ENTORNOS DE DESARROLLO - EXAMEN FINAL 3º EVALUACIÓN

C.F.G.S 1º D.A.M. 20-Marzo-2021 I.E.S. Fernando Aguilar Quignón

- Al entregar tu examen, debes ser capaz de explicar cada ejercicio. No está permitida la copia de compañeros u otro medio. No puedes comunicarte con tus compañeros de ninguna manera mientras el examen está en progreso. En caso de detectar copias, el examen se puntuará con un 0.

- Tu pantalla está siendo monitorizada y solo puedes tener abierto este documento, NetBeans para el ejercicio de refactorización, la aplicación "DIA" para desarrollar los diferentes diagramas y LibreOffice Writer para las preguntas teóricas.

- La entrega del examen consistirá en un proyecto Java con el código del ejercicio de refactorización, los diagramas realizados con la aplicación "DIA", y un documento pdf con las respuestas a las preguntas teóricas.

Ejercicio 1. (3,5 puntos) - Refactorización.

Aplica las refactorizaciones que consideres oportunas al siguiente código. Crea todas las clases necesarias utilizando los conceptos estudiados en clase que permitan que el programa esté preparado ante cambios futuros.

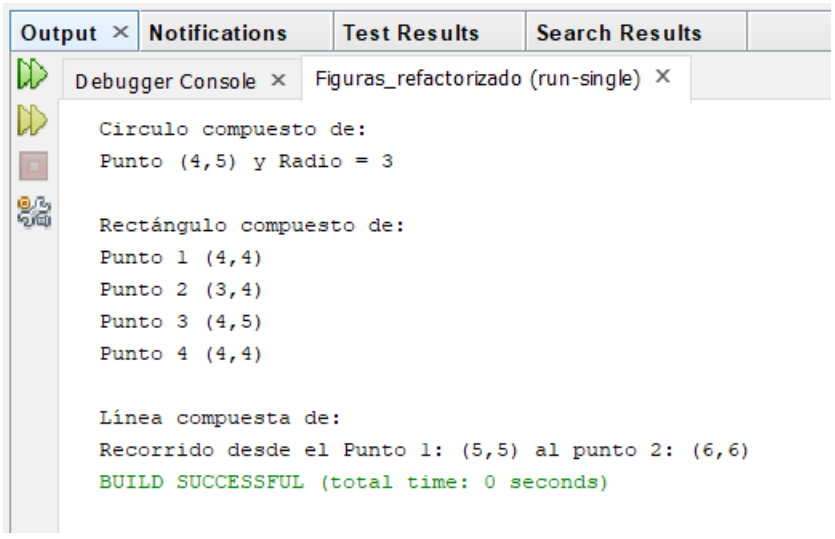
```
class Figura {
    final static int TIPO_LINEA = 0;
    final static int TIPO_RECTANGULO = 1;
    final static int TIPO_CIRCULO = 2;

    int tipoFigura;
    Punto p1;
    Punto p2;

    int radio;
}
class CADApp {
    void dibujaFiguras(Grafica grafica, Figura figuras[]) {
        for (int i = 0; i < figuras.length; i++) {
            switch (figuras[i].getTipo()) {
                case Figura.TIPO_LINEA:
                    grafica.drawLine(figuras[i].getP1(), figuras[i].getP2());
                    break;
                case Figura.TIPO_RECTANGULO:
                    //draw the four edges.
                    grafica.drawLine(...);
                    grafica.drawLine(...);
                    grafica.drawLine(...);
                    grafica.drawLine(...);
                    break;
                case Figura.TIPO_CIRCULO:
                    grafica.drawCircle(figuras[i].getP1(), figuras[i].getRadio());
                    break;
            }
        }
    }
}
```

Ayuda 1: Los métodos “drawLine” y “drawCircle” no debes implementarlos, en su lugar, considera el método “draw” que servirá para definir de forma genérica como se mostrará por pantalla con un mensaje de texto cada tipo objeto.

Ayuda 2: Crea una clase main de nombre “TestFiguras”, que cree los objetos necesarios, guarde dichos objetos en un vector y llame al método estático “dibujaFiguras”. Sigue la siguiente plantilla:

```
public class TestFiguras {

    public static void main(String[] args) {

        // Creación de los objetos

        // Guardamos los objetos en un vector

        // Llamamos al método estático dibujaFiguras

    }

    public static void dibujaFiguras(Figura[] figuras){

        // Método equivalente a dibujaFiguras del enunciado ya refactorizado

    }
}
```

Ayuda 3: La salida por pantalla del programa debe ser igual a la siguiente:



Ejercicio 2. (2 puntos) - Diagramas de casos de uso.

Realiza los diagramas de caso de uso correspondientes al siguiente texto que describe los requisitos para diseñar una aplicación que gestione los tramites a realizar en una clínica veterinaria.

-La clínica veterinaria debe gestionar los datos de contacto de todos sus clientes como pueden ser: Nombre, Apellidos, DNI, Fecha de nacimiento, Teléfono o Email. Estos datos de los clientes podrán ser dados de alta, baja o modificados y son introducidos y gestionados por los auxiliares veterinarios, que ejercen las funciones administrativas.

-Además se debe gestionar la información de cada una de las mascotas de las que es dueño cada cliente. Obviamente, cada cliente puede tener más de una mascota, pero cada mascota solo puede pertenecer a un único cliente. La aplicación además de gestionar los movimientos de mascotas debe permitir cambiar el dueño (cliente) de una mascota por otro.

-Para poder gestionar tanto los datos de los clientes como los datos de las mascotas, el auxiliar debe hacer login en el sistema.

-Cada vez que un veterinario recibe una mascota en la clínica debe crear en la aplicación una consulta. Al crear la consulta se debe identificar al cliente, y a la mascota. Esta consulta incluirá los datos básicos de la consulta, y lo habitual será almacenar también el tiempo de duración de la consulta. Para calcular el tiempo de duración de la consulta, el veterinario tendrá un botón en la aplicación donde pueda pulsar una vez comenzada la consulta para calcular el tiempo a modo de cronómetro. Ese mismo botón servirá para que el cronómetro deje de contar.

-Desde el módulo anterior, una vez terminada la consulta, se podrá calcular el pago del cliente, y recetar algún medicamento a la mascota.

-En caso de que el animal se quede ingresado en la clínica, el cliente debe ser capaz de acceder al estado en tiempo real del animal. Además, tendrá también la posibilidad de ver a su mascota con una cámara que tendrá el animal colocada, donde podrá ver su situación actual y movimientos. Dentro del seguimiento del estado en tiempo real de la mascota, el cliente también podrá ver las recetas del veterinario, que se almacenan en un gestor de contenidos que ya está en funcionamiento en la clínica veterinaria.

-El cliente tiene también la posibilidad de obtener un histórico de todas las consultas que ha recibido cualquiera de sus mascotas.

-Desde su interfaz, el cliente debe poder hacer el pago del servicio realizado..

-La aplicación debe identificar al cliente para que este pueda realizar cualquiera de las funcionalidades que tiene disponible.

Nota: Se recomienda descomponer el problema por roles.

Ejercicio 3. (1,5 puntos) - Diagramas de clases.

Realiza el diagrama de clases de la siguiente aplicación para la gestión de pedidos. La aplicación deberá manejar clientes (se guarda su nombre, dirección, teléfono y email), que pueden realizar pedidos de productos, de los cuales se anota la cantidad de stock, el nombre y el precio. Un cliente puede tener una o varias cuentas para el pago de los pedidos. Cada cuenta está asociada a un número de tarjeta de crédito, y tiene una cierta cantidad disponible de dinero, que el cliente debe aumentar periódicamente para poder realizar nuevos pedidos.

Un cliente puede empezar a realizar un pedido sólo si tiene alguna cuenta con dinero disponible. Al realizar un pedido, un cliente puede agruparlos en pedidos simples o compuestos. Los pedidos simples están asociados a una sola cuenta de pago y (por restricciones en la distribución) contienen un máximo de 20 unidades del mismo o distinto tipo de producto. A su vez, un pedido compuesto contiene dos o más pedidos, que pueden ser simples o compuestos. Como es de esperar, el sistema debe garantizar que todos los pedidos simples que componen un pedido compuesto se paguen con cuentas del mismo cliente. Además, sólo es posible realizar peticiones de productos en stock.

Existe una clase controladora de pedidos, que es responsable de cobrar pedidos, servir pedidos y confirmar pedidos. El cobro de los pedidos se hace una vez al día, y el proceso consiste en comprobar todos los pedidos pendientes de cobro, y cobrarlos de la cuenta de pago correspondiente. Si una cuenta no tiene suficiente dinero, el pedido se rechaza (si es parte de un pedido compuesto, se rechaza el pedido entero). Una vez que el pedido está listo para servirse, se ordena su distribución, y una vez entregado, pasa a estar confirmado.

La entidad pedido tendrá un campo para almacenar el total del pedido y otro para almacenar el estado del pedido que podrá ser “pendiente”, “pagado”, “servido”, “confirmado” y “rechazado”.

Considera además que un pedido tiene líneas de pedido (la entidad de líneas de pedido guardará el número de líneas), y del mismo modo, una línea de pedido tiene uno o más productos, en ambos casos las partes pueden existir por sí mismas.

Ejercicio 4. (3 puntos) - Conceptos teóricos.

Sobre la refactorización…

a. Explica con detalle la metáfora de los dos sombreros de Kent Beck. ¿Qué tareas se realizan bajo cada sombrero?. (0,5)

b. Indica el nombre de los “malos olores” pertenecientes a la categoría de inflados. (0,25)

c. Indica el nombre de los “malos olores” pertenecientes a la categoría de abusadores de la orientación a objetos. (0,25)

Sobre los casos de uso…

d. Definición de caso de uso. (0,25)

e. Explica con detalle y ejemplos los casos de “inclusión”, “extensión” y “puntos de extensión” de la notación UML para diagramas de casos de uso. (0,5)

Sobre los diagramas de clases...

f) ¿Qué es el modelado? (0,25)

g) Además de los diagramas de casos de uso, y los diagramas de clases, ¿cuál es el nombre del tercer diagrama más usado para modelar software con la notación UML? (0,25)

h) ¿Qué son las clases asociativas? (0,25)

Sobre la ingeniería inversa…

i) Definición de ingeniería inversa. (0,25)

j) Indica el nombre de los tres beneficios de la ingeniería inversa. (0,25)