PRÁCTICA 2

Tecnología de la Programación

Ingeniería Informática 2007/2008

Resumen

En esta práctica desarrollaremos un simulador de una pequeña entidad bancaria, que denominaremos MiniBank. Dicho simulador soportará las operaciones básicas: crear y destruir cuentas, ingresar y retirar dinero, y consultar el registro de operaciones.

Para acotar la complejidad del problema se han realizado varias simplificaciones respecto a lo que es una entidad bancaria real.

Objetivos

- Mejorar en el uso de UML
- Mejorar en el análisis y diseño Orientado a Objetos
- Mejorar en la implementación a partir de un diseño dado

Material

La práctica se debe realizar empleando UML como lenguaje de modelado y Java como lenguaje de programación.

Se entregará una pequeña interface de usuario para poder comprobar el funcionamiento global de la aplicación. Dicha interface será de tipo textual.

Requisitos

Todos los atributos definidos en el diseño deben tener ámbito privado y se deben respetar en todo momento los principios básicos de la Orientación a Objetos: encapsulación y Abstracción.

Descripción

Presentación del dominio

El sistema deseamos representar los distintos clientes de la entidad bancaria, MiniBank. Por cada cliente hemos encontrado relevante la siguiente información: nombre, primer apellido, dni y teléfono de contacto, así como la lista de cuentas de las que es titular. Por cada cuenta deseamos tener un número único que actue como identificador y su balance. También nos interesa conocer por cada cuenta quién es un titular, asumiendo que una cuenta tiene exactamente un titular.

Además de crear y eliminar cuentas, se permiten las operaciones típicas sobre ellas: ingresar o retirar dinero y realizar transferencias entre cuentas. La transferencia se limita a su comportamiento básico, es decir, se realiza una reintegro en la cuenta de origen y a continuación se realiza un ingreso en la cuenta destino, ambas por el mismo importe.

Como última característica del sistema, queremos realizar un registro de todas las operaciones que se realizan, anotando para cada una de ellas la fecha, el tipo de operación, y el importe. Además queremos saber por cada operación, sobre que cuenta se realizó, y por cada cuenta, qué operaciones se realizaron sobre ella.

En el registro de operaciones, debemos considerar el caso particular de la transferencia, pues se compone de dos operaciones básicas: ingresar y retirar. Deseamos alguna de las dos alternativas siguientes:

a) Guardamos un único registro por la operación de transferencia, pero no por las operaciones de reintegro e ingreso.

b) Guardamos un registro por cada una de las tres operaciones. En este caso, queremos establecer un vínculo entre ellas que me permita saber para una transferencia cuales son las operaciones asociadas, y viceversa.

Funcionalidades

- Crear cuenta
- Eliminar cuenta
- Ingresar dinero
- Retirar dinero
- Transferir dinero
- Consultar operaciones entre dos fechas

Requisitos de diseño e implementación

- No se permite modificar el código de las clases de la interface de usuario. Por este mismo motivo, tampoco se entrega el código fuente de las mismas. Así mismo, tampoco se permite modifcar la interface IMiniBank, puesto que el funcionamiento de la interface de usuario depende de ella.
- Todas las clases implementadas deben redifinir el método toString(). La información devuelta por dicho método será la información presentada en la interface de usuario.
- Todos los métodos definidos en la interface IMiniBank cuyo resultado es una colección de objetos, devuelven un Iterator. De esta manera se logra una mayor independencia entre las partes principales del sistema.
- La clase FactoriaMiniBank incluye una implementación a modo de ejemplo. Su única labor es crear y devolver una instancia de alguna clase que implemente la interface IMiniBank. También se incluye el esqueleto de una clase MiniBank que implementa IminiBank.
- Entre el código proporcionado ya se encuentra una clase Main con el método main() encargado de lanzar la aplicación. Dicha clase se encarga de crear una instancia del interprete de instrucciones para la interface de usuario y de una instancia que le sirva de fachada de la parte del modelo de la aplicación. Para esto último usa la clase FactoriaMiniBank.
- Las cantidades de dinero se representan como un entero (int), obviando los decimales. Aunque sabemos que en una aplicación real, deberíamos emplear algún tipo de punto fijo y nunca de punto flotante por la perdida de precisión que conlleva estos últimos.
- El ćodigo de la aplicación se divide en dos paquetes:
  - es.udc.fic.tp.minibank.iu La interfaz de usuario
  - es.udc.fic.tp.minibank.model El modelo de la aplicación
- En el diseño no es necesario detallar el contenido del paquete iu.

Entregables

- Breve memoria en formato PDF que incluya diseño de la aplicación modelado según el lenguaje UML y la Justificación de la decisión tomada respecto al registro de transferencias
- El código fuente de la aplicación.

La entrega de las prácticas se realizará a través del repositorio Subversion de cada grupo de prácticas, con la siguiente estructura:

/trunk/P2

src/ (código fuente)

doc/tp-gXXX-p2-memoria.pdf (donde XXX es el número de grupo)

Plazo de entrega

La duración estimada para la realización de la práctica es de dos semanas lectivas, por tanto la fecha de entrega es antes del 01/04/2008.