

PROGRAMACIÓN ORIENTADA A OBJETOS

Construcción. Clases y objetos.


2022-1

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 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

A. 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**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 de la figura?. 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`, revise el atributo `PI` (a) ¿Qué significa que sea `public`? (b) ¿Qué significa que sea `static`? (c) ¿Qué significa que sea `final`? (d) ¿De qué tipo de datos debería ser? ¿Por qué? Actualícenlo.

1 <http://docs.oracle.com/javase/8/docs/api/>
2 Menu: Tools-Project Documentation

- 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 círculos van a ser pequeños (diámetro menor a 100), ¿de que tipo deberían ser este atributo? (c) Si son grandes, pero no tanto (diámetro menor a 30000), ¿de que tipo deberían ser este atributo? (d) ¿qué restricción adicional tendrían este atributo? Refactoricen el código considerando (c) y expliquen claramente sus respuestas.
- ¿Cuál dirían es el propósito del proyecto “shapes”?

B. Manipulando objetos. Usando un objeto.

[En lab01.doc]

- Crean un objeto de cada una de las clases que lo permitan³. (a) ¿Cuántas clases hay? ¿Cuántos objetos crearon? (b) ¿Por qué?
- Inspeccionen el **estado** del objeto `:Circle`⁴, ¿Cuáles son los valores de inicio de todos sus atributos? Capture la pantalla.
- 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?
- 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. (d) Incluyan el logo original.

C. Manipulando objetos. Analizando y escribiendo código.

[En lab01.doc]

<pre>Circle face; Circle rEye,lEye; Rectangle mouth; //1 face=new Circle(); mouth=new Rectangle(); rEye=new Circle(); lEye=rEye; //2 face.changeSize(200); face.changeColor("yellow"); face.makeVisible(); //3</pre>	<pre>mouth.changeSize(10,100); mouth.changeColor("red"); mouth.moveVertical(120); mouth.makeVisible(); //4 rEye.changeSize(20); rEye.moveVertical(50); rEye.moveHorizontal(50); rEye.makeVisible(); //5 lEye.changeSize(20); lEye.moveVertical(50); lEye.moveHorizontal(140); lEye.makeVisible(); //6</pre>
--	---

- Lean el código anterior. (a) ¿cuál es la figura resultante? (b) Píntenla.
- 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.
- 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.

D. Extendiendo una clase. Circle.

[En lab01.doc y *.java]

1. Desarrollen en `Circle` el método `area()`. ¡Pruébenlo! Capturen una pantalla.
2. Desarrollen en `Circle` el método `duplicate()` (duplica su área). ¡Pruébenlo! Capturen dos pantallas.
3. Desarrollen en `Circle` el método `rainbow()` (va cambiando de color siguiendo los colores del arco iris). ¡Pruébenlo! Capturen dos pantallas.
4. Propongan un nuevo método para esta clase.
5. Generen nuevamente la documentación y revise la información de estos nuevos métodos. Capturen la pantalla.

E. Codificando una nueva clase. Dice

[En lab01.doc. Digit.java]

	<table><tr><th>Dice</th></tr><tr><td>+ <code>_(digit : byte) : Dice</code> + <code>get() : byte</code> + <code>next() : void</code> + <code>change(digit : byte) : void</code> + <code>change() : void</code> + <code>moveTo(x : int, y : int) : void</code> + <code>changeColor(color : String) : void</code> + <code>makeVisible() : void</code> + <code>makeInvisible() : void</code></td></tr></table>	Dice	+ <code>_(digit : byte) : Dice</code> + <code>get() : byte</code> + <code>next() : void</code> + <code>change(digit : byte) : void</code> + <code>change() : void</code> + <code>moveTo(x : int, y : int) : void</code> + <code>changeColor(color : String) : void</code> + <code>makeVisible() : void</code> + <code>makeInvisible() : void</code>	<p>Iterativo - Incremental</p> <p>Miniciclos</p> <ol style="list-style-type: none">1. <code>_(digit)</code> <code>get()</code>2. <code>next()</code> <code>change(digit)</code> <code>change()</code>3. <code>makeVisible()</code> <code>makeInvisible()</code>4. <code>moveTo(x,y)</code> <code>changeColor()</code>
Dice				
+ <code>_(digit : byte) : Dice</code> + <code>get() : byte</code> + <code>next() : void</code> + <code>change(digit : byte) : void</code> + <code>change() : void</code> + <code>moveTo(x : int, y : int) : void</code> + <code>changeColor(color : String) : void</code> + <code>makeVisible() : void</code> + <code>makeInvisible() : void</code>				

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.

F. Diseñando y codificando una nueva clase. DiceTaken

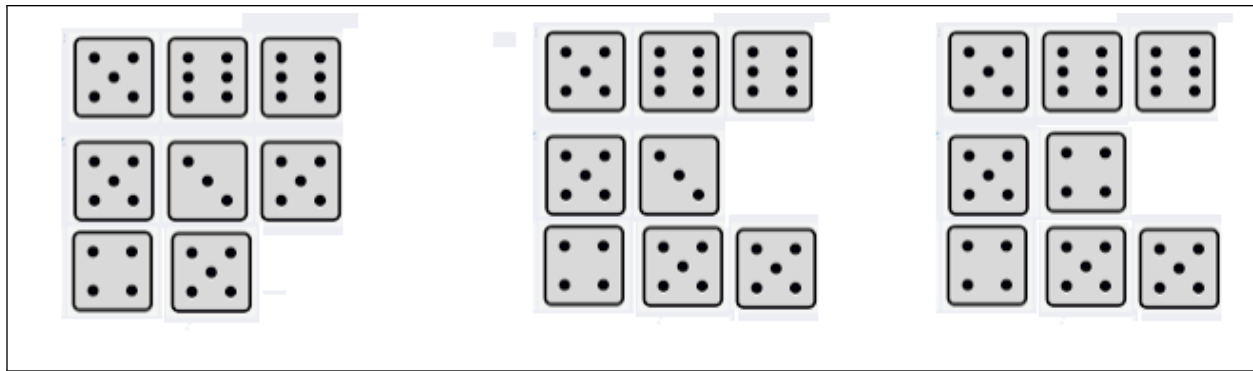
[En lab01.doc. DiceTaken.java]

El objetivo de este trabajo es programar una mini-aplicación para una **DiceTaken**⁷.

El juego DicesTaken es un juego de movimiento de dados que presentan un orden aleatorio inicial dentro de una cajita cuadrada con un espacio vacío. El juego consiste en maniobrar los dados para conseguir que todos los dados queden en orden consecutivo. Existen dos movimiento posibles deslizar un dado al espacio vacío o lanzar un dado.

En la gráfica se presenta un juego de 3x3; en el primer movimiento se desliza el dado de la posición (2,3) y en el segundo movimiento se lanza el dado de la posición (2,2).

⁷ Inspirado en taken



Requisitos funcionales

- Permitir iniciar el juego indicando el tamaño. Los dados toman valores y ubican al azar.
- Permitir rotar uno de los dados
- Permitir deslizar un dado al espacio vacío.

Requisitos de interfaz

- Los dados ordenados deben quedar de color verde y los desordenados de color rojo.
- Debe "sonar" y "parpadear" cuando se logre el objetivo: todos los dados ordenados.
- Debe construirse usando las figuras del paquete `shapes`
- 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 `DiceTaken`, es decir, definan los métodos que debe ofrecer.
2. Planifiquen la construcción definiendo algunos miniciclos. Recuerden que al final de cada miniciclo deben poder probar.
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 el paquete `shapes` y la clase `Dice`. Explique.

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?