Pablo Puigvert

ATIX LABS

Evaluación Developer SR\_ Semi SR

## Resolución de problemas de programación y diseño

**A)**

* ¿Le parece que está bien diseñada la clase?

No está bien diseñada la clase pero funciona.

* ¿la cambiaría o agregaría clases nuevas?

Cambiaria:

Los atributos de la cuenta los asignaría todos privados

CTA\_CORRIENTE y CAJA\_AHORRO son tipos de cuenta con atributos diferentes, así que uso herencia

Convierto la clase Cuenta en interfaz abstracta

El atributo numeroCuenta seria String

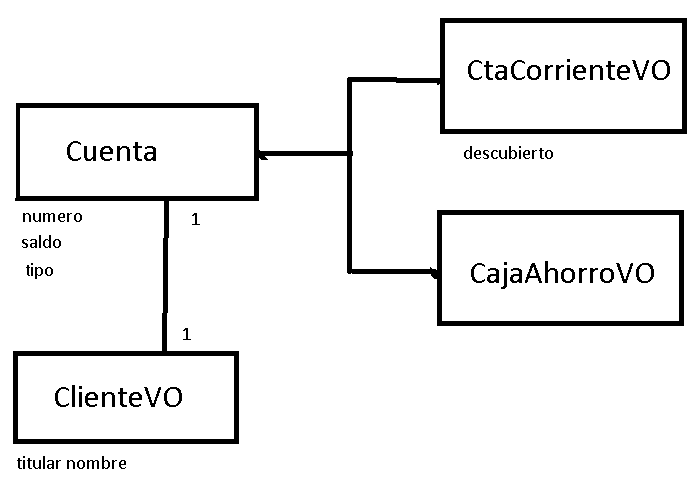
El nombre titular lo pondría en una clase ClienteVO que representa los atributos de un cliente y no de la cuenta

El segundo constructor no bloquea la posibilidad de crear erróneamente una cta corriente con desceunta acordado seteado por el frontend

La función extraer en caso de dar un error en tiempo de ejecución devolvería a la clase superior que la llama erroneamante saldo insuficiente , por lo tanto la convertiría en funciona devolviendo un objeto y manejo el error mediante un atributo del mismo objeto

* Si cambiaría el diseño, exponga todo lo que cambiaría con un diagrama de clases de

UML



* ¿Es seguro hacer estos cambios? ¿Por qué? ¿Qué precauciones tomaría?

Dependiendo del avance del proyecto, ya que no se sabe quiénes están utilizando hoy en día el constructor y las funciones desde la capa de servicios

* ¿Agregaría getters y setters? ¿Cuáles? ¿Por qué?

Agregaría getters y setters, pero numero de cuenta solo getter, ya que el numero de una cta no se modifica, tampoco el tipo CC o CA se modifica así que solo getter

* ¿Su solución usa algún patrón de diseño? ¿Cuál?

Abstract Factory

Interface

Singleton

Factory Method

**B) Código.**



## 2. Aspectos conceptuales

* Se utiliza para cambiar el comportamiento de una clase, se pueden definir familias de algoritmos, encapsularlos, siendo independiente la toma de algoritmo desde los clientes que lo quieran utilizar.
* Necesitamos mínimo dos clases donde asignemos la lógica diferente, para luego desde nuestras clases cliente, asignar la clase según el algoritmo adecuado
* Se puede utilizar Template Method
* Código de Template Method:

Public abstract class Perro

String nombre

abstract boolean ladra;

abstract boolean corre;

Public class Ovejero extendes Perro

Public boolean ladra{

Retunr true

}

Public boolean corre{

Return true

}

Public class Golden extendes Perro

Public boolean ladra{

Retunr false

}

Public boolean corre{

Return false

}

Demo:

Public static void main

Perro perro = null;

System.out.println(“seleccione perro : 1 ovejero 2 Golden”);

If(choice == 1)

Perro = new Ovejero();

If(choice == 2)

Perro = new Golden();

* Las ventajas de escribir pruebas unitarias antes de escribircodigo:

Orientamos el código a desarrollar a darnos el resultado y comportamiento exacto según el requerimiento

Podemos identificar controles necesarios en el desarrollo

Podemos identificar las funcionalidades para luego realizar un agrupamiento de módulos

* Observer: se utilizar para informar a otra clase el cambio de un atributo

La ventaja es que podemos “atar” una notificación de cambio a otras clases que requieren enterarse cuando hay una acción que lo requiera

## 3. Bases de datos y SQL

A-

SELECT U.username

FROM USUARIO U , PERSONA P

WHERE U.id = P.idUsuario

And P.nombre like “Jorg%”

B-

SELECT month(fechaNac)

FROM PERSONA

GROUP BY month(fechaNac)

HAVING COUNT > 10