

PROGRAMACIÓN ORIENTADA A OBJETOS



Introducción. Clases y objetos.

2020-2

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**] [TP 18]

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**² 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 `Circle`? (c) ¿Cuántos métodos ofrece la clase `Circle`? (d) ¿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: (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 `Circle` revisen el detalle del atributo `PI`. (a) ¿Qué se está indicando al decir que es `static`? (b) ¿Cómo decimos que `PI` es una constante? (c) Actualícenlo.
7. En el código de la clase `Circle` revisen el detalle del tipo del atributo `diameter`. (a) ¿Qué se está indicando al decir que es `int`? (b) Si sabemos todos nuestros círculos

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

2 Menu: Tools-Project Documentation

van a ser muy pequeños (diámetro menor a 100), ¿de que tipo podría ser `diameter`? Si algunos de nuestros círculos pueden ser muy grandes (diámetro mayor a 220000000), (c) ¿de que tipo debería ser `diameter`? (d) ¿qué restricción tendría? Expliquen claramente sus respuestas.

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

Manipulando objetos. Usando opciones.

[En lab01.doc] [TP 8]

1. Creen un objeto de cada una de las clases que lo permitan³. (a) ¿Cuántas clases hay? ¿Cuántos objetos crearon? (b) ¿Por qué?
2. Inspeccionen el **estado** del objeto `:Circle`⁴, ¿Cuáles son los valores de inicio de todos sus atributos? Capture la pantalla.
3. Inspeccionen el **comportamiento** que ofrece el objeto `:Circle`⁵. (a) Capturen la pantalla. (b) ¿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. (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]

```
Circle face;
Circle leftEar;
Circle rightEar;
//1
face=new Circle();
face.changeSize(50);
//2
face.moveVertical(20);
face.changeColor("black");
face.makeVisible();
//3

leftEar=new Circle();
leftEar.changeColor("black");
leftEar.moveHorizontal(-10);
//4
rightEar=leftEar;
rightEar.moveHorizontal(30);
rightEar.makeVisible();
//5
leftEar.makeVisible();
//6
```

1. Lean el código anterior. (a) ¿cuál es la figura resultante? (b) Píntenla.
2. Habiliten la ventana de código en línea⁶, 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.
3. 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 `Circle` el método `rainbow()` (pasa por los colores del arco iris) . ¡Pruébenlo! Capturen dos pantallas.
2. Desarrollen en `Circle` el método `area()` . ¡Pruébenlo! Capturen una pantalla.
3. Desarrollen en `Circle` el método `zoom(char c)` (aumenta (+) y disminuye (-) 10% de su `area()`) . ¡Pruébenlo! Capturen dos pantallas.
4. Generen nuevamente la documentación y revise la información de estos nuevos métodos. Capturen la pantalla.

SELF-ASSEMBLY. Pieces.

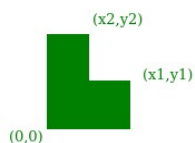
Durante las Finales Mundiales ACM ICPC del 2017 en Marrakesh, uno de los jueces compró un bonito rompecabezas de madera. A diferencia de los tradicionales, que son creados cortando una imagen rectangular existente, todas las piezas de este rompecabezas han sido cortadas y pintadas por separado. Dadas estas propiedades, la forma de cada pieza individual es la única manera de saber dónde se debe colocar cada pieza. El juez quiere saber si es posible escribir un programa para resolver este rompecabezas

[De 2016 WorldFinals Problem H Polygonal Puzzle]

NO DEBEN RESOLVER EL PROBLEMA DE LA MARATÓN



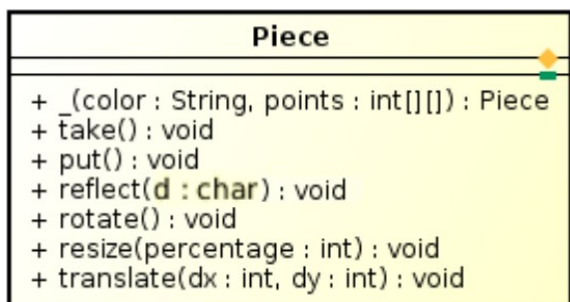
LAS PIEZAS VAN A SER ESPECIALES
 Para crearlas se darán las esquinas superior derecha de los rectángulos que la conforman de izquierda a derecha
`figura = {rectangulo1, rectangulo2, ... ,rectangulon}`
`rectangulo={x, y}`



`{{x2,y2}, {(x1,y1)}}`

Implementando una nueva clase. Piece.

[En lab01.doc. Piece.java]



Una pieza recién creada queda en la mano en la posición (0,0).
 Las piezas que están en la mano deben tener una señal que las distinga.
 La pieza se refleja sobre sus cuatro lados: N, S, E, W. (Norte, Sur, Este, Oeste)
 La rotación siempre es de 90 grados en sentido de las agujas del reloj.

MINICICLOS

7. `_(color:String, points: int[][])`
8. `take()`
`put()`
9. `reflect(d: char)`
`rotate()`
10. `resize(percentage:int)`
`translate(dx:int,dy:int)`

1. Revisen el diseño y clasifiquen los métodos en: constructores, analizadores y modificadores.
2. Desarrollen la clase `Piece` 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. Polygonal Puzzle

[En lab01.doc. Polygonal Puzzle.java]

Requisitos funcionales

1. Crear un rompecabezas indicando sus dimensiones.
2. Crear una nueva pieza. La pieza recién creada queda en la mano.
Deben ofrecer dos formas de crear la pieza: (a) dada su información (b) al azar
3. Tomar una que ya está en el rompecabezas identificándola con su color.
4. Rotar, reflejar, trasladar, cambiar el tamaño de la pieza que está en la mano.
5. Colocar la pieza que están en la mano en el rompecabezas.
Debe validar que si se pueda ubicar la pieza en esa posición.
6. Conocer si el rompecabezas está armado

Requisitos de interfaz

1. Se debe presentar un mensaje de felicitación cuando el rompecabezas esté armado.
Use `showMessageDialog` de la clase `JOptionPane`.
2. En caso que no sea posible realizar una de las acciones, el juego debe generar un sonido para alertar el error.

1. Diseñen la estructura de la clase, es decir, definan los atributos que debe tener.
2. Diseñen los servicios de la clase, es decir, definan todos los métodos que debe ofrecer para cumplir los requisitos.
3. Planifiquen la construcción considerando algunos mini-ciclos que puedan ser probados.
4. Implementen la clase por mini-ciclos. Al final de cada mini-ciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes.
5. Expliquen claramente el alcance de su solución en términos de mini-ciclos.
6. Indiquen las extensiones necesarias para reutilizar la clase `Piece`. Expliquen
7. Propongan un nuevo método para enriquecer el simulador.

Extendiendo una clase. Polygonal Puzzle

[En lab01.doc. Polygonal Puzzle.java]

El objetivo es extender el simulador para que la máquina ayude a armar el rompecabezas

Nuevos requisitos funcionales

- Dada el área, crear una pieza que pueda ubicarse para ir armando el rompecabezas (la máquina lo decide). Explique la estrategia.
- Ubicar la pieza que está en la mano en un sitio adecuado en el rompecabezas (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?