

# PROGRAMACIÓN ORIENTADA A OBJETOS



## Introducción. Clases y objetos.

Agosto 2017

Laboratorio 1/6

### OBJETIVOS

Desarrollar competencias básicas para:

1. Apropiar un paquete de clases 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 preparados para tal fin.
- En el foro de entrega de avance deben indicar los logros y los problemas pendientes por resolver.

### SHAPES

#### Conociendo el proyecto shapes

[En **lab01.doc**]

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](#)
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” ¿qué clases ofrece? ¿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 (visible desde su navegador): ¿qué clases tiene el paquete `shapes`? ¿qué atributos tiene la clase `Rectangle`? ¿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: ¿cuántos atributos tiene? ¿cuáles son privados y cuáles públicos? ¿por qué?. Con respecto a los métodos: ¿cuántos métodos tiene en total? ¿cuáles son privados? ¿quiénes los usan?
5. ¿Cuál dirían es el propósito del proyecto “shapes”?

---

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

2 Menu: Tools-Project Documentation

## Manipulando objetos. Usando opciones.

[En lab01.doc]

1. Creen un objeto de cada una de las clases que lo permitan<sup>3</sup>. ¿cuántas clases hay? ¿cuántos objetos crearon? ¿por qué?
2. Inspeccionen el **estado** del objeto :Rectangle<sup>4</sup>. ¿cuáles son los valores de inicio de todos sus atributos? Capturen las pantallas
3. Inspeccionen el **comportamiento** que ofrece el objeto :Rectangle<sup>5</sup>. Capturen la pantalla. ¿por qué no aparecen todos los que están en el código?
4. Construyan, con “shapes” sin escribir código, una propuesta de la imagen de su mascota favorita (incluyan una copia del original). ¿Cuántas y cuáles clases se necesitan? ¿Cuántos objetos se usan en total? Capturen la pantalla.

## Manipulando objetos. Analizando y escribiendo código.

[En lab01.doc]

```
Rectangle yellow;  
Rectangle blue;  
Rectangle red;  
//1  
yellow= new Rectangle();  
blue= new Rectangle();  
red = blue;  
//2  
yellow.changeSize(30,80);  
yellow.changeColor("yellow");  
yellow.makeVisible();  
//3  
blue.changeSize(20,80);  
blue.changeColor("blue");  
blue.makeVisible();  
//4  
blue.moveVertical(30);  
//5  
red.changeSize(20,80);  
red.changeColor("red");  
red.moveVertical(50);  
red.makeVisible();  
//6
```

1. Revisen el código anterior e indiquen ¿cuál es la figura resultante? Píntenla.
2. Habiliten la ventana de código en línea<sup>6</sup>, escriban el siguiente código y en cada punto señalado indiquen: ¿cuántas variable existen? ¿cuántos objetos existen? ¿cuántos objetos se ven? ¿qué color tiene cada uno de ellos? Explique.
3. Es igual la figura la figura generada a la inicial, ¿por qué?

## Extendiendo clases

[En lab01.doc y \*.java]

1. Desarrollen en Rectangle el método rainbow() (que hace que recorra los colores del arco iris y luego vuelva al original) . ¡Pruébenlo!
2. Desarrollen en Rectangle el método twice() (que hace que tenga el doble del área manteniendo sus proporciones) . ¡Pruébenlo!
3. Desarrollen en Rectangle el método perimetro() . ¡Pruébenlo!
4. Genere nuevamente la documentación y revise la información de estos nuevos métodos. Capture la pantalla.

---

3 Clic derecho sobre la clase

4 Clic derecho sobre el objeto

5 Hacer clic derecho sobre el objeto.

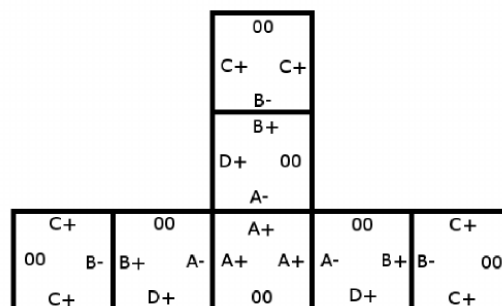
6 Menú. View-Show Code Pad.

## SELF-ASSEMBLY. Molecules.

ACM está experimentando con un proceso llamado ensamblaje automático. En este proceso las moléculas con afinidad se unen para formar estructuras. Las moléculas son descritas indicando su tipo; es decir, sus cuatro conectores: letras mayúsculas seguidas de un signo. Las moléculas son compatibles si sus letras son iguales y sus signos contrarios. Existe un conector especial, el 00, que no es compatible con ningún conector. Las moléculas pueden ser rotadas, reflejadas. El objetivo es, dado un conjunto de tipos de moléculas, generar estructuras cerradas.

[De 2013 World Finals Problem A Self-Assembly]

**NO DEBEN RESOLVER EL PROBLEMA DE LA MARATÓN**



## Implementando una nueva clase. Molecule.

[En lab01.doc. Molecule.java]

Molecule
+ <code>_(type : String) : Molecule</code> + <code>getType() : String</code> + <code>makeVisible() : void</code> + <code>makeInvisible() : void</code> + <code>moveDown() : void</code> + <code>moveRight() : void</code> + <code>reflect() : void</code> + <code>rotate() : void</code>

### MINICICLOS

1. `_(type:String)`  
(`'00 C+ B- C+'`) Sentido del reloj 12-3-6-9  
`getType()`
2. `makeVisible()`  
`makeInvisible()`
3. `reflect()`  
`rotate()`
4. `moveRight()`  
`moveDown()`

Las moléculas se deben visualizar con colores diferentes para cada letra y formas o rellenos diferentes para los símbolos positivo y negativo. Los movimientos corresponden a su tamaño.

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

## Definiendo y creando una nueva clase. SelfAssembly

[En lab01.doc. SelfAssembly.java]

### Requisitos funcionales

- Crear un proceso `SelfAssembly` dados los tipos de moléculas
- Crear una molécula dada la posición de su tipo
- Rotar, reflejar, mover (derecha y abajo) y colocar en la estructura la última molécula creada. Debe validar que se cumplan las condiciones de compatibilidad
- Conocer si una estructura está cerrada (no puede crecer más)

### Requisitos de interfaz

- Las numeraciones inician en 1.
- La nueva molécula creada inicia en la posición de pantalla (0,0)
- La nueva molécula, hasta antes de ser colocada, debe lucir diferente.
- La estructura es una matriz y sus posiciones se indican en términos de filas x columnas.
- Se debe presentar un mensaje de felicitación cuando la estructura se cierre `showMessageDialog` de la clase `JOptionPane`.
- En caso que no sea posible realizar una de las acciones, el juego debe simulador con un sonido.

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 la clase `Molecule`. Expliquen
5. Propongan un nuevo método para enriquecer el simulador.

## **Extendiendo una clase. SelfAssembly**

[En lab01.doc. SelfAssembly.java]

El objetivo es extender el simulador para que la máquina ayude a crear la estructura

### **Nuevos requisitos funcionales**

- Permitir pedir crecer la estructura de la mejor manera sin decir cuál tipo de molécula ni en cuál posición (la máquina lo decide). Explique la estrategia.

## **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?