Nombre y apellido: Maximiliano Nielson Legajo: N-1218/1

Ingeniería de Software 2
L.C.C. – F.C.E.I.A. – U.N.R.

Cristiá, Krapf, Scandolo
Viernes 1 de diciembre de 2023
Duración aproximada 3 hs.

Recuperatorio Primer Parcial

Nota: La interpretación de las consignas es parte del examen. El parcial se aprueba con al menos dos problemas bien y no menos del 65% de la puntuación. Problemas parcialmente correctos no necesariamente suman puntos.

Se evalúa: DBOI, DTAD, DOO, documentación de diseño.

Problemas

1. Realice un análisis de ventajas y desventajas entre el uso de la herencia de interfaces y de la composición de objetos.

2. Suponga que en cierto diseño existe un módulo, M, con algunos procedimientos en su interfaz. Digamos que es necesario extender la funcionalidad de algunos de esos procedimientos sin perder el módulo original y sin tener que duplicar código. Extender la funcionalidad de un método significa que hay que ejecutar código antes de llamar al método, o después de llamarlo, o en ambos casos. Además, la aplicación debe interactuar con las instancias de M y con las del módulo que tiene la funcionalidad extendida de la misma forma. Muestre cómo sería su diseño para este caso.

3. Considere que dispone de un módulo que abstrae el funcionamiento básico de una cuenta bancaria (número de cuenta, depositar, extraer, solicitar el saldo). Extienda ese módulo con lo siguiente:

   (a) La operación de transferencia de una suma de dinero de una cuenta a otra.

   (b) La operación que permite guardar en almacenamiento secundario el saldo de una cuenta asociado a su número de cuenta.

   (c) La operación que permite leer de almacenamiento secundario el saldo de una cuenta bancaria y crear un objeto del tipo correspondiente.

   (d) Operaciones como las dos anteriores pero para una lista de cuentas.

   Documente el diseño con la especificación de interfaces y la guía de módulos.

4. En un sistema operativo las contraseñas de los usuarios se pueden encriptar con dos algoritmos diferentes. Uno de los dos algoritmos es el que se usa por defecto pero cada usuario puede cambiarlo por el otro cuando crea su cuenta. El nombre de usuario y su contraseña se guardan en un archivo de texto ubicado en `/etc`.

   Claramente los requerimientos sugieren dos ítem de cambio. Muestre la especificación de interfaces y la guía de módulos de un diseño que tenga en cuenta los requerimientos y los ítem de cambio.

Nielsen Maximiliano legajo: N-1218/1

1) La herencia de interfaces tiene la ventaja de que se puede usar el supertipo en las interfaces clientes y de esta forma utilizarla sin necesidad de cambios si se agrega un subtipo nuevo. Esto permite abstraerse aun mas de la implementacion. Este mecanismo tiene la desventaja de ser estatico y por lo tanto solo se puede determinar en tiempo de compilacion. Ademas un abuso de esta lleva a un codigo mas sensible a los cambios, ya que un cambio en un supertipo podria modificar a muchos subtipos y requerir su retesteo.

La composicion de objetos tiene como ventaja que es dinamica, es decir se da en tiempo de ejecucion. Esto permite que el sistema cambie como funciona sin requerir recompilarlo. Tambien sirve en gran medida para no tener que repetir codigo entre herederos de la misma clase.

Como desventaja, presenta que, al determinarse en tiempo de ejecucion es dificil saber como se comportará hasta la misma. Ademas puede impedir el uso de algunas herramientas de analisis de codigo.

2)

```
MODULE  MAbstracto
EXPORTS operacion ()
        ...
```

```
MODULE  M  INHERITHS FROM MAbstracto
```

```
MODULE  MDecorado INHERITHS FROM MAbstracto
EXPORTS setM (i MAbstracto)
```

```
MODULE  Cliente
IMPORTS MAbstracto
EXPORTS operar (i MAbstracto)
        ...
```

Se crea el modulo abstracto con exactamente la misma interfaz de M para despues poder usar decoradores que usan la comunicacion con el modulo M para poder utilizar sus operaciones.

Nielsen
Hoja 2 de 4

3)

MODULE	CuentaBancaria
IMPORTS	NumeroCuenta , Monto
EXPORTS	setNumeroCuenta (i NumeroCuenta) numeroCuenta (): NumeroCuenta depositar (i Monto) extraer (i Monto) saldo (): Monto

MODULE	GestorDeTransferencias
IMPORTS	CuentaBancaria , Monto
EXPORTS	transferir (i CuentaBancaria, i CuentaBancaria, i Monto)

MODULE	AlmacenamientoSecundario
IMPORTS	Identificador, Value , Bool
EXPORTS	almacenar (i Identificador, i Value) primero () identificador (): Identificador Valor (): Value hayMas (): Bool Siguiente () eliminar ()

MODULE	AlmacenadorDeCuenta
IMPORTS	AlmacenamientoSecundario, CuentaBancaria
EXPORTS	setAlmacenamiento (i AlmacenamientoSecundario) almacenarSaldo (i CuentaBancaria)

MODULE	CreadorDeCuentas
IMPORTS	AlmacenamientoSecundario, NumeroCuenta, CuentaBancaria
EXPORTS	setAlmacenamiento (i AlmacenamientoSecundario) elegirCuenta (i NumeroCuenta) obtenerCuenta (): CuentaBancaria

MODULE Almacenador Multiple
IMPORTS AlmacenadorDeCuenta, List(CuentaBancaria), CuentaBancaria
EXPORTS setAlmacenador (i AlmacenadorDeCuenta)
almacenar (i List(CuentaBancaria))

MODULE CreadorMultiple
IMPORTS CreadorDeCuentas, List(NumeroCuenta), NumeroCuenta, List(CuentaBancaria), CuentaBancaria
EXPORTS setCreador (i CreadorDeCuentas)
elegir (i List(NumeroCuenta))
crear (): List(CuentaBancaria)

MODULE NumeroCuenta INHERITHS FROM Identificador

MODULE Monto INHERITHS FROM Valor

GUIA DE MODULOS

1. CuentaBancaria
   1 secreto: Oculta la implementacion de una cuenta bancaria y como esta almacena su saldo
   Funcion: Sirve para representar una cuenta bancaria
   setNumeroCuenta (i NumeroCuenta) toma un numero de cuenta y lo establece como el asociado a esta cuenta bancaria
   numeroCuenta (): NumeroCuenta devuelve el numero de cuenta asociado
   depositar (i Monto) toma un Monto y lo añade al saldo de la cuenta
   extraer (i Monto) toma un Monto y lo extrae del saldo de la cuenta
   saldo (): Monto devuelve el saldo de la cuenta.

2. GestorDeTransferencias
   secreto: Oculta como se realizan las transferencias entre cuentas bancarias
   Funcion: Sirve para realizar transferencias entre 2 cuentas bancarias
   transferir (i CuentaBancaria, i CuentaBancaria, i Monto) toma 2 cuentas bancarias y un Monto, siendo la primera a la cual se le debita el monto y la segunda a cual se le acredita

3. AlmacenamientoSecundario
   secreto: Oculta los detalles de almacenamiento secundario se usa y como se almacena los datos de el mismo.
   Funcion: Provee una interfaz abstracta para utilizar el almacenamiento secundario permitiendo almacenar valores en base a un identificador.
   almacenar (i Identificador, i Valor) toma un identificador y un valor. Asocia el identificador con el valor y lo almacena
   primero () mueve el iterador al primer valor almacenado
   identificador (): Identificador devuelve el identificador del elemento actual
   valor (): Valor devuelve el valor del elemento actual

hayMas(): Bool devuelve un booleano indicando si hay mas elementos
siguiente() mueve el iterador al siguiente elemento
eliminar() elimina el elemento actual

4. AlmacenadorDeCuenta
secreto: Oculta como se utiliza el almacenamiento secundario para almacenar el saldo de una cuenta
funcion: Permite almacenar el saldo de una cuenta en el almacenamiento secundario
setAlmacenamiento (i AlmacenamientoSecundario) toma un almacenamiento secundario y lo establece como el que se va a utilizar
almacenarSaldo (i CuentaBancaria) toma una cuenta bancaria y almacena su saldo asociado al numero de cuenta

5. CreadorDeCuentas
secreto: Oculta como se crean las cuentas en base al almacenamiento secundario
funcion: sirve para crear instancias de cuentas bancarias si estas estan almacenadas en el almacenamiento secundario.
setAlmacenamiento (i AlmacenamientoSecundario) toma un almacenamiento secundario y lo establece como el que se va a utilizar
elegirCuenta (i NumeroCuenta) toma un numero de cuenta y la busca en el almacenamiento secundario. Causa una excepcion si no se encuentra
obtenerCuenta(): CuentaBancaria devuelve una instancia de la cuenta bancaria elegida con elegirCuenta

6. AlmacenadorMultiple
secreto: Oculta como se almacenan multiples saldos de cuentas bancarias
funcion: Permite utilizar un AlmacenadorDeCuenta para almacenar multiples saldos de cuentas
setAlmacenador (i AlmacenadorDeCuenta) toma un almacenador de cuentas y lo establece como el que se utilizara
almacenar (i List(CuentaBancaria)) toma una lista de cuentas bancarias y almacena el saldo de cada una asociado al numero de cuenta

7. CreadorMultiple
secreto: Oculta como se crean multiples cuentas apartir del almacenamiento secundario
funcion: Sirve para crear multiples cuentas utilizando un CreadorDeCuentas
setCreador (i CreadorDeCuentas) toma un creador de cuentas y lo establece como el que se va a utilizar
elegir (i List(NumeroCuenta)) toma una lista de numeros de cuenta y los selecciona para luego ser creados
crear(): List(CuentasBancarias)) devuelve una lista de instancias de cuentas bancarias creadas en base a los elegidos

8. NumeroCuenta
secreto: Oculta como se forma un numero de una cuenta bancaria
funcion: Sirve para representar el numero de una cuenta bancaria.
Hereda de IdentificadorYa que se puede utilizar cada uno.

9. Monto
secreto: Oculta como se representa un MONTO de dinero
función: Sirve para representar un monto de dinero. Hereda el valor x q. que se puede almacenar

4)

MODULE	Usuario
IMPORTS	Nombre, Contraseña
EXPORTS	set Nombre (i Nombre) set Contraseña (i Contraseña) nombre (): Nombre contraseña (): Contraseña

MODULE	Algoritmo Encriptacion
IMPORTS	Contraseña, Encriptado
EXPORTS	encriptar (i Contraseña) obtenerEncriptado (): Encriptado desencriptar (i Encriptado) obtenerContraseña (): Contraseña

MODULE	Algoritmo ACME	INHERITHS	FROM	AlgoritmoEncriptacion

MODULE	Algoritmo EMCA	INHERITHS	FROM	AlgoritmoEncriptacion

MODULE	Almacenador
IMPORTS	Identificador, Value, Bool
EXPORTS	almacenar (i Identificador, i Value) Primero () identificador (): Identificador valor (): Value hayMas (): Bool siguiente ()

Nielsen
Hoja 4
de 4

eliminar ()

MODULE Gestor De Usuarios
IMPORTS Almacenador, AlgoritmoEncriptacion, Nombre, Contraseña, Boolean
EXPORTS setAlmacenador (i Almacenador)
setEncriptadorDefault (i AlgoritmoEncriptacion)
crearUsuario (i Nombre, i Contraseña)
elegirAlgoritmo (i AlgoritmoEncriptacion)
validar (i Nombre, i Contraseña)
esValido (): Boolean

MODULE Encriptado INHERITS FROM Value

MODULE Nombre INHERITS FROM Identificador

GUIA DE MODULOS

1. Usuario
   secreto: oculta como se representa un usuario
   FUNCION: permite representar un usuario
   setNombre (i Nombre) toma un nombre y establece el nombre del usuario
   setContraseña (i Contraseña) toma una contraseña y la establece como la misma del usuario
   nombre (): Nombre devuelve el nombre del usuario
   contraseña (): Contraseña devuelve la contraseña del usuario

2. AlgoritmoEncriptacion
   secreto: Oculta como se implementan los algoritmos de encriptacion
   FUNCION: Sirve para encriptar una contraseña. Es un modelo abstracto.
   encriptar (i Contraseña) toma una contraseña la cual sera encriptada
   obtenerEncriptado (): Encriptado devuelve un encriptado de la contraseña pasada en encriptar
   desencriptar (i Encriptado) toma un encriptado el cual sera desencriptado
   obtenerContraseña (): Contraseña devuelve la contraseña desencriptada asociada al encriptado de desencriptar

3. AlgoritmoACME
   secreto: Oculta como se implementa el algoritmo de encriptacion ACME
   FUNCION: Hereda de AlgoritmoEncriptacion ya que aplica el algoritmo ACME

4. AlgoritmoEMAC
   secreto: Oculta como se implementa el algoritmo de encriptacion EMAC
   FUNCION: Hereda de AlgoritmoEncriptacion ya que aplica el algoritmo EMAC

5. Almacenador
secreto: Oculta como se implementa el almacenamiento de valores asociados a un identificador
Funcion: sirve para almacenar valores asociados a un identificador en un archivo de texto
almacenar (i identificador, i value) toma un identificador y un valor, almacenando el valor asociado al identificador en un archivo de texto.
primero() mueve el iterador al primer elemento
identificador(): Identificador devuelve el identificador del elemento actual
valor(): Value devuelve el valor del elemento actual
hayMas(): Bool devuelve un booleano indicando si hay mas elementos
siguiente() mueve el iterador al elemento siguiente
eliminar() elimina el elemento actual

6. GestorDeUsuarios
secreto: Oculta como se gestionan y crean usuarios.
Funcion: sirve para crear nuevos usuarios y almacenarlos en un almacenador. Tambien sabe como se encriptaran las contraseñas y como validarlas
setAlmacenador (i Almacenador) toma un almacenador y lo establece como el que se usara
setEncriptadorDefault (i AlgoritmoEncriptacion) toma un algoritmo de encriptacion y lo toma por defecto. Este sera usado si no se llama a elegirAlgoritmo antes de crearUsuario
crearUsuario (i Nombre, i Contraseña) toma un nombre y una contraseña y crea un usuario con esos datos. Si no se llamo al metodo elegirAlgoritmo usa el algoritmo por defecto para guardar la contraseña encriptada en el almacenador
elegirAlgoritmo (i AlgoritmoEncriptacion) toma un algoritmo de encriptacion y lo usara para encriptar la contraseña del proximo usuario que se cree
validar (i Nombre, i Contraseña) toma un nombre y contraseña y los validara con los almacenados
esValido(): Bool devuelve un booleano dependiendo si los datos pasados en validar son validos

7. Encriptado
secreto: Oculta como se representa una contraseña encriptada
cuncion: Sirve para representar una contraseña encriptada. Hereda de Value ya que se puede almacenar

8. Nombre
secreto: Oculta como se representa un nombre
Funcion: sirve para representar un nombre. Hereda de Identificador ya que se usa como uno

9. Contraseña
secreto: Oculta como se representa una contraseña
Funcion: sirve para representar una contraseña