

## **Laboratorio 01**

### **Integrantes:**

**Sebastián Cardona Parra**

**Diego Alexander Cardenas Beltrán**

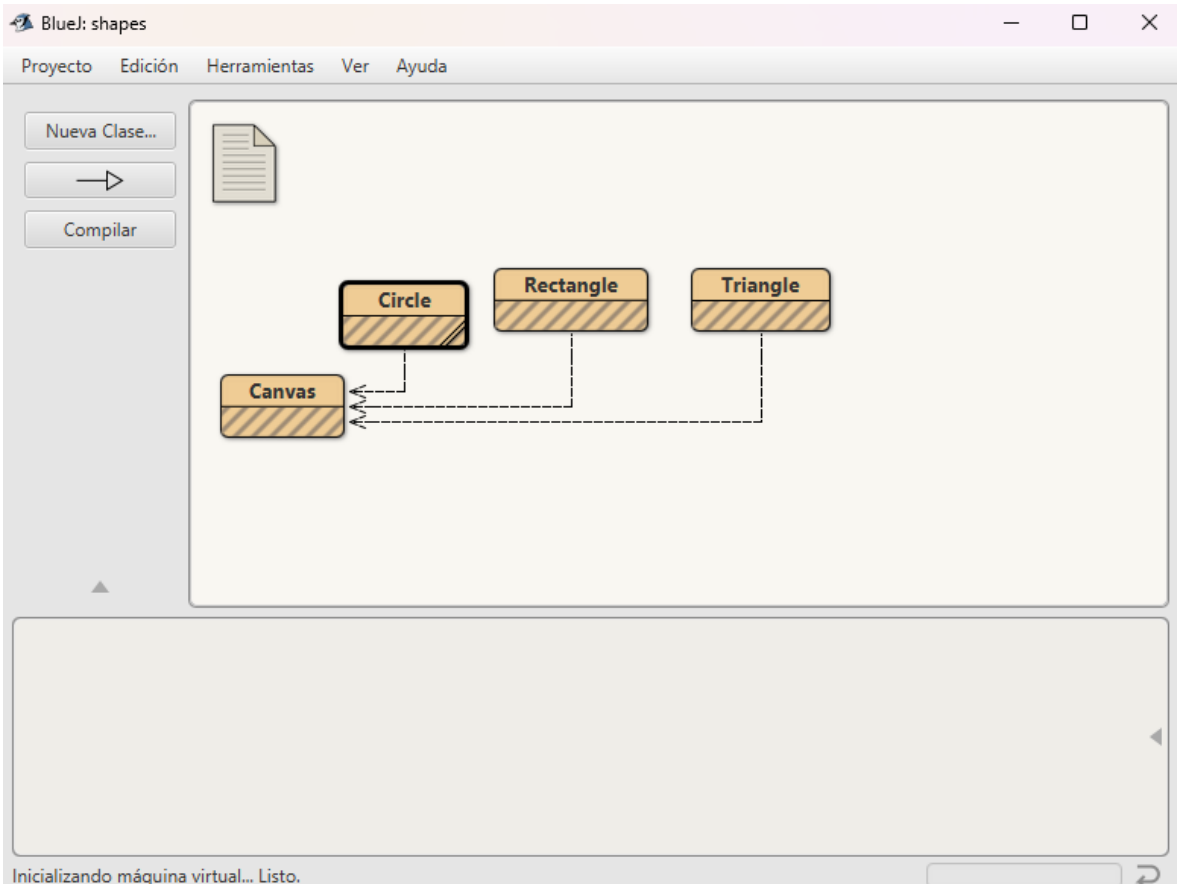
**Universidad Escuela Colombiana de Ingeniería Julio Garavito**  
**Programación Orientada a Objetos**  
**María Irma Rozo**

**01/02/2024**

## SHAPES

### 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 ábralo en BlueJ. Capturen la pantalla.



2. El **diagrama de clases** permite visualizar las clases de un artefacto software y las relaciones entre ellas. Considerando el diagrama de clases de “shapes” (a) ¿Qué clases ofrece? (b) ¿Qué relaciones existen entre ellas?
  - a) Ofrece 4 clases, Canvas, Circle, Rectangle, Triangle
  - b) Las relaciones que hay es que, circle, Rectangle, Triangle están relacionadas a Canvas para pintarse y visualizarse
3. La **documentación**<sup>2</sup> presenta las clases del proyecto y, en este caso, la especificación de sus componentes públicos. De acuerdo con la documentación generada: (a) ¿Qué clases tiene el paquete `shapes`? (b) ¿Qué atributos tiene la clase `Rectangle`? (c) ¿Cuántos métodos ofrece la clase `Rectangle`? (d) ¿Cuáles métodos ofrece la clase `Rectangle` para que la figura cambie su tamaño (incluya sólo el nombre)?
  - a) Canvas, Circle, Rectangle and Triangle
  - b) No se mencionan atributos de Rectangle públicos, además del atributo `edges`
  - c) Ofrece 12 Métodos y 1 constructor
  - d) `changeSize`

4. En el **código** de cada clase está el detalle de la implementación. Revisen el código de la clase `Rectangle`. Con respecto a los atributos: (a) ¿Cuántos atributos realmente tiene? (b) ¿Cuáles atributos determinan su forma?. Con respecto a los métodos: (c) ¿Cuántos métodos tiene en total? (d) ¿Quiénes usan los métodos privados?
- a) Tiene 7 atributos
  - b) EDGES, height, width, (color)
  - c) 14 métodos y 1 constructor
  - d) Los métodos privados son draw and erase, el cual se usa en *makeVisible*, *makeInvisible*, *moveHorizontal*, *moveVertical*, *slowMoveHorizontal*, *slowMoveVertical*, *changeSize*, *changeColor*.
5. Comparando la **documentación** con el **código** (a) ¿Qué no se ve en la documentación? (b) ¿por qué debe ser así?
- a) No aparecen los atributos y métodos privados.
  - b) Debido a que estos mismos pertenecen a la clase `Rectangle`, y no tienen acceso a otras clases, además cabe mencionar que la documentación es de uso para el usuario, por lo cual, no es buena práctica mostrar los atributos o métodos privados, ya que igualmente no podrá usarlos.
6. En el código de la clase `Rectangle`, revise el atributo `EDGES` (a) ¿Qué significa que sea `public`? (b) ¿Qué significa que sea `static`? (c) ¿Qué significaría que fuera `final`? ¿Debe serlo? (d) ¿De qué tipo de dato debería ser (byte, short, int, long)? ¿Por qué? (e) Actualícenlo.
- a) Que cualquier otra clase y/o instancia puede acceder a él y lo puede modificar o consultar.
  - b) Significa que todos los objetos de esta clase van a tener este mismo valor sin importar el caso, es decir, es única para todas las instancias de dicha clase.
  - c) Que sería una constante, en ese caso al asignarle el valor dicho atributo, ya no se puede modificar de ninguna manera, en este caso sí, pues un rectángulo siempre tendrá 4 bordes sin importar que.
  - d) Debería ser de tipo byte, ya que este atributo usará un número pequeño y único, por lo cual, si se usa byte, se garantiza buen uso de memoria.
  - e)

```
public class Rectangle{  
  
    public static final byte EDGES = 4;  
  
    private int height;  
    private int width;  
    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 indicando al decir que es `int`? (b) Si `width` fuera `byte`, ¿cuál sería el rectángulo más grande posible? (c) y ¿si fuera `long`? d) ¿qué restricción adicional deberían tener este atributo? (e) Refactoricen el código considerando (d).
- a) Que el valor numérico entero que puede tener esta entre -2,147,483,648 a 2,147,483,647.  
b) El máximo que podría tocar es 127 de `width`.  
c) Si fuera `long` tendría un tamaño entre -9,223,372,036,854,775,808 a 9,223,372,036,854,775,807.  
d) Que el número fuera positivo y si o si entero.  
e)

```
public class Rectangle{

    public static final byte EDGES = 4;

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

    /**
     * Change the size to the new size
     * @param newHeight the new height in pixels. newHeight must be >=0.
     * @param newWidth the new width in pixels. newWidth must be >=0.
     */
    public void changeSize(int newHeight, int newWidth) {
        erase();
        height = newHeight;
        if(newWidth>0){
            width = newWidth;
            draw();
        }else{
            System.out.println("No puede ser negativo");
        }
    }
}
```

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

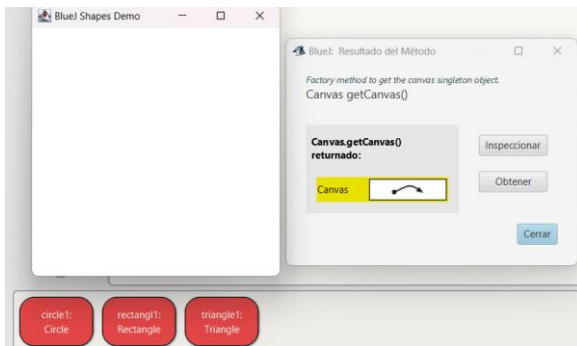
Su propósito es demostrar algunas características de objetos y clases como su estructura, esto por medio de un entorno donde se puedan realizar distintas figuras geométricas, modificando varios de sus atributos y dibujándolos

## B. Manipulando objetos. Usando un objeto.

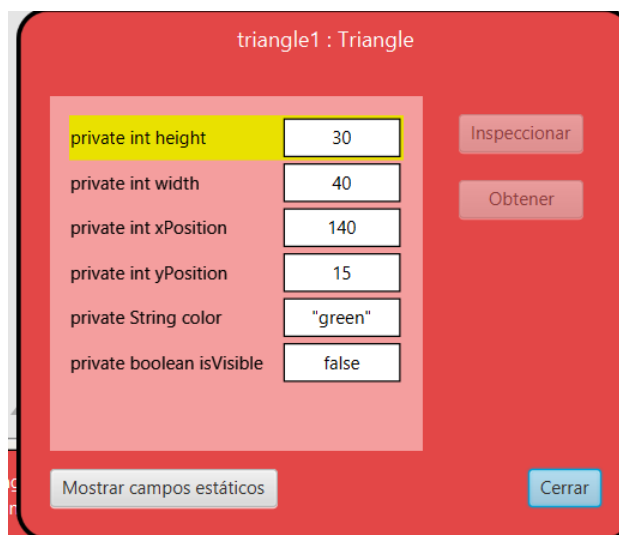
[En lab01.doc]

1. Creen un objeto de cada una de las clases que lo permitan<sup>3</sup>. (a) ¿Cuántas clases hay? (b) ¿Cuántos objetos crearon? (c) ¿Quién se crea de forma diferente? ¿Por qué?

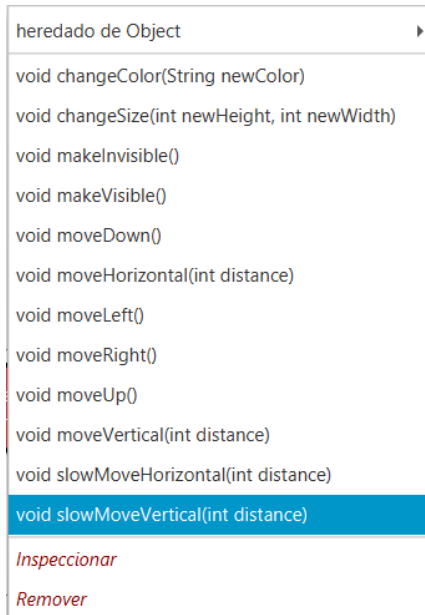
- a) Hay en total 4 clases
- b) Cree 4 objetos
- c) Se crea de manera diferente La clase Canvas, debido a que esta abre una interfaz, pero si es un objeto, aunque no lo podemos manipular directamente, sino por medio de las figuras creadas en este.



2. Inspeccionen los creadores de cada una de las clases. (a) ¿Cuál es la principal diferencia entre ellos? (b) ¿Qué se busca con la clase que tiene el creador diferente?
  - a) El creador de Canvas es privado y es accesible desde dentro de la clase, por eso, otro método lo llama "getCanvas()". Los demás tienen constructor público, pueden ser creador por otras clases.
  - b) Que en él se ubiquen todos los objetos creados de Circle, Triangle, Rectangle.
3. Inspeccionen el **estado** del objeto :Triangle<sup>4</sup>. (a) ¿Cuáles son los valores de inicio de todos sus atributos? (b) Capturen la pantalla.



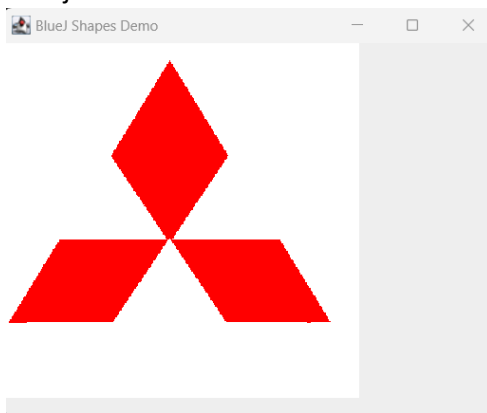
- a)
4. Inspeccionen el **comportamiento** que ofrece el objeto :Triangle<sup>5</sup>. (a) Capturen la pantalla. (b) ¿Por qué no aparecen todos los que están en el código?



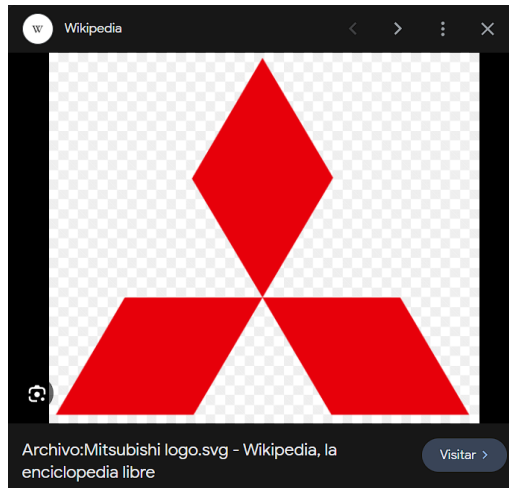
- a)  
b) Debido a solo muestra los métodos públicos.

5. Construyan, con “shapes” sin escribir código, una propuesta de la imagen del logo de su marca de carro favorita. (a) ¿Cuántas y cuáles clases se necesitan? (b) ¿Cuántos objetos se usan en total? (c) Capturen la pantalla. (d) Incluyan el logo original.

- a) Solo usamos dos clases, El Canvas y Triangle.  
b) 5 Objetos.



c)



d)

## C. Manipulando objetos. Analizando y escribiendo código.

[En lab01.doc]

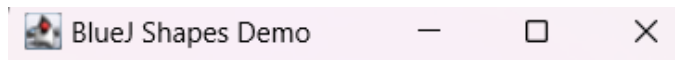
```
Rectangle face;
Circle pointOne;
Circle pointTwo;
//1
face=new Rectangle();
pointOne = new Circle();
pointTwo = new Circle();
//2
face.changeColor("black");
face.changeSize(100,100);
face.makeVisible();
//3
```

```
pointOne.changeColor("red");
pointOne.changeSize(10);
pointTwo=pointOne;
//4
pointOne.moveHorizontal(70);
pointOne.moveVertical(20);
pointTwo.moveHorizontal(120);
pointTwo.moveVertical(70);
//5
pointOne.makeVisible();
pointTwo.makeVisible();
//6
```

1. Lean el código anterior. (a) ¿cuál creen que es la figura resultante? (b) Píntenla.

a) Creemos que es un dado, debido a que nos imaginamos en nuestra cabeza el paso a paso, haciendo esto nos dimos cuenta que los círculos pequeños posiblemente estén dentro del cuadrado negro y estén en sus esquinas opuestas.

b)



2. Habiliten la ventana de código en línea<sup>6</sup>, escriban el código. Para cada punto señalado indiquen: (a) ¿cuántas variables existen? (b) ¿cuántos objetos existen? (no cuenten ni los objetos `String` ni el objeto `Canvas`) (c) ¿qué color tiene cada uno de ellos? (d) ¿cuántos objetos se ven? Al final, (e) Expliquen sus respuestas. (f) Capturen la pantalla.

- a) 3 variable.
- b) 3 Objetos.
- c) Fase tiene el color "black", y pointOne tiene el color "red" y el pointTwo tiene color azul.
- d) Solo se ven 2, el pointOne y el face.
- e) Todo iba bien, hasta que se puso la línea de código "*pointTwo = pointOne;*", esto pues al ser el igual un operador de asignación y siendo ambos puntos de la misma clase, lo que en realidad se hizo fue que ahora pointTwo fuera literalmente pointOne, es decir como si el PointOne (objeto), tuviera dos nombres y todo lo que haga a cualquiera de los dos puntos, siempre me afectará a pointOne y el original pointTwo "dejó" de existir después de la asignación.



- f) *... (text obscured) ...*
3. Compare figura pintada en 1. con la figura capturada en 2. , (a) ¿son iguales? (b) ¿por qué?
- a) No son para nada iguales.
  - b) Una vez que hice la asignación todo lo que le hiciera a pointOne o a pointTwo, siempre me iba a alterar pointOne por la explicación anterior, por lo que el verdadero pointTwo, nunca cambio de tamaño, de color, de posición y nunca se hizo visible porque en palabras coloquiales se quedó sin nombre para referirse a él.

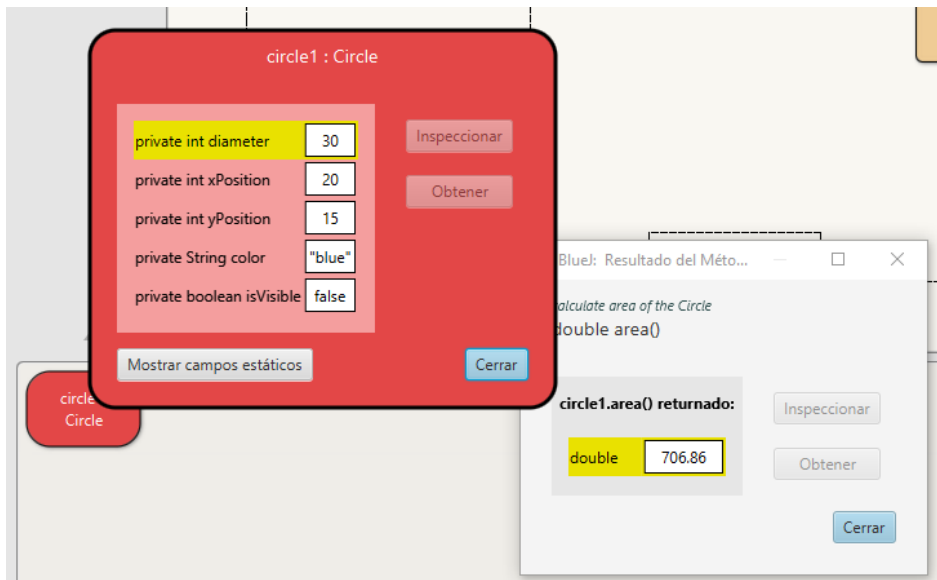
#### D. Extendiendo una clase. Rectangle.

[En lab01.doc y \*.java]

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

```
/**
 * calculate area of the Circle
 */
public double area(){
    return (Math.pow((diameter/2),2)*PI);
}
```






2. Desarrollen en `Circle` el método `duplicate()` (duplica el área del círculo) . ¡Pruébenlo! Capturen dos pantallas.

```
/**
 * Duplicate the diameter of the Circle
 */
public void duplicate(){
    erase();
    int newDiameter = diameter;
    diameter = 2*newDiameter;
    draw();
}
```

Antiguo

 Blue Shapes Demo



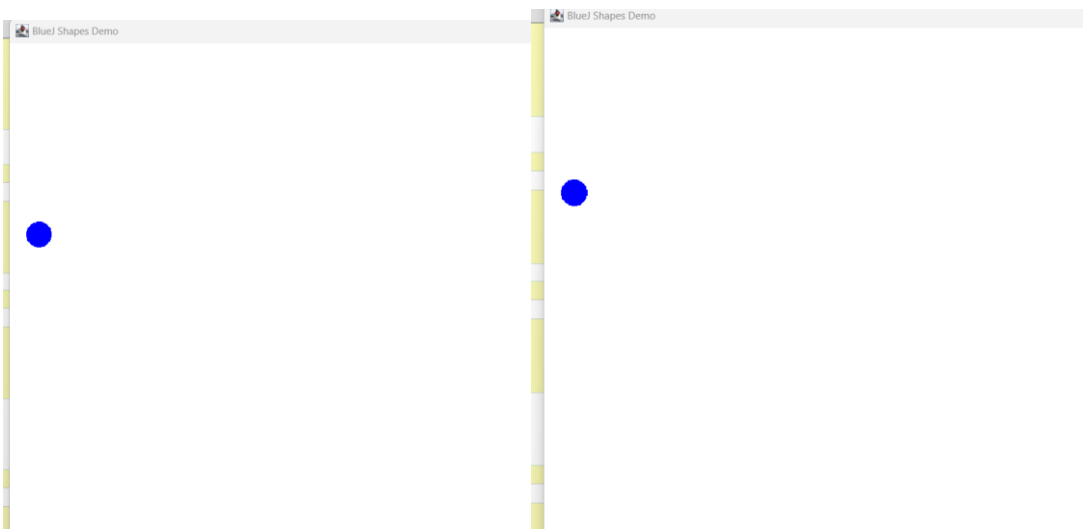
Nuevo

### BlueJ Shapes Demo



3. Desarrollen en `Circle` el método `bounce(times:int, height:int)` (salta el número de veces indicado en `times`. Las alturas de los saltos se seleccionan al azar con valores menores de `height` <sup>7</sup>). ¡Pruébenlo! Capturen dos pantallas.

```
/**
 * Genera saltos de una pelota en un rango aleatorio.
 * @param Cantidad de saltos que quiere dar
 * @param Altura maxima que recibe
 */
public void bounce(int times, int height){
    for(int i = 0; i<times;i++){
        int newaltura = (int)(Math.random()*height);
        slowMoveVertical(-newaltura);
        slowMoveVertical(newaltura);
    }
}
```



4. Desarrollen en `Circle` un nuevo creador que dada una posición, crea círculos negros de radio 50. ¡Pruébenlo! Capturen una pantalla.

```
public Circle(int xPos, int yPos){
    diameter = 100;
    xPosition = xPos;
    yPosition = yPos;
    color = "black";
    isVisible = false;
}
```



- de radio 500p, haciendo capturen una pantalla.
5. Propongan un nuevo método para esta clase. Desarrollen y prueben el método.

```
/**
 * Retorna el perimetro del circulo
 */
public double pelimeter(){
    return PI*(diameter);
}
```

BlueJ: Resultado del Método

Retorna el perimetro del circulo  
double pelimeter()

circle1.pelimeter() retornado:

double 314.15999999999997

Inspeccionar

Obtener

Cerrar

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

#### Constructor Summary

##### Constructors

Constructor	Description
-------------	-------------

Circle ()	
-----------	--

Circle (int xPos, int yPos)	
-----------------------------	--

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
double	<code>area()</code>	calculate area of the Circle
void	<code>bounce(int times, int height)</code>	Genera saltos de una pelota en un rango aleatorio.
void	<code>changeColor(String? newColor)</code>	Change the color.
void	<code>changeSize(int newDiameter)</code>	Change the size.
void	<code>duplicate()</code>	Duplicate the diameter of the Circle
void	<code>makeInvisible()</code>	
void	<code>makeVisible()</code>	
void	<code>moveDown()</code>	Move the circle a few pixels down.
void	<code>moveHorizontal(int distance)</code>	Move the circle horizontally.
void	<code>moveLeft()</code>	Move the circle a few pixels to the left.
void	<code>moveRight()</code>	Move the circle a few pixels to the right.
void	<code>moveUp()</code>	Move the circle a few pixels up.
void	<code>moveVertical(int distance)</code>	Move the circle vertically.
double	<code>perimeter()</code>	Retorna el perimetro del círculo
void	<code>slowMoveHorizontal(int distance)</code>	Slowly move the circle horizontally.
void	<code>slowMoveVertical(int distance)</code>	Slowly move the circle vertically

## E. Usando un paquete. shapes

[En lab01.doc y \*.java]

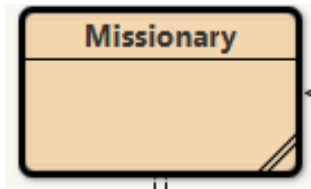
En este punto vamos a crear tres figuras usando el paquete shapes: misionero, canibal y barca. El diseño gráfico lo definen ustedes. Estos son algunos ejemplos.



Las tres figuras deben ofrecer los siguientes métodos.

X	
<pre> + _(): X + makeVisible(): void + makeInvisible(): void + rotate(): void + moveTo(f: int, c: int): void + changeSize(newSize: int): void + changeColor(newColor: String): void </pre>	<p><b>Mini-ciclo: 1</b></p> <pre> _(): X makeVisible() makeInvisible() </pre> <p><b>Mini-ciclo: 2</b></p> <pre> changeSize changeColor </pre> <p><b>Mini-ciclo: 3</b></p> <pre> moveTo rotate </pre>

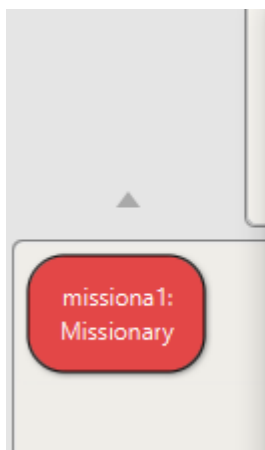
## Missionary




```
public class Cannibal{
    private Rectangle body;
    private Circle head;
    private Circle eyeD;
    private Circle eyeI;
    private Rectangle mouth;
    private Rectangle armD;
    private Rectangle armI;
    private Rectangle leg;
    private int rotacion;
    public String name;
    /**
     * Constructor del Cannibal
     */
    public Cannibal(){
        this.body = new Rectangle();
        this.head = new Circle();
        this.eyeD = new Circle();
        this.eyeI = new Circle();
        this.mouth = new Rectangle();
        this.armD = new Rectangle();
        this.armI = new Rectangle();
        this.leg = new Rectangle();
        initDraw();
    }
}
```

### miniciclo1:

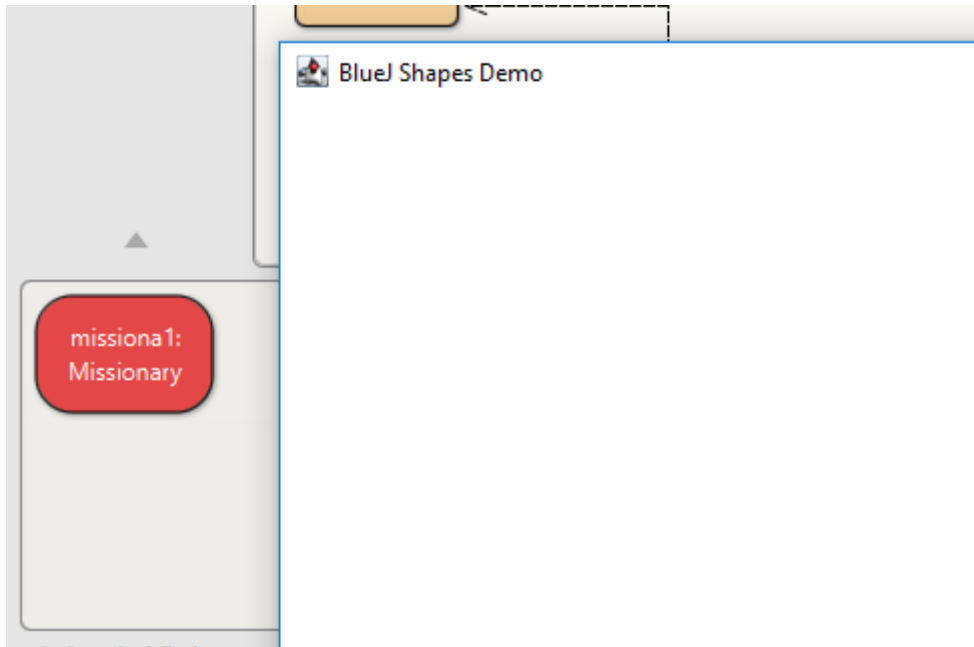
Makevisible: hacer a todo el misionero visible



 BlueJ Shapes Demo

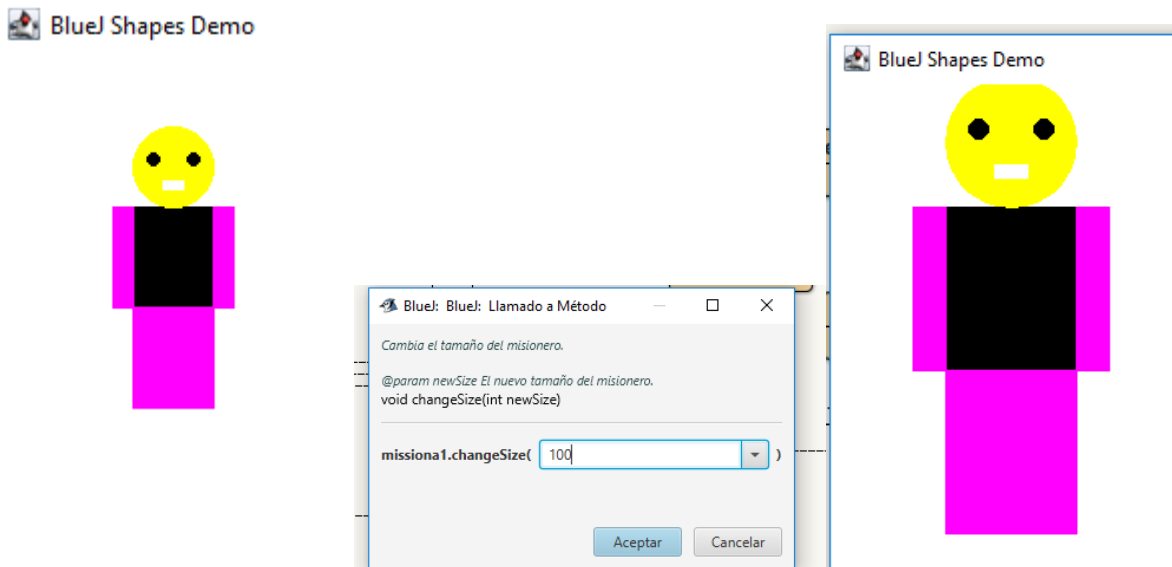


Makeinvisible: hacer a todo el misionero invisible




**miniciclo2:**

ChangeSize: vamos a duplicar el tamaño de el misionero



ChangeColor: cambiaremos el pantalón a verde, las mangas azules y el chaleco rojo


 BlueJ Shapes Demo

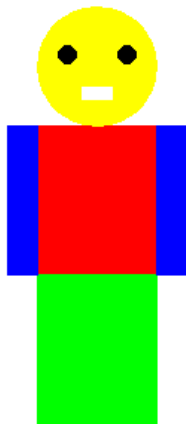


### **miniciclo3:**

MoveTo: moveremos el misionero, tome como referencia el encabezado del BlueJ Shapes Demos

---

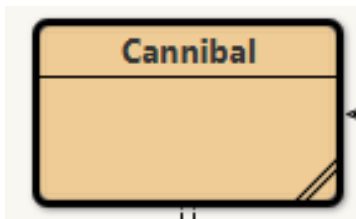
 BlueJ Shapes Demo



rotate: rotaremos 90 grados al misionero hacia la derecha, cada vez que llamamos dicho método, voltará el objeto 90 grados



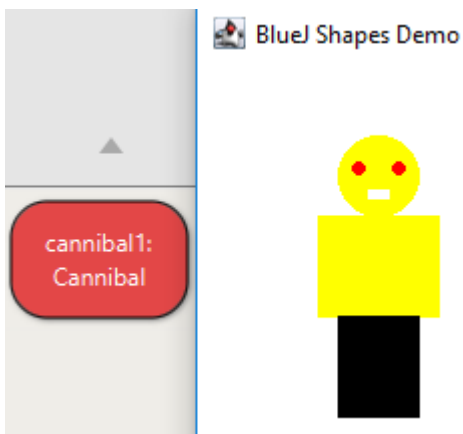
## Cannibal



```
public class Cannibal{
    private Rectangle body;
    private Circle head;
    private Circle eyeD;
    private Circle eyeI;
    private Rectangle mouth;
    private Rectangle armD;
    private Rectangle armI;
    private Rectangle leg;
    private int rotacion;
    public String name;
    /**
     * Constructor del Cannibal
     */
    public Cannibal(){
        this.body = new Rectangle();
        this.head = new Circle();
        this.eyeD = new Circle();
        this.eyeI = new Circle();
        this.mouth = new Rectangle();
        this.armD = new Rectangle();
        this.armI = new Rectangle();
        this.leg = new Rectangle();
        initDraw();
    }
}
```

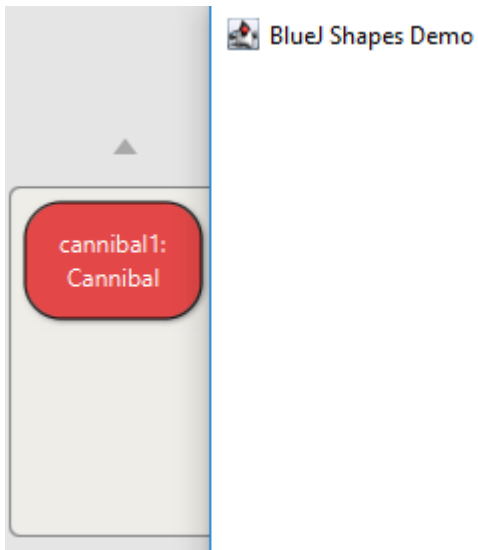
## miniciclo1

Makevisible: hacer a todo el caníbal visible



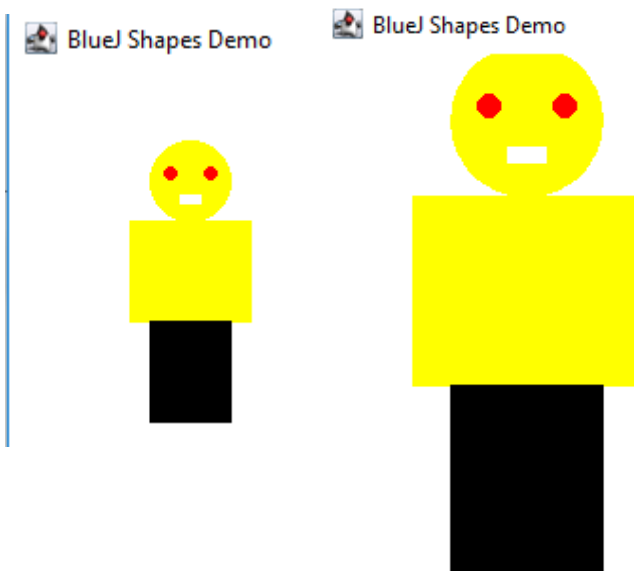


Makeinvisible: hacer a todo el caníbal invisible



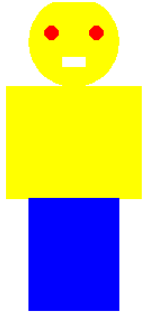
**miniciclo2:**

ChangeSize: vamos a duplicar el tamaño del caníbal



ChangeColor: cambiaremos el pantalón a azul

BlueJ Shapes Demo



### miniciclo3:

MoveTo: moveremos el caníbal, tome como referencia el encabezado del BlueJ Shapes Demos

BlueJ Shapes Demo

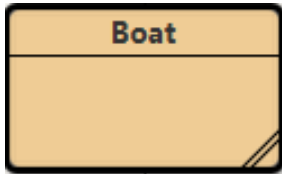


rotate: rotaremos 90 grados al caníbal hacia la derecha, cada vez que llamamos dicho método, volteará el objeto 90 grados

BlueJ Shapes Demo



## Boat

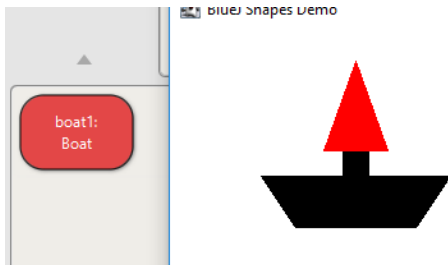


```
public class Boat{
    private Rectangle base;
    private Triangle ladoIzquierdo;
    private Triangle ladoDerecho;
    private Rectangle palo;
    private Triangle velero;
    public boolean pozIz = false;
    public boolean posDe = false;
    public String orilla;
    private int rotate;

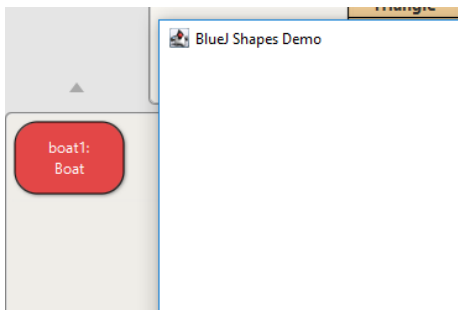
    /**
     * Constructor por defecto de la clase Boat.
     * Inicializa las partes del bote y configura su representación gráfica.
     */
    public Boat(){
        this.base = new Rectangle();
        this.ladoIzquierdo = new Triangle();
        this.ladoDerecho = new Triangle();
        this.velero = new Triangle();
        this.palo = new Rectangle();
        initDraw();
    }
}
```

### miniciclo1

Makevisible: hacer a todo el caníbal visible

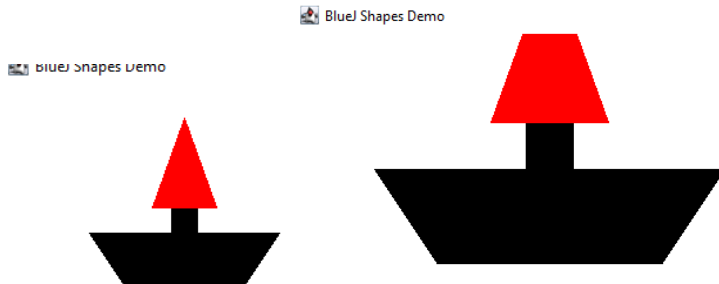


Makeinvisible: hacer a todo el caníbal invisible

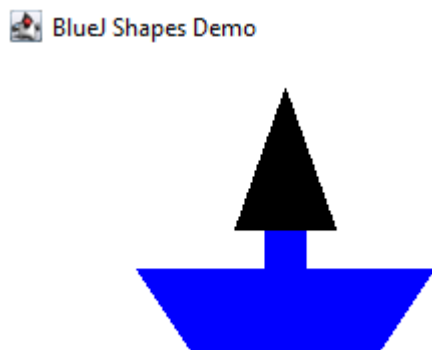


### miniciclo2:

ChangeSize: vamos a duplicar el tamaño del caníbal

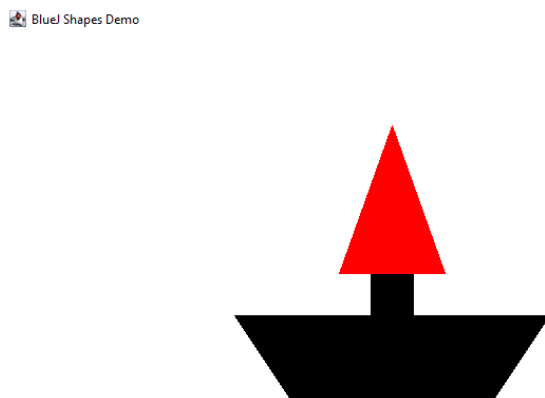


ChangeColor: cambiaremos el color del barco a azul y el velero a negro

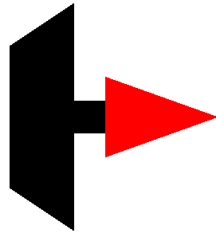


### miniciclo3:

MoveTo: moveremos el caníbal, tome como referencia el encabezado del BlueJ Shapes Demos



rotate: rotaremos 90 grados al caníbal hacia la derecha, cada vez que llamamos dicho método, volteará el objeto 90 grados



## F. Definiendo y creando una nueva clase. **MissionariesCannibals.**

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

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

### Requisitos funcionales

- Crear el estado inicial
- Subir personas en la barca indicando las personas que se deben subir
- Mover la barca hacia la otra orilla
- Bajar las personas de la barca

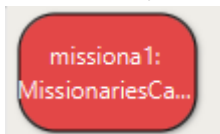
### Requisitos de interfaz

- Las personas únicamente se identifican por su tipo: misionero o canibal
- En caso que no sea posible realizar una de las acciones, debe generar un mensaje de error. Use JOptionPane.

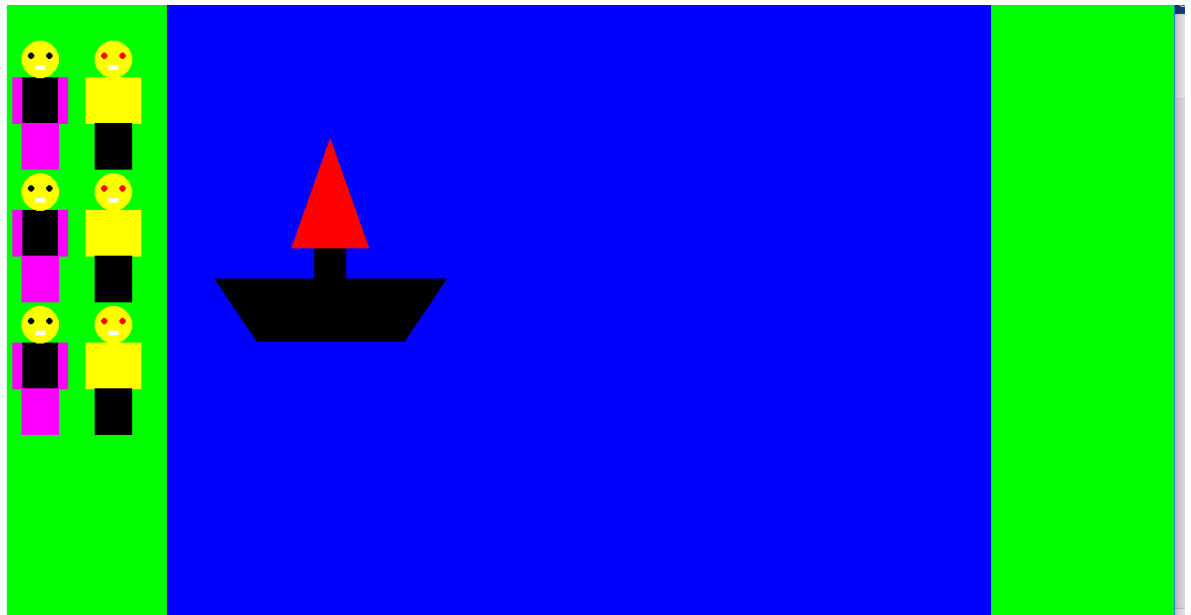
1. Diseñen la clase, es decir, definan los métodos que debe ofrecer.
2. Planifiquen la construcción considerando algunos mini-ciclos.
3. Implementen la clase. Al final de cada mini-ciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes.
4. Indiquen las extensiones necesarias para reutilizar las clases de las figuras y el paquete `shapes`. Expliquen.

MissionariesCannibals	
<pre> + _() : MissionariesCannibals - makeVisible() : void - posicionesIniciales() : void - posicionesPaisaje() : void - posicionesInicialesBoat() : void - posicionesInicialesMissionary() : void - posicionesInicialesCannibal() : void - verificarPerderOrillaIzquierda() : boolean - verificarPerderOrillaDerecha() : boolean - verificarCantidadMisionerosCannibalesOrillas() : boolean + insertarMissionary() : void - insertarMissionaryEnOrilla(misionaryes : list) : void - moverMissionary(misioneros : Missionary, x : int, y : int) : void + insertarCannibal() : void - insertarCannibalEnOrilla(canibales : list) : void - moverCannibal(canibales : Cannibal, x : int, y : int) : void + moverBote() : void - moverBoteIzquierda(cantidad : int) : void - moverBoteDerecha(cantidad : int) : void + descargarMissionary() : void - descargarMissionaryIzquierda() : void - descargarMissionaryDerecha() : void + descargarCannibal() : void - descargarCannibalIzquierda() : void - descargarCannibalDerecha() : void - verificarSiMisionerosGanan() : void + control() : void </pre>	

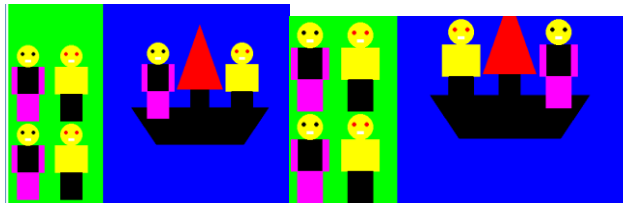
- 1.
2. **Miniciclo1:** desde el constructor hasta todo lo que tiene que ver con la posición inicial del juego  
**Miniciclo2:** entrada de caníbales y misioneros al barco  
**Miniciclo3:** salida de los misioneros y caníbales del barco  
**Miniciclo4:** movimiento del bote
3. Miniciclo1: el propósito de esta prueba es rectificar que las posiciones iniciales sean verídicas, que todos los jugadores estén y que ninguna figura se sobreponga mal de otra, en resumen, si tenemos bien esto, podemos continuar con las funcionalidades del juego



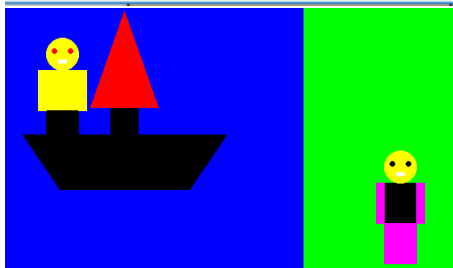
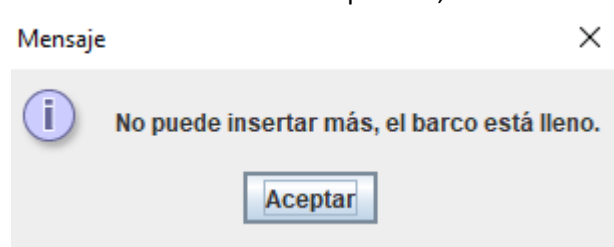
Una vez se crea la clase, se inicializa el primer objeto con ya todo visible



Miniciclo2: debemos garantizar que los caníbales y misioneros puedan entrar al barco sin que uno se sobreponga a otro y evitando sobre cupo, además se deben dejar insertar desde cualquier lado de la orilla.



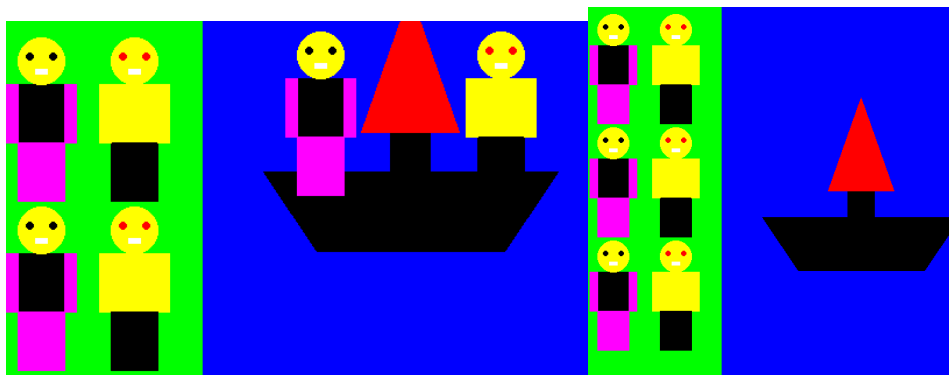
Si intentamos añadir otro tripulante, salta el error:



Ya sea de la orilla izquierda o derecha, siempre se dejan insertar

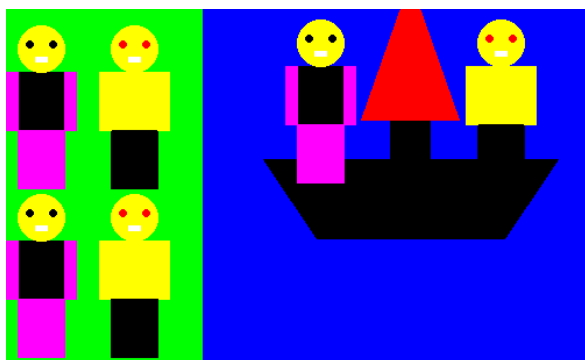
Minicilo3:

Así como es importante que los personajes se suban, de igual manera deben bajar

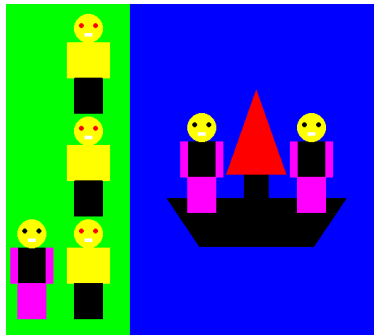


Miniciclo4:

Por último, el barco debe poder moverse, pero jamás dejar que queden más caníbales que misioneros en una orilla:

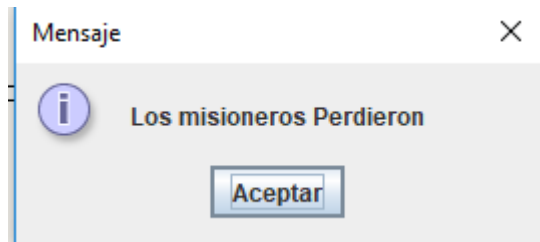


Pero si tenemos dos misioneros que quieren partir:

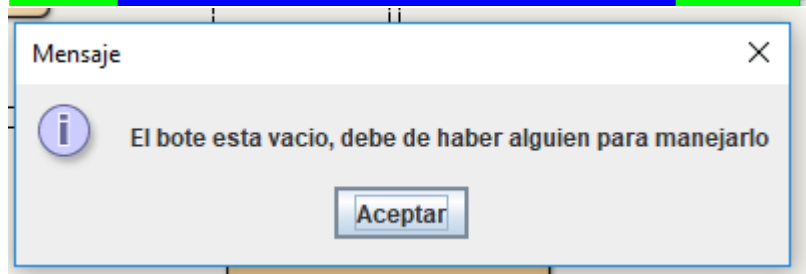
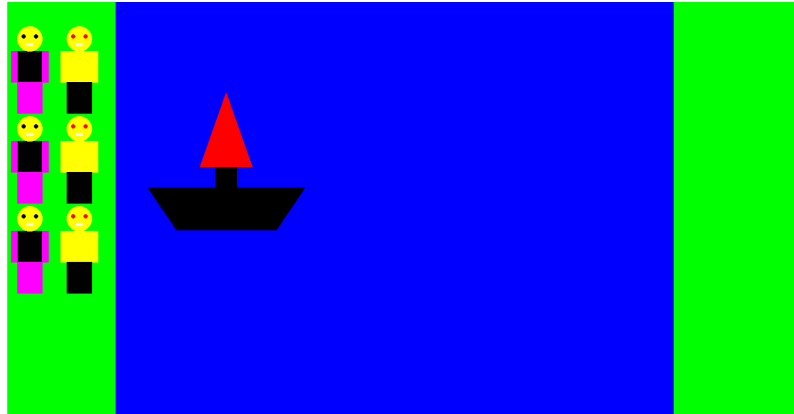


Salta el siguiente error al momento de querer mover el bote:





De igual manera si el bote está vacío y quiere moverse, debe saltar otro error:



4. Prácticamente reusamos todas las figuras anteriores, iniciamos creando los personajes a través de círculos y rectángulos, y el barco lo hicimos usando triángulos y rectángulos. Esto se hizo desde los atributos, es decir, indicando que figuras íbamos a usar y que parte del objeto serían, por ejemplo, si queríamos guardar la cabeza de un personaje, usaríamos el círculo, y en los atributos lo declaramos:  
`private head = new Cricle();`

## G. De python a java

[En [lab01.doc](#)]

En este punto vamos a evaluar el video [DE PYTHON A JAVA](#) en la encuesta preparada con ese objetivo. ¿En cuáles puntos estuvieron en desacuerdo<sup>9</sup>? Expliquen.

En realidad, en ninguno, sentimos que la explicación fue suficiente para entender lo básico sobre el lenguaje de programación Java y de cómo entender la POO en java, aunque si sentimos que estaría genial con otro ejemplo sobre POO, para apropiarnos más de los conceptos dados.

## BONO. Nuevos requisitos funcionales. MissionariesCannibals.

[En lab01.doc. MissionariesCannibals.java]

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

### Nuevos requisitos funcionales

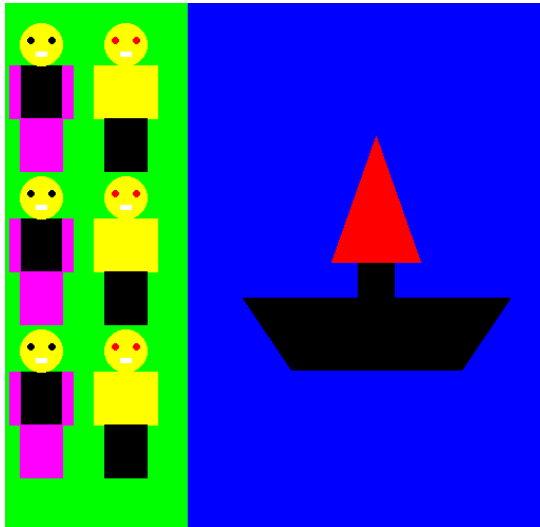
- Subir las personas a la barca (la máquina decide que personas subir). Explique la estrategia.
  - Deshacer el último movimiento
1. Diseñen, es decir, definan los métodos que debe ofrecer.
  2. Implementen los nuevos métodos . Al final de cada método realicen una prueba indicando su propósito. Capturen las pantallas relevantes.

Realizamos el bono de deshacer el último movimiento:

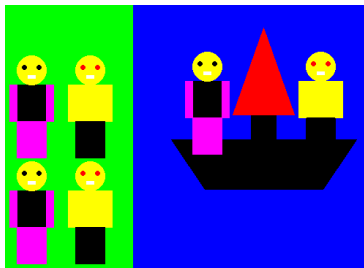
```
- descargarCannibalDerecha() : void  
- verificarSiMisionerosGanan() : void  
+ controlz() : void
```

Solo usamos un método demás llamado control()

1. Tener las posiciones iniciales:



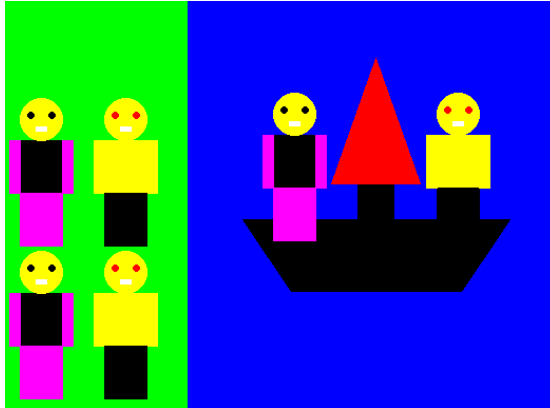
2. Subir a misionero y a un caníbal en ese orden:



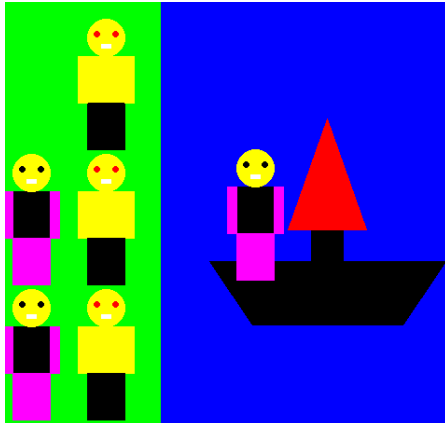
3. Mover el barco:



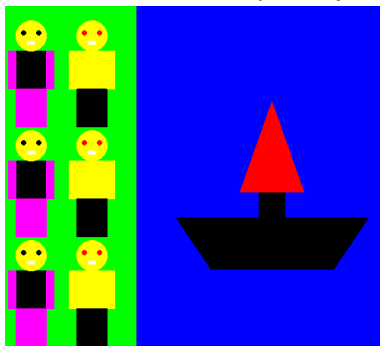
4. Usar el control Z



5. Otra vez lo usamos, y se bajará primero el caníbal



6. Lo usamos de nuevo y se baja el misionero



## **RETROSPECTIVA**

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes? (Horas/Hombre)
2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?
4. ¿Cuál consideran fue el mayor logro? ¿Por qué?
5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?
6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?

1. 12 horas por Cardona y 12 horas por Cárdenas.
2. El laboratorio se encuentra totalmente finalizado, puesto que trabajamos desde el primer día arduamente para dejarlo en dicho estado.
3. La práctica xp más útil fue para nosotros la programación a pares, pues eso nos ayudó a rendir en el procedimiento del laboratorio y a resolvernos las dudas del otro.
4. Nuestro mayor logro fue dejar el juego gráficamente bonito y funcional, puesto que cada método lo logramos realizar.
5. Fueron varios, entre ellos diseñar la lógica de la inserción de los personajes al barco, cambiar el tamaño de los objetos, rotarlos, entre otros. No obstante, buscamos ayuda del otro y con ello fue suficiente para completar todas las funcionalidades.
6. Fue que nos entendimos muy bien, no hubieron pleitos o desacuerdos grandes y cualquier problema que uno tuviera el otro ayudaba.