

PROGRAMACIÓN ORIENTADA A OBJETOS



Introducción. Clases y objetos.

2020-1

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 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](#).

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]

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**² presenta las clases del proyecto y, en este caso, la especificación de sus componentes públicos. De acuerdo con la documentación generada: ¿qué clases tiene el paquete `shapes`? ¿qué atributos tiene la clase `Circle`? ¿cuáles métodos ofrece la clase `Circle` 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 `Circle`. Con respecto a los atributos: ¿cuántos atributos realmente tiene? ¿cuáles son privados y cuáles públicos?. Con respecto a los métodos: ¿cuántos métodos tiene en total? ¿cuáles son privados?. ¿Quiénes usan los componentes privados?
5. ¿Qué no se ve en la documentación? ¿por qué debe ser así?
6. En el código de la clase `Circle` revisen el detalle del atributo `PI`. ¿qué se está indicando?
7. ¿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 permitan³. ¿cuántas clases hay? ¿cuántos objetos crearon? ¿por qué?
2. Inspeccionen el **estado** del objeto :Circle⁴, ¿cuáles son los valores de inicio de todos sus atributos? Capturen las pantallas
3. Inspeccionen el **comportamiento** que ofrece el objeto :Circle⁵. 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 *comic* favorito. ¿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= 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
```

1. Lean el código anterior ¿cuál es la figura resultante? Píntenla.
2. Habiliten la ventana de código en línea⁶, escriban el código y para cada punto señalado indiquen: ¿cuántas variables existen? ¿cuántos objetos existen? ¿qué color tiene cada uno de ellos? ¿cuántos objetos se ven? Expliquen. Capturen la pantalla.
3. Es la figura pintada en 1. igual a la figura capturada en 2. , ¿por qué?

Extendiendo clases

[En lab01.doc y *.java]

1. Desarrollen en Rectangle el método `blick(times)` (parpadea el número dado de veces) . ¡Pruébenlo!
2. Desarrollen en Rectangle el método `perimeter()` . ¡Pruébenlo!
3. Desarrollen en Rectangle el método `rotate()` (que hace que rote a la derecha trasladándose) . ¡Pruébenlo!
4. Generen 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.

NÚMEROS MAYAS

En este punto vamos a construir dos herramientas para los calculistas Mayas. Xook y CalcXook
Los mayas utilizaban un sistema numérico de base 20; es decir, los “dígitos mayas” van de 0 a 19.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

Implementando una nueva clase. Xook.

[En lab01.doc. XoX.java]

	<div style="border: 1px solid black; padding: 5px; margin: 5px;"> <p style="text-align: center; margin: 0;">Xook</p> <hr/> <pre> + _(value : int) : Xook + getValue() : int + random() : void + makeInvisitable() : void + makeVisible() : void + moveHorizontal(distance : int) : void + changeColor(color : String) : void </pre> </div>	<p>Miniciclo 1 Xook getValue</p> <p>Miniciclo 2 makeVisible makeInvisible</p> <p>Miniciclo 3 random setValue</p> <p>Miniciclo 4 moveHorizontal changeColor</p>
--	---	--

1. ¿Cuántos objetos conforman un Xook? ¿Cuáles clases se usan?
2. Clasifiquen los métodos en: constructores, analizadores y modificadores.
3. ¿Cuáles métodos requieren un prerequisite? Explique su respuesta.
4. Desarrollen la clase Xook considerando los miniciclos. Al final de cada miniciclo realicen una prueba. Capturen las pantallas relevantes.

Definiendo y creando una nueva clase. CalcXook

El objetivo de esta clase es ofrecer a los matemáticos mayas una calculadora simple. Esta calculadora es de pila de tamaño máximo dos, maneja sólo números positivos y ofrece las operaciones básicas: suma, resta, multiplicación y división entera.

Una calculadora de pila es la que opera almacenando números en la pila y para realizar las operaciones utiliza los dos últimos números (los desempila y empile su resultado).

Requisitos funcionales

- Permitir crear la calculadora, indicando el color.
- Permitir al usuario Introducir un número en decimal
- Realizar las operaciones básicas (+,-,*)

Requisitos de interfaz

- Las operaciones de la pila se ofrecen como métodos públicos de la clase CalcXook.
- La calculadora debe presentar los números de la pila en números mayas.
- 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 `CalXook`, 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 `CalXook`. Explique.
5. Propongan un nuevo método para enriquecer la calculadora.

Extendiendo una clase. `CalcXook`

[En `lab01.doc`. `CalcXoX.java`]

El objetivo es extender la calculadora maya para los matemáticos más avanzados.

Nuevos requisitos

1. Permite trabajar con un número indefinido de operandos, no sólo dos.
2. Permite indicar el número máximo de operandos que quiere tener visible.
3. Ofrece operaciones adicionales a las operaciones básicas.

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?