Examen Final de Junio

Proyecto de Análisis y Diseño de Software (2010/2011)

Se quiere extender la aplicación de gestión del bar que has realizado en la asignatura con funcionalidad adicional que permita la creación y gestión de cuentas para clientes. De esta manera, la aplicación deberá permitir registrar clientes habituales del bar, y asignarles una cuenta. Las cuentas permitirán el pago diferido de las consumiciones, hasta un máximo de dinero que se establece al crear la cuenta. El objetivo de esta extensión es que los clientes no tengan que pagar inmediatamente al realizar una consumición, sino que puedan hacerlo de manera periódica.

Más en detalle, la aplicación deberá guardar datos relativos a los clientes, tales como su nombre y apellidos, teléfono, dirección, código postal y e-mail. Cada cliente registrado puede tener una cuenta como máximo. Una cuenta se crea con un importe máximo a acordar entre el gestor y el cliente al abrir la cuenta, así como un identificador asignado automáticamente por el sistema (y que debe proporcionar el cliente cuando quiere cargar una consumición a su cuenta). Las cuentas pueden tener un saldo positivo (si el cliente realizó ingresos en la cuenta que superan el total de sus consumiciones), negativo (si se han realizado consumiciones que en total superan los ingresos realizados en la cuenta) o cero. En caso de saldo negativo, éste ha de ser en valor absoluto menor que el tope prefijado para la cuenta. El sistema no debe permitir cargar una consumición a una cuenta si ésta excede su tope prefijado. El dueño de una cuenta con saldo positivo o cero puede desactivarla para evitar que se carguen consumiciones. En cualquier momento, una cuenta inactiva puede activarse otra vez. Finalmente, una cuenta con saldo positivo o cero (activa o no) puede cancelarse, eliminándose del sistema. En este caso, si el saldo es positivo, se le devolverá al cliente el importe restante.

Las cuentas pueden tener un conjunto de clientes autorizados a realizar consumiciones a su cargo. El dueño de la cuenta está siempre autorizado a cargar consumiciones en su cuenta, pero se quiere además que el dueño de la cuenta pueda incluir a otros clientes en su cuenta, que están autorizados a cargar consumiciones. Sólo el dueño de una cuenta puede realizar ingresos en la misma, así como activarla, desactivarla o cancelarla. El sistema deberá mantener un histórico de los ingresos realizados en cada cuenta (fecha e importe de cada ingreso).

Para dar mayor comodidad a los clientes se quiere ofrecer dos tipos especiales de cuentas (además de las normales). La primera lleva asociada una tarjeta de crédito. De esta manera, además de en efectivo, el cliente puede hacer ingresos a la cuenta con la tarjeta cuyos datos (número y pin) tiene guardados el sistema. El segundo tipo especial de cuenta es el de pago periódico. Este tipo de cuentas llevan asociadas también una tarjeta, pero además se configuran con un importe y un periodo de tiempo en días. En este caso el importe especificado se ingresa periódicamente en la cuenta usando la tarjeta asociada.

IMPORTANTE: Limita tus respuestas a lo que dice estrictamente el enunciado. Es posible que tengas que hacer alguna suposición sobre algún detalle que no incluye el enunciado. Puedes hacerlo, pero siempre de manera bien fundamentada, razonable y sin contradecir el enunciado. Documenta tales suposiciones en tu respuesta.

CONTESTA A CADA APARTADO EN HOJAS SEPARADAS

Apartado 1. (3 puntos)

a) Dibuja el diagrama de casos de uso que representa esta nueva funcionalidad. No es necesario que dibujes los casos de uso de la aplicación existente que ya hiciste en la primera entrega. Inclúyelos sólo si necesitas relacionarlos de alguna manera con los de la nueva funcionalidad.

b) Describe detalladamente el caso de uso necesario para crear una cuenta (normal) asociada a un nuevo cliente.

Apartado 2. (5 puntos)

a) Dibuja el diagrama de clases que representa esta nueva funcionalidad. Incluye en el diagrama los atributos y métodos necesarios para implementar la funcionalidad. No es necesario que incluyas constructores, getters o setters. Tampoco es necesario que incluyas las clases de la aplicación ya existente, salvo si necesitas relacionarlas de alguna manera con las clases encargadas de la nueva funcionalidad.

b) Dibuja el diagrama de transición de estados asociado a la clase que gestiona una cuenta normal de cliente.

c) Dibuja el diagrama de secuencia que refleje el proceso de crear una cuenta normal asociada a un nuevo cliente, y realizar un ingreso en la misma.

Apartado 3. (2 puntos)

Responde brevemente a las siguientes preguntas:

a) En la fase de pruebas de un proyecto software, es importante distinguir entre *verificación* y *validación*. Explica la diferencia entre ambos términos.

b) Al preparar las pruebas unitarias con JUnit, normalmente se utiliza un método anotado con `@Test` que contiene expresiones `Assert.assertEquals(e1,e2)` o similares, pero, ¿cómo se comprueba que un método lanza una excepción? ¿Ofrece JUnit algún mecanismo que facilite este tipo de pruebas?

c) Escribe las pruebas unitarias para la siguiente clase:

```
public class Punto {

   private float x, y;

   public Punto (float x, float y) {
        this.x = x;
        this.y = y;
   }

   public float getX () { return x; }
   public float getY () { return y; }
   public void  setX (float x) { this.x = x; }
   public void  setY (float y) { this.y = y; }

   public boolean desplazar (float distancia_x, float distancia_y) {
        boolean desplazado = false;
        if (x + distancia_x > 0) {
           x += distancia_x;
           desplazado = true;
        }
        if (y + distancia_y > 0) {
           y += distancia_y;
           desplazado = true;
        }
        return desplazado;
   }
}
```

d) En la librería *Swing*, explica brevemente qué es un gestor de *layout* y cómo se utiliza. Pon un ejemplo de uso de cualquier gestor de *layout* que hayas usado en tu aplicación.