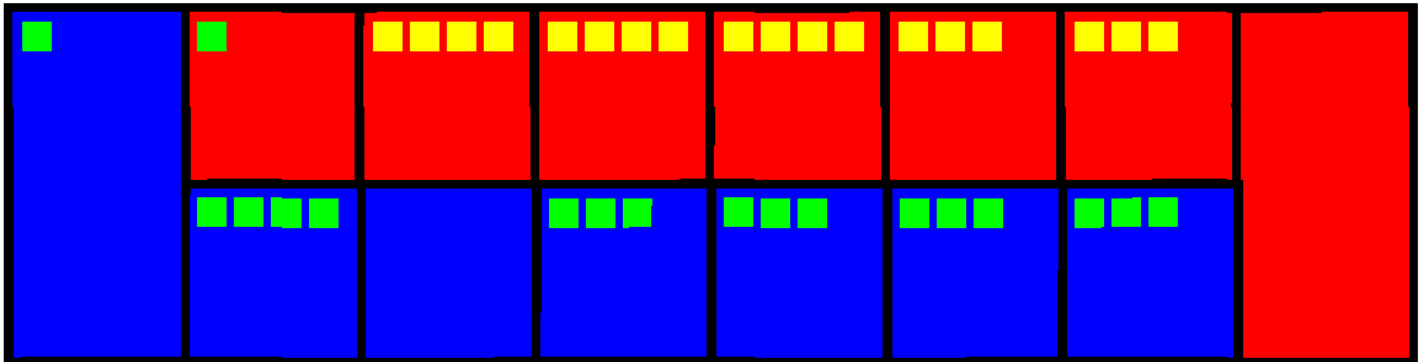
Como se puede observar las casillas de la mita del board son modificadas con for, teniendo en cuenta su indice y teniendo en cuenta una variable spacing para poder separar de manera justa los centros de los Rectangle, ademas los casos especiales como los costados del tablero se editaron directamente pues son los ubicados en la posicion 6 y 13.

Una vez creado, es reutilizado para reiniciar el juego, cuando sea deseado:

```
public void restartGame() {  
    initializeGame();  
}
```

Ahora nos ocupamos de los movimintos, para esto usamos:

```
public boolean makeMove(int pitIndex) {
    if ((playerTurn && (pitIndex < 0 || pitIndex > 5)) || (!playerTurn && (pitIndex < 7 || pitIndex > 12))) {
        JOptionPane.showMessageDialog(null, "Movimiento inválido. Debes jugar en tu lado.");
        return false;
    }
    int seeds = board[pitIndex].seeds();
    if (seeds == 0) {
        JOptionPane.showMessageDialog(null, "Mejor pideme dividir por cero. !No hay semillas en esta casa!. Busca otra.");
        return false;
    }
    ArrayList<String> seedColors = board[pitIndex].removeSeedsAndGetColors();
    int currentIndex = pitIndex;
    while (!seedColors.isEmpty()) {
        currentIndex = (currentIndex + 1) % 14;
        if ((playerTurn && currentIndex == 13) || (!playerTurn && currentIndex == 6)) {
            continue;
        }
        board[currentIndex].putSeed(seedColors.remove(0));
    }
    boolean extraTurn = (playerTurn && currentIndex == 6) || (!playerTurn && currentIndex == 13);
    if (!extraTurn && board[currentIndex].seeds() == 1 && ((playerTurn && currentIndex < 6) || (!playerTurn && currentIndex > 6 && currentIndex < 13))) {
        int oppositeIndex = 12 - currentIndex;
        int capturedSeeds = board[oppositeIndex].seeds();
        if (capturedSeeds > 0) {
            ArrayList<String> capturedColors = board[oppositeIndex].removeSeedsAndGetColors();
            int storeIndex = playerTurn ? 6 : 13;
            board[storeIndex].putSeeds(1);
            for (String color : capturedColors) {
                board[storeIndex].putSeed(color);
            }
            JOptionPane.showMessageDialog(null, "¡Captura realizada! semillas secuestradas =");
        }
    }
}
```



makeMove se encarga de todos los movimientos del juego, teniendo en cuenta el valor booleano sabe si le corresponde jugar al jugador Sur o Norte, así obedeciendo la condicional que respeta los índices de 0 a 5 para un jugador y de 7 a 12 para el segundo, además, de comprobar los movimientos espaciales

como turno extra o captura de semillas, una vez terminado el turno hace un check de victoria el cual definimos así:

```
public void checkWinCondition() {
    int northSeeds = 0, southSeeds = 0;
    for (int i = 0; i < 6; i++) northSeeds += board[i].seeds();
    for (int i = 7; i < 13; i++) southSeeds += board[i].seeds();

    if (northSeeds == 0 || southSeeds == 0) {
        for (int i = 0; i < 6; i++) {
            board[6].putSeeds(board[i].seeds());
            board[i].removeSeeds(board[i].seeds());
        }
        for (int i = 7; i < 13; i++) {
            board[13].putSeeds(board[i].seeds());
            board[i].removeSeeds(board[i].seeds());
        }

        int northStore = board[6].seeds();
        int southStore = board[13].seeds();
        String winner = northStore > southStore ? "N" : (northStore < southStore ? "S" : "Empate");
        JOptionPane.showMessageDialog(null, "Juego terminado. El dulce Ganador: " + winner);
        restartGame();
    }
}
```

El cual confirma la existencia de semillas, condición de cerrado del juego y también hace el conteo para poder seleccionar el ganador, o en su defecto dejar un empate, al finalizar se reinicia el juego.

Finalmente una manera de ver el estado del juego:

```
public void mostrarEstadoJuego() {
    StringBuilder estado = new StringBuilder("Estado actual del juego:\n\n");

    // Mostrar casas del sur (jugador sur)
    estado.append("Sur:\n");
    for (int i = 0; i < 6; i++) {
        estado.append("Casa ").append(i).append(": ").append(board[i].seeds()).append(" semillas\n");
    }
    estado.append("Almacén Sur (6): ").append(board[6].seeds()).append(" semillas\n\n");

    // Mostrar casas del norte (jugador norte)
    estado.append("Norte:\n");
    for (int i = 12; i > 6; i--) {
        estado.append("Casa ").append(i).append(": ").append(board[i].seeds()).append(" semillas\n");
    }
    estado.append("Almacén Norte (13): ").append(board[13].seeds()).append(" semillas");

    JOptionPane.showMessageDialog(null, estado.toString(), "Estado del Juego", JOptionPane.INFORMATION_MESSAGE);
}
```

con ciclos for recopila la informacion a fin de unirla y dar un reporte e igual que en su creacion los costados del juego se recopilan por indice, 6 y 13 para luego ser visto por el usuario.

Cabe resaltar que hicimos unos cambios en Pit a fin de lograr el juego (aunque son pequeños, son significativos)

G. De python a java

1.

Atrás

PREGUNTA 1

Sin responder aún

Se puntúa como 0 sobre 6,0

Marcar pregunta

Los prompts son útiles para aprender a hacer la transición de Python a Java

☐ a. En desacuerdo

☐ b. De acuerdo

☒ c. Totalmente de acuerdo

Quitar mi elección

PREGUNTA 2

Sin responder aún

Se puntúa como 0 sobre 6,0

Marcar pregunta

Las respuestas se presentan de forma clara

☐ a. En desacuerdo

☐ b. De acuerdo

☒ c. Totalmente de acuerdo

Quitar mi elección

PREGUNTA 3

Sin responder aún

Se puntúa como 0 sobre 6,0

Marcar pregunta

Los ejemplos seleccionados son apropiados para ilustrar los conceptos

☐ a. En desacuerdo

☐ b. De acuerdo

☒ c. Totalmente de acuerdo

Quitar mi elección

PREGUNTA 4

Sin responder aún

Se puntúa como 0 sobre 6,0

Marcar pregunta

Los ejemplos propuestos permiten ilustrar las respuestas

☐ a. En desacuerdo

☐ b. De acuerdo

☒ c. Totalmente de acuerdo

Quitar mi elección

PREGUNTA 5

Sin responder aún

Se puntúa como 0 sobre 6,0

Marcar pregunta

La estructura del formulario y de las preguntas es adecuada

☐ a. En desacuerdo

☐ b. De acuerdo

☒ c. Totalmente de acuerdo

Quitar mi elección

32

PREGUNTA 6

Sin responder aún

Se puntúa como 0 sobre 6.0

🚩 Marcar pregunta

El uso de IA generativa contiene elementos que lo hacen atractivo

- ☐ a. En desacuerdo
- ☐ b. De acuerdo
- ☒ c. Totalmente de acuerdo

[Quitar mi elección](#)

PREGUNTA 7

Sin responder aún

Se puntúa como 0 sobre 11.0

🚩 Marcar pregunta

Lo que no sabía y aprendí usando los prompts fue:

Lo que no sabía y aprendí usando los prompts fue a saber la diferencia de sintaxis entre java y python, conociendo mejor sus diferencias y similitudes entre programar en cada uno de estos lenguajes.



PREGUNTA 8

Sin responder aún

Se puntúa como 0 sobre 11.0

🚩 Marcar pregunta

Lo que más me gustó de los prompts fue:

Lo que mas me gustó fueron como los roles que tenia que tener en cada ejercicio ya que pues cada rol correspondía a un ejercicio diferente y también la forma de explicarlo me ayudo a entender mejor.

PREGUNTA 9

Sin responder aún

Se puntúa como 0 sobre 21.0

🚩 Marcar pregunta

Las sugerencias que haría para mejorar estos prompts son:

Abarcar mas tipos de situaciones creo que seria una de las cosas a mejorar .



PREGUNTA 10
Sin responder aún
Se puntúa como 0 sobre 21.0
🚩 Marcar pregunta

Los errores que encontré en los prompts son:

No, todo estaba en orden

Terminar intento...

COB_2025-1 / So2. Clases y objetos / Evaluación del video De Python a Java

EVALUACIÓN DEL VIDEO DE PYTHON A JAVA

Atrás

PREGUNTA 1
Sin responder aún
Se puntúa como 0 sobre 6.0
🚩 Marcar pregunta

Los prompts son útiles para aprender a hacer la transición de Python a Java

- ☐ a. En desacuerdo
☐ b. De acuerdo
☒ c. Totalmente de acuerdo


[Quitar mi elección](#)

PREGUNTA 2
Sin responder aún
Se puntúa como 0 sobre 6.0
🚩 Marcar pregunta

Las respuestas se presentan de forma clara

- ☐ a. En desacuerdo
☐ b. De acuerdo
☒ c. Totalmente de acuerdo

[Quitar mi elección](#)

 PREGUNTA 3
Sin responder aún
Se puntúa como 0 sobre 6.0
🚩 Marcar pregunta

Los ejemplos seleccionados son apropiados para ilustrar los conceptos

- ☐ a. En desacuerdo
☐ b. De acuerdo
☒ c. Totalmente de acuerdo

[Quitar mi elección](#)

PREGUNTA 4
Sin responder aún
Se puntúa como 0 sobre 6.0
🚩 Marcar pregunta

Los ejemplos propuestos permiten ilustrar las respuestas

- ☐ a. En desacuerdo
☐ b. De acuerdo
☒ c. Totalmente de acuerdo

[Quitar mi elección](#)

PREGUNTA 5
Sin responder aún
Se puntúa como 0 sobre 6.0
🚩 Marcar pregunta

La estructura del formulario y de las preguntas es adecuada

- ☐ a. En desacuerdo
☐ b. De acuerdo
☒ c. Totalmente de acuerdo

[Quitar mi elección](#)

PRECUNTA 6

Sin responder aún

Se puntúa como 0 sobre 6,0

🚩 Marcar pregunta

El uso de IA generativa contiene elementos que lo hacen atractivo

- ☐ a. En desacuerdo
- ☐ b. De acuerdo
- ☒ c. Totalmente de acuerdo

[Quitar mi elección](#)

PRECUNTA 7

Sin responder aún

Se puntúa como 0 sobre 11,0

🚩 Marcar pregunta

Lo que no sabía y aprendí usando los prompts fue:

Lo que no sabía y aprendí usando los prompts fue a saber la diferencia de sintaxis entre java y python, conociendo mejor sus diferencias y similitudes entre programar en cada uno de estos lenguajes.

PRECUNTA 8

Sin responder aún

Se puntúa como 0 sobre 11,0

🚩 Marcar pregunta

Lo que más me gustó de los prompts fue:

Lo que mas me gustó fueron como los roles que tenía que tener en cada ejercicio ya que pues cada rol correspondía a un ejercicio diferente y también la forma de explicarlo me ayudó a entender mejor.

PRECUNTA 9

Sin responder aún

Se puntúa como 0 sobre 21,0

🚩 Marcar pregunta

Las sugerencias que haría para mejorar estos prompts son:

Abarcar mas tipos de situaciones creo que seria una de las cosas a mejorar .

PREGUNTA 10

Sin responder aún

Se puntúa como 0 sobre 21,0

 Marcar pregunta

Los errores que encontré en los prompts son:

No, todo estaba en orden.

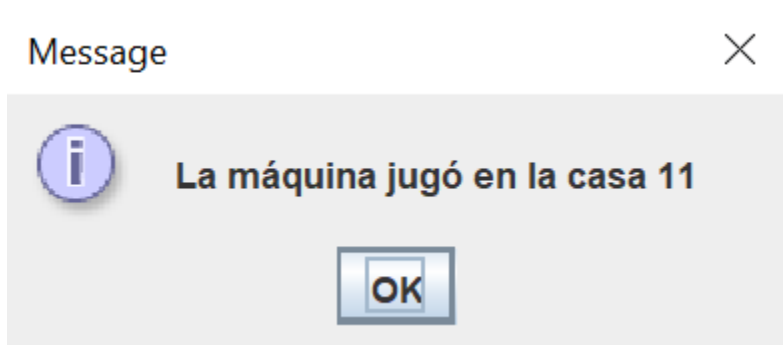
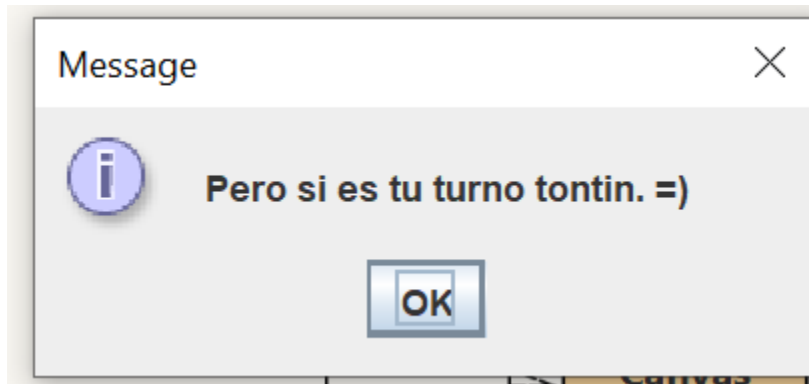
BONO**Movimiento de la maquina:**

```
public void machineMove() {  
    if (!playerTurn) {  
        ArrayList<Integer> validMoves = new ArrayList<>();  
        for (int i = 7; i < 13; i++) {  
            if (board[i].seeds() > 0) {  
                validMoves.add(i);  
            }  
        }  
        if (!validMoves.isEmpty()) {  
            int chosenMove = validMoves.get(new Random().nextInt(validMoves.size()));  
            makeMove(chosenMove);  
            JOptionPane.showMessageDialog(null, "La máquina jugó en la casa " + chosenMove);  
        }  
    }  
    else{  
        JOptionPane.showMessageDialog(null, "Pero si es tu turno tontin. =)");  
    }  
}
```

Utilizamos Random para generar el punto que la maquina usará para mover en el juego (se importó en la parte superior)

Se verifica que este en un turno Valido para mover, de lo contrario, se manda un mensaje de advertencia

Se selecciona de entre los valores validos, se hace el movimiento con makeMove y se genera un mensaje donde se indica la casa que fue utilizada.



Moviendo graficamente tambien el juego.

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/ Hombre)
Alrededor de 8 horas por persona
2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
Finalizado, el laboratorio se encuentra totalmente desarrollado.
3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?
Trabajo Rotativo, en donde uno da ideas el otro escribe y el cambio de puesto generó un ecosistema eficaz
4. ¿Cuál consideran fue el mayor logro? ¿Por qué?
El establecimiento de un orden para el desarrollo del trabajo en equipo, pues no habíamos trabajado juntos antes.
5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?
Ninguno, todo salió bien.
6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?
El Área de comunicación, como abordamos el trabajo y la buena disposición del tiempo.
7. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.
Para el tema del Bono y el posicionamiento de las seeds sin que cambiaran de color debido al tablero, usamos la IA, en cualquier otro tema, fue trabajo original nuestro.