

PROGRAMACIÓN ORIENTADA A OBJETOS



Introducción. Clases y objetos.

2019-1

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.
- ➔ 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**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 :Triangle⁴. ¿cuáles son los valores de inicio de todos sus atributos? Capturen las pantallas
3. Inspeccionen el **comportamiento** que ofrece el objeto :Triangle⁵. 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 cómic 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 variable existen? ¿cuántos objetos existen? ¿qué color tiene cada uno de ellos? ¿cuántos objetos se ven? Expliquen. Capturen la pantalla.
3. Es igual 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 blink(times) (que hace que parpadee el número dado de veces) . ¡Pruébenlo!
2. Desarrollen en Rectangle el método perimeter() . ¡Pruébenlo!
3. Desarrollen en Rectangle el método turn() (que hace que gire 90 grados a la derecha) . ¡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.

CEILING FUNCTION

Ustedes han sido contratados para construir un simulador para el diseñador de techos de ACM (Advanced Ceiling Manufacturers).

Los techos deben poder verse en forma de techo o en forma de árbol. La representación por defecto es de techo (secuencia de capas).

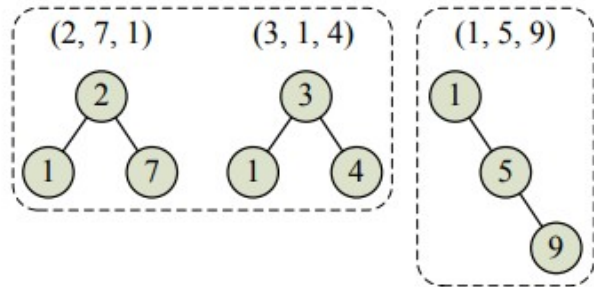
Los valores de las resistencias van entre 1 y 10. Cada valor de resistencia debe tener asociado un color (no se presentan los números)

Los techos se identifican con un color. Este color debe aparecer en los bordes de las capas del techo en sus dos representaciones.

Sólo se pueden adicionar o eliminar las capas de la parte superior del techo.

[De 2016 Problem C Ceiling Function)

NO DEBEN RESOLVER EL PROBLEMA DE LA MARATÓN



Implementando una nueva clase. Ceiling.

[En lab01.doc. Ceiling.java]

El objetivo es implementar una clase ya diseñada.

Ceiling
+ _(layers : int[], color : String) : Ceiling + showCeiling() : void + showTree() : void + add(layer : int) : void + delete() : void + move(x : int, y : int) : void

MINICICLOS

1. _(layers: int[], color:String)
2. add(layer)
delete()
3. move(x,y)
4. showTree()
showCeiling()

1. Revisen el diseño y clasifiquen los métodos en: constructores, analizadores y modificadores.
2. Desarrollen la clase `Ceiling` considerando los mini-ciclos. No olviden la documentación. Al final de cada mini-ciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes.

Definiendo y creando una nueva clase. **CeilingManager**

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

El objetivo es implementar diseñar e implementar una nueva clase.

Requisitos funcionales

- ☐ Crear un gestor de techos vacío
- ☐ Cambiar el modo de visualización del gestor: techo o árbol.
- ☐ Adicionar un nuevo techo dadas sus capas y color. El color debe permitir identificar el techo. No se permiten adicionar techos con las mismas capas.
- ☐ Eliminar un techo
- ☐ Adicionar y eliminar capas a un techo.
- ☐ Adicionar una capa a todos los techos
- ☐ Eliminar una capa de todos los techos
- ☐ Decidir si dos techos son equivalentes (tienen la misma forma de árbol)

NOTAS: Los techos no se pueden sobreponer.

Requisitos de interfaz

- ☐ Se debe presentar un mensaje de alerta cuando no se puedan adicionar más techos `showMessageDialog` de la clase `JOptionPane`.
- ☐ En caso que no sea posible realizar una de las acciones, debe generar un sonido de alerta.

1. Diseñen la clase, es decir, definan los métodos que debe ofrecer.
2. Planifiquen la construcción considerando algunos mini-ciclos.
3. Implementen la clase siguiendo los ciclos definidos. 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 `Ceiling`. Expliquen
5. Propongan un nuevo método para enriquecer esta clase.

Extendiendo una clase. **CeilingManager**

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

El objetivo es crear una nueva funcionalidad que permita dejar sólo un representante de los techos que son equivalentes: el de mínima resistencia. Los techos deben re-acomodarse en el espacio sin perder la forma en que están siendo visualizados.

Nuevos requisitos funcionales

1. Depurar gestor

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?