PROGRAMACIÓN ORIENTADA A OBJETOS
Construcción. Clases y objetos.
2024-1
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 java[1].
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]

1. El proyecto "`shapes`" es una versión modificada de un recurso ofrecido por BlueJ. Para trabajar con él, bajen `shapes.zip` y ábranlo 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 tiene la clase `Rectangle`? (c) ¿Cuántos métodos ofrece la clase `Rectangle`? (d) ¿Cuáles métodos ofrece la clase `Rectangle` para que la figura cambie su tamaño (incluya sólo el nombre)?
4. En el **código** de cada clase está el detalle de la implementación. Revisen el código de la clase `Rectangle`. Con respecto a los atributos: (a) ¿Cuántos atributos realmente tiene? (b) ¿Cuáles atributos determinan su forma?. 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 `Rectangle`, revise el atributo `EDGES` (a) ¿Qué significa que sea `public`? (b) ¿Qué significa que sea `static`? (c) ¿Qué significaría que fuera `final`? ¿Debe serlo? (d) ¿De qué tipo de dato debería ser (byte, short, int, long)? ¿Por qué? (e) Actualícenlo.

---

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

2 Menu: Tools-Project Documentation

7. En el código de la clase `Rectangle` revisen el detalle del tipo del atributo `width` (a) ¿Qué se está indicándo al decir que es `int`?. (b) Si `width` fuera `byte`, ¿cuál sería el rectangulo más grande posible? (c) y ¿si fuera `long`? d) ¿qué restricción adicional deberían tener este atributo? (e) Refactoricen el código considerando (d).
8. ¿Cuál dirían es el propósito del proyecto "`shapes`"?

B. Manipulando objetos. Usando un objeto.

[En lab01.doc]

1. Creen un objeto de cada una de las clases que lo permitan[3]. (a) ¿Cuántas clases hay? (b) ¿Cuántos objetos crearon? (c) ¿Quién se crea de forma diferente? ¿Por qué?
2. Inspeccionen los creadores de cada una de las clases. (a) ¿Cuál es la principal diferencia entre ellos? (b) ¿Qué se busca con la clase que tiene el creador diferente?
3. Inspeccionen el **estado** del objeto `:Triangle`[4], (a) ¿Cuáles son los valores de inicio de todos sus atributos? (b) Capturen la pantalla.
4. Inspeccionen el **comportamiento** que ofrece el objeto `:Triangle`[5]. (a) Capturen la pantalla. (b) ¿Por qué no aparecen todos los que están en el código?
5. Construyan, con "`shapes`" sin escribir código, una propuesta de la imagen del logo de su marca de carro 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]

```
Rectangle face;
Circle pointOne;
Circle pointTwo;
//1
face=new Rectangle();
pointOne = new Circle();
pointTwo = new Circle();
//2
face.changeColor("black");
face.changeSize(100,100);
face.makeVisible();
//3
```

```
pointOne.changeColor("red");
pointOne.changeSize(10);
pointTwo=pointOne;
//4
pointOne.moveHorizontal(70);
pointOne.moveVertical(20);
pointTwo.moveHorizontal(120);
pointTwo.moveVertical(70);
//5
pointOne.makeVisible();
pointTwo.makeVisible();
//6
```

1. Lean el código anterior. (a) ¿cuál creen que es la figura resultante? (b) Píntenla.
2. Habiliten la ventana de código en línea[6], escriban el código. Para cada punto señalado indiquen: (a) ¿cuántas variables existen? (b) ¿cuántos objetos existen? (no cuenten ni los objetos `String` ni el objeto `Canvas`) (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.

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 el área del círculo) . ¡Pruébenlo! Capturen dos pantallas.
3. Desarrollen en `Circle` el método `bounce(times:int, height:int)` (salta el número de veces indicado en times. Las alturas de los saltos se seleccionan al azar con valores menores de height [7]). ¡Pruébenlo! Capturen dos pantallas.
4. Desarrollen en `Circle` un nuevo creador que dada una posición, crea circulos negros de radio 50.¡Pruébenlo! Capturen una pantalla.
5. Propongan un nuevo método para esta clase. Desarrollen y prueban el método.
6. Generen nuevamente la documentación y revise la información de estos nuevos métodos. Capturen la pantalla.

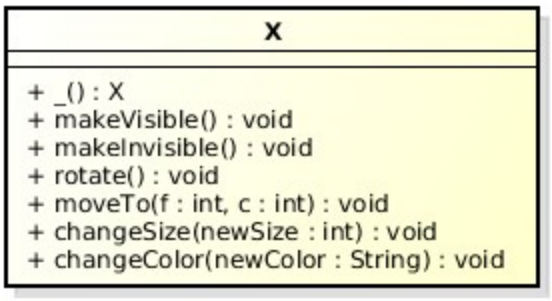
E. Usando un paquete. shapes

[En lab01.doc y *.java]

En este punto vamos a crear tres figuras usando el paquete shapes: misionero, canibal y barca. El diseño gráfico lo definen ustedes. Estos son algunos ejemplos.



Las tres figuras deben ofrecer los siguientes métodos.



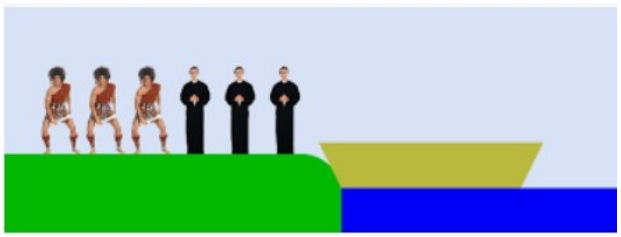
Mini-ciclo: 1
`_():X`
`makeVisible()`
`makeInvisible()`
Mini-ciclo: 2
`changeSize`
`changeColor`
Mini-ciclo: 3
`moveTo`
`rotate`

Construya las clases en el siguiente orden: `Cannibal,Missionary, Boat`
Para cada clase:

1. Inicie la construcción únicamente con los atributos. Adicione pantallazo con los atributos.
2. Desarrollen la clase considerando los 3 mini-ciclos. Al final de cada mini-ciclo realicen dos pruebas indicando su propósito. Capturen las pantallas relevantes.

7 Consulte la clase Math del API java

CANIBALES Y MISIONEROS

Tres misioneros se perdieron explorando una jungla. Separados de sus compañeros, sin alimento y sin radio, solo sabían que para llegar a su destino debían ir siempre hacia adelante. Los tres misioneros se detuvieron frente a un río que les bloqueaba el paso, preguntándose que podían hacer. De repente, aparecieron tres caníbales llevando un bote, pues también ellos querían cruzar el río. Ya anteriormente se habían encontrado grupos de misioneros y caníbales, cada uno respetaba a los otros, pero sin confiar en ellos. Los tres caníbales deseaban ayudar a los misioneros a cruzar el río, pero su bote no podía llevar más de dos personas a la vez y los misioneros no querían que los caníbales les superaran en número para evitar que los devoraran.[8] ¿Cómo podrían atravesar el rio?	

F. Definiendo y creando una nueva clase. MissionariesCannibals.

[En lab01.doc. MissionariesCannibals.java]

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

Requisitos funcionales

- Crear el estado inicial
- Subir personas en la barca indicando las personas que se deben subir
- Mover la barca hacia la otra orilla
- Bajar las personas de la barca

Requisitos de interfaz

- Las personas únicamente se identifican por su tipo: misionero o canibal
- En caso que no sea posible realizar una de las acciones, debe generar un mensaje de error. Use JoptionPane.

1. Diseñen la clase, es decir, definan los métodos que debe ofrecer.
2. Planifiquen la construcción considerando algunos mini-ciclos.
3. Implementen la clase . Al final de cada mini-ciclo realicen una prueba indicando su propósito. Capturen las pantallas relevantes.
4. Indiquen las extensiones necesarias para reutilizar las clases de las figuras y el paquete `shapes`. Expliquen.

G. De python a java

[En lab01.doc]

En este punto vamos a evaluar el video DE PYTHON A JAVA en la encuesta preparada con ese objetivo. ¿En cuáles puntos estuvieron en desacuerdo[9]? Expliquen.

BONO. Nuevos requisitos funcionales. MissionariesCannibals.

[En lab01.doc. MissionariesCannibals.java]

El objetivo de este trabajo es extender la mini-aplicación **MissionariesCannibals.**

Nuevos requisitos funcionales

- Subir las personas a la barca (la máquina decide que personas subir). Explique la estrategia.
- Deshacer el último movimiento

1. Diseñen, es decir, definan los métodos que debe ofrecer.
2. Implementen los nuevos métodos . Al final de cada método realicen una prueba indicando su propósito. Capturen las pantallas relevantes.

---

8 https://www.psicoactiva.com/puzzleclopedia/misioneros-y-canibales/

9 Uno evaluó de acuerdo y otro en desacuerdo.

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?