

# PROGRAMACIÓN ORIENTADA A OBJETOS


## Construcción. Clases y objetos.

### 2021-1

### Laboratorio 1/6

## 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 java<sup>1</sup>.
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](#).

## ENTREGA

- ➔ Incluyan en un archivo **.zip** los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
- ➔ Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios correspondientes.

## SHAPES

### Conociendo el proyecto shapes

[En lab01.doc] [TP 20]

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?
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 ofrece la clase `Rectangle`? (c) ¿Cuántos métodos ofrece la clase `Rectangle`? (d) ¿Cuáles métodos ofrece la clase `Rectangle` para que la figura cambie (incluya sólo el nombre)?
4. En el **código** de cada clase está el detalle de la implementación. Revisen el código de la clase `Rectangle`. Con respecto a los atributos: (a) ¿Cuántos atributos realmente tiene? (b) ¿Cuáles atributos describen la forma del círculo?. Con respecto a los métodos: (c) ¿Cuántos métodos tiene en total? (d) ¿Quiénes usan los métodos privados?
5. Comparando la **documentación** con el **código** (a) ¿Qué no se ve en la documentación? (b) ¿por qué debe ser así?
6. En el código de la clase `Rectangle` revisen el detalle del atributo `EDGES`. (a) ¿Qué se está indicando al decir que es `static`? (b) ¿Cómo decimos que `EDGES` es una constante? (c) Actualícenlo.

---

1 <http://docs.oracle.com/javase/8/docs/api/>

2 Menu: Tools-Project Documentation

- En el código de la clase `Rectangle` revisen el detalle del tipo de los atributos `height` y `width` (a) ¿Qué se está indicando al decir que es `int`? (b) Si sabemos todos nuestros rectángulos van a ser muy pequeños (lados menor a 100), ¿de qué tipo deberían ser estos atributos? Si algunos de nuestros rectángulos pueden ser muy grandes (lados mayor a 220000000), (c) ¿de qué tipo deberían ser estos atributos? (d) ¿qué restricción tendría? Expliquen claramente sus respuestas.
- ¿Cuál dirían es el propósito del proyecto “shapes”?

## Manipulando objetos. Usando opciones.

[En lab01.doc] [TP 8]

- Crean un objeto de cada una de las clases que lo permitan<sup>3</sup>. (a) ¿Cuántas clases hay? ¿Cuántos objetos crearon? (b) ¿Por qué?
- Inspeccionen el **estado** del objeto `:Rectangle`<sup>4</sup>. ¿Cuáles son los valores de inicio de todos sus atributos? Capture la pantalla.
- Inspeccionen el **comportamiento** que ofrece el objeto `:Rectangle`<sup>5</sup>. (a) Capturen la pantalla. (b) ¿Por qué no aparecen todos los que están en el código?
- Construyan, con “shapes” sin escribir código, una propuesta de la imagen del logo de su red social favorita. (a) ¿Cuántas y cuáles clases se necesitan? (b) ¿Cuántos objetos se usan en total? (c) Capturen la pantalla.

## Manipulando objetos. Analizando y escribiendo código.

[En lab01.doc] [TP 30]

```
Rectangle yellow;
Rectangle blue;
Rectangle red;
//1
yellow= new Rectangle();
blue= new Rectangle();
red= yellow;
yellow.makeVisible();
//2
yellow.changeSize(30,80);
yellow.changeColor("yellow");
//3

blue = new Rectangle();
blue.changeSize(20,80);
blue.changeColor("blue");
blue.moveVertical(30);
//4
red.changeColor("red");
red.changeSize(20,80);
red.moveVertical(50);
red.makeVisible();
//5
blue.makeVisible();
//6
```

- Lean el código anterior. (a) ¿cuál es la figura resultante? (b) Píntenla.
- 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? (c) ¿qué color tiene cada uno de ellos? (d) ¿cuántos objetos se ven? Al final, (e) Expliquen sus respuestas. (f) Capturen la pantalla.
- Compare figura pintada en 1. con la figura capturada en 2. , (a) ¿son iguales? (b) ¿por qué?

3 Clic derecho sobre la clase

4 Clic derecho sobre el objeto

5 Hacer clic derecho sobre el objeto.

6 Menú. View-Show Code Pad.

## Extendiendo clases

[En lab01.doc y \*.java]

1. Desarrollen en `Rectangle` el método `rainbow(String [] colors)` (pasa por los colores indicados en al parametros) . ¡Pruébenlo! Capturen dos pantallas.
2. Desarrollen en `Rectangle` el método `rotate()` (rota el rectangulo 90 grados en sentido del reloj) . ¡Pruébenlo! Capturen dos pantallas.
3. Desarrollen en `Rectangle` el método `area()` . ¡Pruébenlo! Capturen una pantalla.
4. Desarrollen en `Rectangle` el método `zoom(char c)` (aumenta (+) y disminuye (-) 10% de su area) . ¡Pruébenlo! Capturen dos pantallas.
5. Generen nuevamente la documentación y revise la información de estos nuevos métodos. Capturen la pantalla.

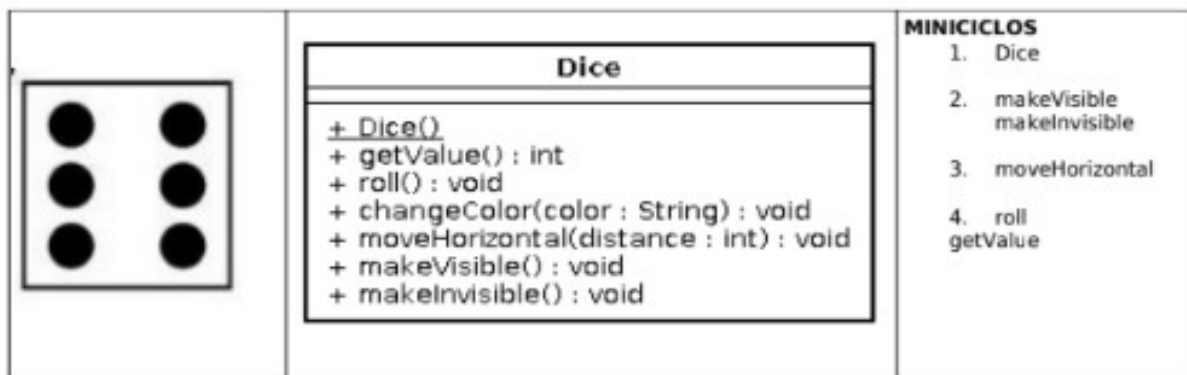
## DICESE. Dice.

DICESE es un juego de dados solitario.  
En este punto vamos a construir dos versiones de DICESE  
La **primera versión** es simple: sólo tiene una fila. Para ganar se deben tener todos los elementos ordenados.  
La **segunda versión** es cuadrado. Para ganar se deben tener ordenados todos los elementos de una fila, una columna o una diagonal.  
Las dimensiones debe estar entre 2 y 6.



## Implementando una nueva clase. Dice.


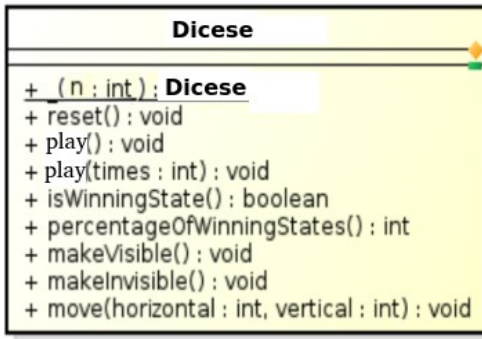
[En lab01.doc. Dice.java]



1. Revisen el diseño y clasifiquen los métodos en: constructores, analizadores y modificadores.
2. Desarrollen la clase `Dice` considerando los 4 mini-ciclos. Al final de cada mini-ciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes.

## Implementando una nueva clase. Dicese.

[En lab01.doc. Dicese.java]

	 <pre>classDiagram     class Dicese {         + (n : int) : Dicese         + reset() : void         + play() : void         + play(times : int) : void         + isWinningState() : boolean         + percentageOfWinningStates() : int         + makeVisible() : void         + makeInvisible() : void         + move(horizontal : int, vertical : int) : void     }</pre>	<b>Miniciclo 1</b> Dicese play play(times) isWinningState <b>Miniciclo 2</b> reset percentageOfWinning <b>Miniciclo 3</b> makeVisible makeInvisible move
---	---	---

1. ¿De qué tipo son los elementos en Dicese? ¿Cuántos posibles estados puede tener?
2. ¿Cuál es la probabilidad de ganar? Explique su respuesta.
3. Clasifiquen los métodos en: constructores, analizadores y modificadores.
4. Desarrollen la clase `Dicese` considerando los miniciclos. Al final de cada miniciclo realicen una prueba. Capturen las pantallas relevantes.
5. **¿Cual es el porcentaje de estados ganadores después de hacer 1, 10, 100 y 1000 jugadas? Presente un análisis de los datos considerando la respuesta dada en 2.**

## Definiendo y creando una nueva clase. BiDicese

[En lab01.doc. BiDicese.java]

El objetivo es implementar diseñar e implementar un tragamonedas cuadrado.

### Requisitos funcionales

3. Permitir crear un `BiDicese` indicando el tamaño.
4. Permitir reiniciar
5. Permitir jugar (todas filas giran)
6. Permitir jugar indicando la fila con el que se desea jugar (1 ...n de derecha a izquierda)
7. Permitir jugar indicando el dado con el que se desea jugar (fila, columna)
8. Permitir consultar si se ha ganado
9. Permitir consultar el porcentaje de juegos ganadores desde el último reinicio

### Requisitos de interfaz

9. Las operaciones se deben ofrecer como métodos públicos de la clase
10. `BiDicese` debe "sonar" cada vez que llega a un estado ganador
11. Se debe presentar un mensaje amable al usuario si hay algún problema. Consulte y use el método `showMessageDialog` de la clase `JOptionPane`.
1. Diseñen la clase, es decir, definan los métodos que debe ofrecer.
2. Planifiquen la construcción considerando algunos miniciclos.
3. Implementen la clase. Al final de cada miniciclo realicen una prueba de aceptación. Capturen las pantallas relevantes.
4. Indiquen las extensiones necesarias para reutilizar la clase `Dicese`. Explique.
5. Propongan un nuevo método para enriquecer el juego.

## **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?