

PROGRAMACIÓN ORIENTADA A OBJETOS

Introducción. Clases y objetos.

Agosto 2015

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 la práctica XP : **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**² 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 `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 tiene? ¿cuáles son privados y cuáles públicos? ¿por qué? ¿cuáles son constantes? ¿por qué? ¿cuáles son de clase? ¿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 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 del logo de su red social 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]

<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>
--	---

1. Revisen el código anterior e indiquen ¿cuál es la figura resultante? Píntenla.
2. Habiliten la ventana de código en línea⁶, 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 Circle el método zoom(char sign) (aumentar '+' o disminuir '-' una unidad su diametro) . ¡Pruébenlo!
2. Desarrollen en Circle el método area() . ¡Pruébenlo!
3. 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.

TRAGAMONEDAS

Desarrollando dos aplicaciones

Implementando una nueva clase. `SlotMachine`.

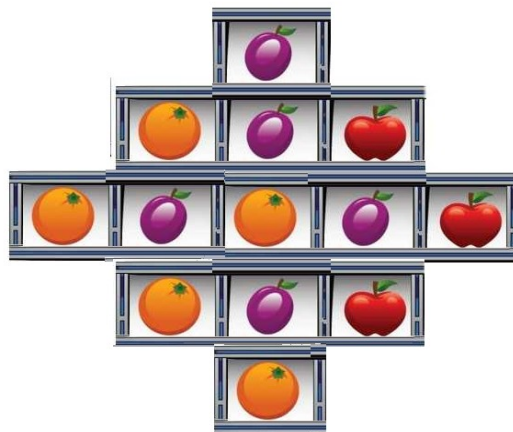
[En lab01.doc. `SlotMachine.java`]

 <p>Las posibles frutas son : naranja, ciruela, manzana y limón. Para ganar se deben tener las tres frutas iguales.</p>	<div>SlotMachine</div> <div><pre>+ _() : slotMachine + reset() : void + pull() : void + pull(times : int) : void + isWinningState() : boolean + percentageOfWinningStates() : int + makeVisible() : void + makeInvisible() : void + move(horizontal : int, vertical : int) : void</pre></div>	<p>Miniciclo 1</p> <pre>SlotMachine pull pull(times) isWinningState</pre> <p>Miniciclo 2</p> <pre>reset percentageOfWinning</pre> <p>Miniciclo 3</p> <pre>makeVisible makeInvisible move</pre>
--	--	---

- ¿Cuál es la probabilidad de ganar en esta máquina? Explique su respuesta.
- Clasifiquen los métodos en: constructores, analizadores y modificadores.
- Desarrollen la clase `SlotMachine` considerando los miniciclos. Al final de cada miniciclo realicen una prueba. Capturen las pantallas relevantes.
- ¿Cual es el porcentaje de estados ganadores después de hacer 1, 10, 100 y 1000 jugadas? Presente un análisis de los datos.

Diseñando e implementando una clase. `DiamondSlotMachine`.

[En lab01.doc. `SuperSlotMachine.java` `DiamondSlotMachine.java`]



El objetivo de esta clase es ofrecer una máquina tragamonedas bidimensional. Esta máquina tiene forma de diamante y puede tener diferentes dimensiones. En esta máquina se gana si se logran tener una fila, columna o diagonal de elementos iguales.

Requisitos funcionales

- Permitir crear un tragamonedas compuesto, indicando el tamaño.
- Permitir reiniciar el tragamonedas compuesto
- Permitir jugar (todos los tragamonedas giran)
- Permitir jugar indicando el tragamonedas con el que se desea jugar (1 ...n de arriba a abajo)
- Permitir consultar si se ha ganado
- Permitir consultar el porcentaje de juegos ganadores desde el último reinicio

Requisitos de interfaz

- Las operaciones se deben ofrecer como métodos públicos de la clase `DiamondSlotMachine`
- El tragamonedas compuesto debe “sonar” cada vez que llega a un estado ganador
- 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 `DiamondSlotMachine`, 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 `SlotMachine`. Explique.
5. Propongan un nuevo método para enriquecer el juego.

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 la práctica XP **Coding**. *All production code is [pair programmed](#)*, ¿por qué consideran que es importante?
4. ¿Cuál consideran fue su mayor logro? ¿Por qué? ¿Cuál consideran que fue su mayor problema? ¿Qué hicieron para resolverlo?
5. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?