Escuela Colombiana de Ingeniería Julio Garavito
Carrera / Semestre: Ing. Sistemas/ Séptimo Semestre
Materia: POOB
Título: Construcción. Clases y objetos.
Estudiantes: Santiago Hurtado Martínez, Santiago Andrés Arteaga

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO
VIGILADA MINEDUCACIÓN

PROGRAMACIÓN ORIENTADA A OBJETOS

Construcción. Clases y objetos.

Laboratorio ⅙

2024-2

OBJETIVOS

Desarrollar competencias básicas para:

1. Apropiar un paquete revisando: diagrama de clases, documentación y código.

2. Crear y manipular un objeto. Extender y crear una clase.

3. Entender el comportamiento básico de memoria en la programación OO.

4. Investigar clases y métodos en el API de java1

5. Utilizar el entorno de desarrollo de BlueJ

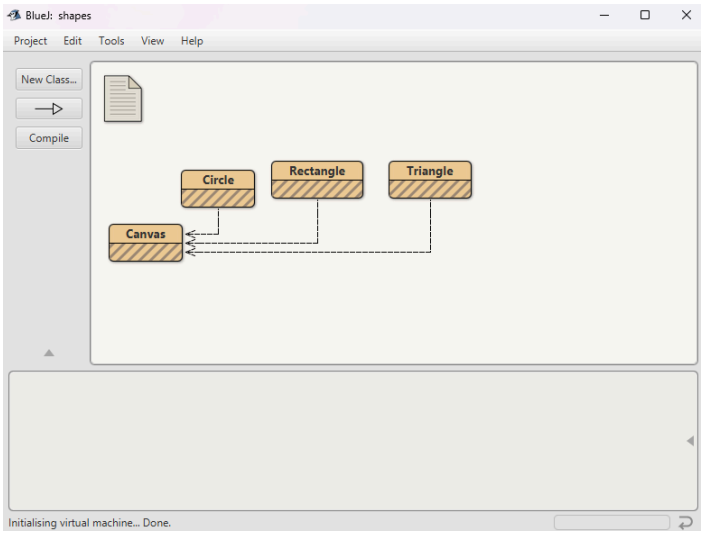
6. Vivenciar las prácticas XP : Planning The project is divided into iterations.

Coding All production code is pair programmed.

A. 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?

- **Tenemos las clases:**
  - **Canvas**
  - **Circle**
  - **Rectangle**
  - **Triangle.**

(b) ¿Qué relaciones existen entre ellas?

- Tenemos un tipo de relación de dependencia ya que nos indica que la clase canvas utiliza o depende de estas otras clases.

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?

- **Canvas, Circle, Rectangle y Triangle.**

(b) ¿Qué atributos tiene la clase Circle?

(c) ¿Cuántos métodos ofrece la clase Circle?

- Ofrece 12 métodos esta clase sin contar el constructor ya que este no se considera un método en el sentido tradicional (ya que no tiene un tipo de retorno)

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

1 Método public void changeSize(int newDiameter)

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

(a) ¿Cuántos atributos realmente tiene?

- **6 atributos**

  double PI=3.1416

  diameter

  xPosition

  yPosition

(b) ¿Cuáles atributos determinan su forma?

- **Solamente el atributo de diámetro.**

Con respecto a los métodos:

(c)¿Cuántos métodos tiene en total?

- **Tiene 14 métodos.**

(d) ¿Quiénes usan los métodos privados?

- **Canvas**

5. Comparando la documentación con el código

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

No se pueden ver los métodos privados.

(b) ¿por qué debe ser así?

- **Por seguridad y privacidad.**
- **Los atributos privados están ocultos del exterior para proteger la integridad del objeto.**

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

(a) ¿Qué significa que sea public?

- **Significa que otras clases y métodos puedes acceder a el si tiene alguna instancia a la clase.**

(b) ¿Qué significa que sea static?

- **Significa que pertenece a la clase en si misma.**

(c) ¿Qué significaría que fuera final?

- **significa que su valor no puede cambiar una vez que ha sido asignado.**

¿Debe serlo?

- **Si, ya que pi es una constante universal y su valor que determinan ahí es correcto.**

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

- **Debería ser flotante, ya que un flotante solo admite 6 dígitos y es lo que se utiliza en el código**.

(e) Actualícenlo.

```java
public class Circle{

    public static final float PI= 3.1416f;

    private int diameter;
    private int xPosition;
    private int yPosition;
    private String color;
    private boolean isVisible;
```

7. En el código de la clase Rectangle revisen el detalle del tipo del atributo width

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

- **Que el valor que se espera tiene que ser entero.**

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

- **el rectángulo más grande que podríamos construir sería de 127 unidades por lado.**

(c) y ¿si fuera long?

- **tendríamos un reptando de lado 2^(63)-1**

d) ¿qué restricción adicional deberían tener este atributo?

- **Que no se acepten valores negativos.**

(e) Refactoricen el código considerando **(d).**

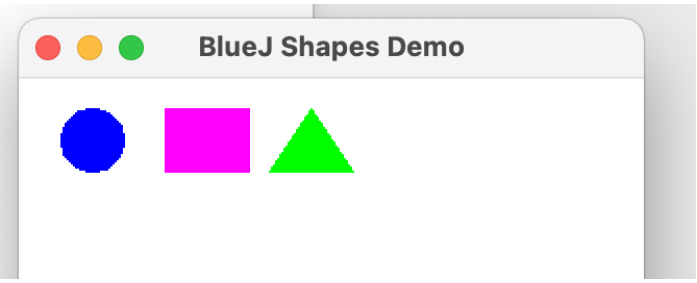
```java
/**
 * Create a new rectangle at default position with default color.
 */
public Rectangle(){
    height = 30;
    width = 40;
    xPosition = 70;
    yPosition = 15;
    color = "magenta";
    isVisible = false;
}

public int getWidth() {
    return width;
}

public void setWidth(int width) {
    if (width > 0) {
        this.width = width;
    } else {
        throw new IllegalArgumentException("Width must be positive.");
    }
}
```

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

- **Diríamos que el propósito de “shapes” es poder entender el concepto de clases y objetos para poderlos manipular y darles una representación o función según lo que queramos representar.**

B. Manipulando objetos. Usando un objeto.

1. Creen un objeto de cada una de las clases que lo permitan3



(a) ¿Cuántas clases hay?

Hay 4 clases

(b) ¿Cuántos objetos crearon?

Creamos 4 objetos: circulo, cuadrado, rectángulo y un canvas que los contiene.

(c) ¿Quién se crea de forma diferente? ¿Por qué?

Canvas es la clase que se “crea” diferente, pues está lista para contener a los demás y presenta mayor “dificultad”.

2. Inspeccionen los creadores de cada una de las clases. (**a)** ¿Cuál es la principal diferencia entre ellos?

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

```
public Rectangle(){
    height = 30;
    width = 40;
    xPosition = 70;
    yPosition = 15;
    color = "magenta";
    isVisible = false;
}
```

```
public Circle(){
    diameter = 30;
    xPosition = 20;
    yPosition = 15;
    color = "blue";
    isVisible = false;
}
```

```
private Canvas(String title, int width, int height, Color bgColour){
    frame = new JFrame();
    canvas = new CanvasPane();
    frame.setContentPane(canvas);
    frame.setTitle(title);
    canvas.setPreferredSize(new Dimension(width, height));
    backgroundColour = bgColour;
    frame.pack();
    objects = new ArrayList <Object>();
    shapes = new HashMap <Object,ShapeDescription>();
}
```

- **El color en los constructores de las figuras es un objeto String, pero en el canvas el color que define al background es un objeto Color.**

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

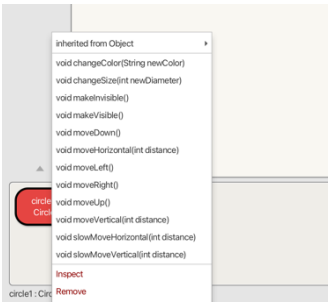
- **La clase Canvas contiene a todas las demás y está diseñada para mostrar en pantalla lo que se genera mediante las otras tres.**

3. Inspeccionen el estado del objeto :Circle4,

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

- **diameter = 30;**
- **xPosition = 20;**
- **yPosition = 15;**
- **color = "blue";**
- **isVisible = false;**

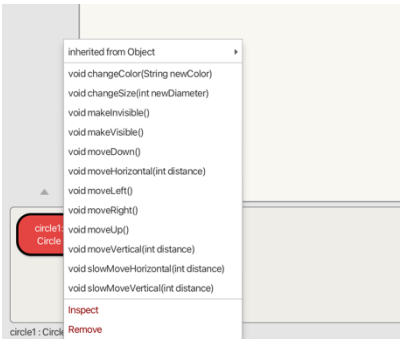
(b) Capturen la pantalla.




```
public Circle(){
    diameter = 30;
    xPosition = 20;
    yPosition = 15;
    color = "blue";
    isVisible = false;
}
```

4. Inspeccionen el comportamiento que ofrece el objeto :Circle5.

(**a)** Capturen la pantalla.



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

- **Porque algunos son privados**

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

(**a)** ¿Cuántas y cuáles clases se necesitan?

- **Solamente una clase “rectangle”**

(**b)**¿Cuántos objetos se usan en total?

- **Utilice cinco objetos de la clase ”rectangle”**

(d) Capturen la pantalla.

(e) Incluyan el logo original.

C. Manipulando objetos. Analizando y escribiendo código.

[En lab01.doc]

```
Rectangle rod;
Rectangle bigDisk;
Rectangle smallDisk;
//1
rod=new Rectangle();
bigDisk = new Rectangle();
smallDisk = bigDisk;
//2
rod.changeColor("black");
rod.changeSize(100,10);
rod.makeVisible();
//3
```

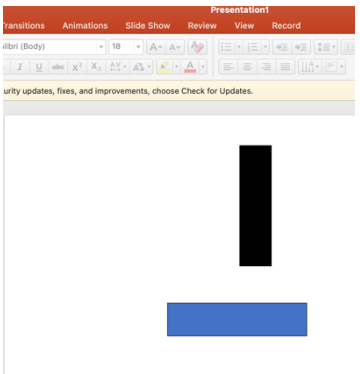
```
bigDisk.moveHorizontal(-5);
bigDisk.moveVertical(80);
smallDisk.moveHorizontal(-20);
smallDisk.moveVertical(90);
//4
bigDisk.changeColor("red");
bigDisk.changeSize(10,20);
smallDisk.changeColor("blue");
smallDisk.changeSize(10,50);
//5
rod.makeVisible();
bigDisk.makeVisible();
smallDisk.makeVisible();
//6
```

1. Lean el código anterior.

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

Lo que se puede apreciar en el código es la creación de 3 objetos donde se les asigna una posición, tamaño, color y visualización.

(b) Píntenla.



2. Habiliten la ventana de código en línea 6, escriban el código. Para cada punto señalado indiquen:

(a) ¿Cuántas variables existen?

Existen tres variables.

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

Existen dos objetos

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

Uno Negro y otro Azul

(d) ¿Cuántos objetos se ven?

Dos objetos.

(e) Expliquen sus respuestas.

Dado que smallDisk está apuntando al mismo objeto es decir bigDisk, todos los cambios que hagamos se van a ver referenciados al mismo objeto, no importa que se llamen diferente solamente están apuntando al mismo objeto.

(f) Capturen la pantalla.

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

(a) ¿son iguales?

- **Son casi iguales.**

(b)¿por qué?

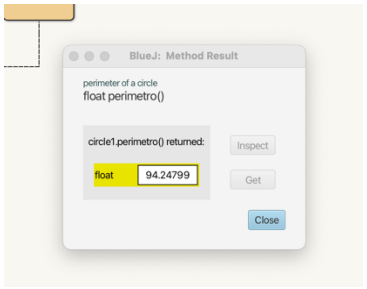
- **Seguimos los mismos pasos que se indican en el código pero tanto las dimensiones como las posiciones no son exactas.**

D. Extendiendo una clase. Circle.

[En lab01.doc y *.java]

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

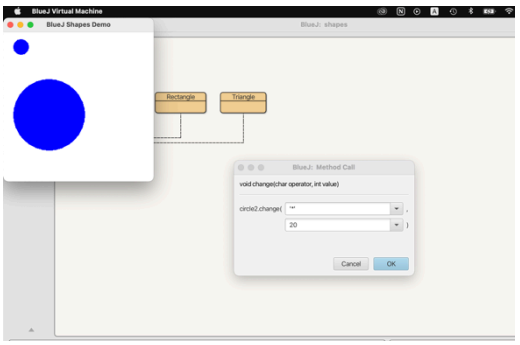
```java
public Circle(){
    diameter = 30;
    xPosition = 20;
    yPosition = 15;
    color = "blue";
    isVisible = false;
}

/**
 * perimeter of a circle
 */
public float perimetro() {
    return PI * diameter;
}
```



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

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

¡Pruébenlo! Capturen dos pantallas.




```java
public double area() {
    double radius = diameter / 2.0;
    return PI * radius * radius;
}

// Método change para modificar el área del círculo
public void change(char operator, int value) {
    double currentArea = area();
    double newArea = currentArea;

    switch (operator) {
        case '+':
            newArea = currentArea + value;
            break;
        case '-':
            newArea = currentArea - value;
            break;
        case '*':
            newArea = currentArea * value;
            break;
        case '/':
```

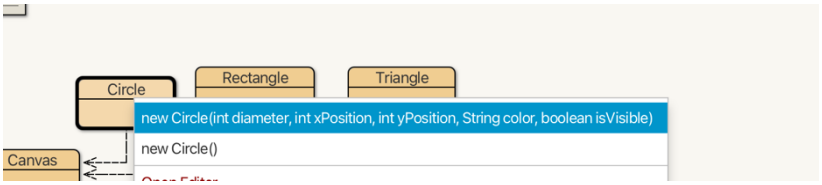
3. Desarrollen en Circle el método roll(times:int, maxDistance: int)

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

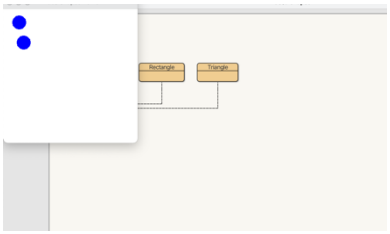
```java
public void roll(int times, int maxDistance) {
    Random random = new Random();

    for (int i = 0; i < times; i++) {
        int distance = random.nextInt(maxDistance);
        int angle = random.nextInt(91); // Ángulo entre 0 y 90 grados

        double radians = Math.toRadians(angle);
        int deltaX = (int) (distance * Math.cos(radians));
        int deltaY = (int) (distance * Math.sin(radians));

        xPosition += deltaX;
        yPosition += deltaY;

        draw();
    }
}
```

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

```java
/**
 * constructor 2
 */

public Circle(int diameter, int xPosition, int yPosition, String color, bool
    this.diameter = diameter;
    this.xPosition = xPosition;
    this.yPosition = yPosition;
    this.color = color;
    this.isVisible = isVisible;

    if (isVisible) {
        draw();  // Dibuja el círculo si es visible al crearlo
    }
}
```

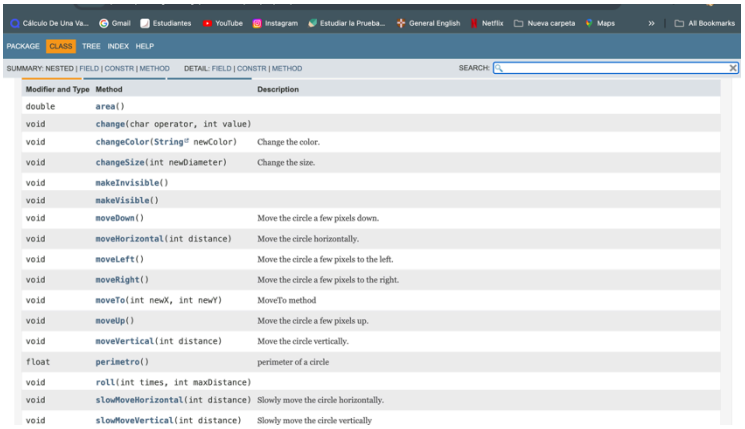


5. Propongan un nuevo método para esta clase. Desarrollen y prueban el método.



```java
/**
 * MoveTo method
 */
public void moveTo(int newX, int newY) {
    erase();
    xPosition = newX;
    yPosition = newY;
    draw();
}
```

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



Modifier and Type	Method	Description
double	area()	
void	change(char operator, int value)	
void	changeColor(String newColor)	Change the color.
void	changeSize(int newDiameter)	Change the size.
void	makeInvisible()	
void	makeVisible()	
void	moveDown()	Move the circle a few pixels down.
void	moveHorizontal(int distance)	Move the circle horizontally.
void	moveLeft()	Move the circle a few pixels to the left.
void	moveRight()	Move the circle a few pixels to the right.
void	moveTo(int newX, int newY)	MoveTo method
void	moveUp()	Move the circle a few pixels up.
void	moveVertical(int distance)	Move the circle vertically.
float	perimetro()	perimeter of a circle
void	roll(int times, int maxDistance)	
void	slowMoveHorizontal(int distance)	Slowly move the circle horizontally.
void	slowMoveVertical(int distance)	Slowly move the circle vertically

E. Usando un paquete. shapes

[En lab01.doc y *.java]

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

Las tres figuras deben ofrecer los siguientes métodos.

Tower	Mini-ciclo: 1 _(disks:int):Tower makeVisible() makeInvisible() **Mini-ciclo: 2** moveTo changeColors **Mini-ciclo: 3** push pop
+ _(disks : int) : Tower + changeColors() : void + makeVisible() : void + makeInvisible() : void + moveTo(x : int, y : int) : void + pop() : boolean + push(disk : int) : void	

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

```
private Rectangle base;
private Rectangle pole;
private Map<Rectangle, Position> removedDiscs;
private ArrayList<Rectangle> discs;
private boolean isVisible;
```

2. Desarrollen la clase considerando los 3 mini-ciclos. Al final de cada mini[1]ciclo realicen dos pruebas indicando su propósito. Capturen las pantallas relevantes.

```java
public void makeVisible() {
    base.makeVisible();
    pole.makeVisible();

    // Hacer visibles todos los discos
    for (Rectangle disc : discs) {
        disc.makeVisible();
    }

    isVisible = true;
}

public void makeInvisible() {
    base.makeInvisible();
    pole.makeInvisible();
    for (Rectangle disc : discs) {
        disc.makeInvisible();
    }
    isVisible = false;
}

public void moveTo(int x, int y) {
    base.moveTo(x, y + 100);
    pole.moveTo(x + 65, y);
    updateDiscPositions();
}
```

```java
public Rectangle pop() {
    if (discs.isEmpty()) {
        return null; // No hay discos para sacar
    }
    Rectangle topDisc = discs.remove(discs.size() - 1); // Elimina el disco de la lista
    topDisc.makeInvisible(); // Hacer el disco invisible
    // Guarda la posición del disco eliminado
    removedDiscs.put(topDisc, new Position(topDisc.getX(), topDisc.getY()));
    return topDisc; // Devolver el disco eliminado
}

public void push() {
    if (removedDiscs.isEmpty()) {
        return; // No hay discos para agregar
    }

    // Recupera el disco eliminado más recientemente
    Rectangle discToPush = null;
    for (Rectangle disc : removedDiscs.keySet()) {
        discToPush = disc;
        break;
    }

    // Recupera la posición original del disco
    Position position = removedDiscs.remove(discToPush);
    if (position != null) {
        // Coloca el disco en la posición original
        discToPush.moveTo(position.xPosition, position.yPosition);
```

compiled - no syntax errors saved

7 Consulte la clase Math del API java

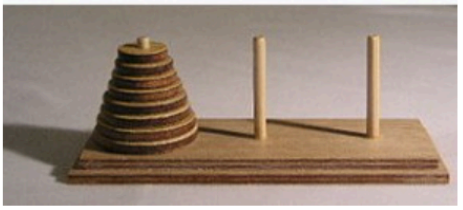
F. De python a java prompts

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

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

El prompts de Java no fue tan intuitivo a la hora de explicar que debía leerse desde una IA generativa.
Además de eso las explicaciones fueron bastante claras

TORRES DE HANOI

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

G. Definiendo y creando una nueva clase. TowerOfHanoi.

[En lab01.doc. TowerOfHanoi.java]

[En lab01.doc. TowerOfHanoi.java]

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

Requisitos funcionales

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

Requisitos de interfaz

- Las torres se identifican por su posición [1,2,3] de izquierda a derecha.
- En caso que no sea posible realizar una de las acciones, debe generar un mensaje de error. Use JoptionPane.

JoptionPane.

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

vamos a crear la clase TowerOfHanoi y los métodos que planificamos utilizar son:

makeVisible

makeInvisible

pop

push

moveDisc

2. Planifiquen la construcción considerando algunos mini-ciclos.

```
+ _(disks : int) : Tower
+ changeColors() : void
+ makeVisible() : void
+ makeInvisible() : void
+ moveTo(x : int, y : int) : void
+ pop() : boolean
+ push(disk : int) : void
```

3. Implementan la clase . Al final de cada mini-ciclo realicen una prueba indicando su

ciclo1

makeVisible()

makeInvisible()

ciclo2

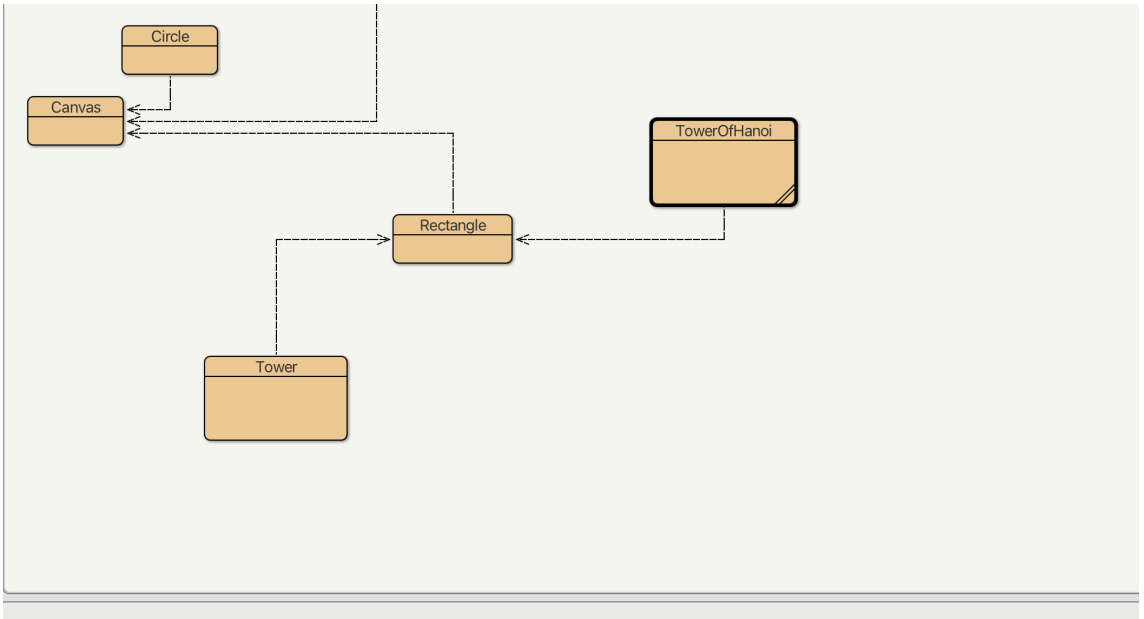
pop()

push()

ciclo3

moveDisc()

propósito. Capturen las pantallas relevantes.



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

shapes. Expliquen.

H. De python a java video

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

Completar el algoritmo de búsqueda binaria en Java no fue tan complicado una vez conociendo bien la sintaxis de Java

RETROSPECTIVA

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

Hombre)

Santiago Arteaga 10 horas

Santiago Hurtado 10 horas

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

9/10 ya que en el punto g hubo un método que no nos funcionó.

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

Todas fueron útiles para la implementación del trabajo. Porque sin esas prácticas no habríamos podido desarrollar algunos puntos como lo fueron el c ,d o e

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

hacer los puntos E y G ya que pudimos manipular objetos y crear clases.

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

el mayor problema fue haber trabajado con arraylist siendo que existen hashmap, solamente investigamos un poco mas.

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

resultados?

dimos soluciones coherentes al problema e hicimos investigaciones de forma dinámica para llegar a las soluciones más óptimas.

REFERENCIAS

No olviden incluir las referencias.

https://es.stackoverflow.com/questions/233793/como-iterar-un-arraylist-que-esta-dentro-de-un-hashmap

https://jnjsite.com/java-diferencias-entre-arraylist-y-hashmap/

https://www.programarya.com/Cursos/Java/Objetos-y-Clases

https://www.programarya.com/Cursos/Java/Funciones

RETROSPECTIVA

El taller proporcionó una comprensión integral de los principios de la Programación Orientada a Objetos y cómo aplicarlos a problemas prácticos como la Torre de Hanoi. La implementación en Java permitió a los participantes ver cómo estos conceptos se traducen en código funcional, ayudando a fortalecer su comprensión de la POO y la resolución de problemas.