# Técnicas de Desarrollo de Programas
# Ingeniería Informática
# Curso 2008 / 2009

## Ejercicios de Patrones de Diseño:

*Patrones de Creación, Command*

## Ejercicio 1 (examen de septiembre año 2004/05)

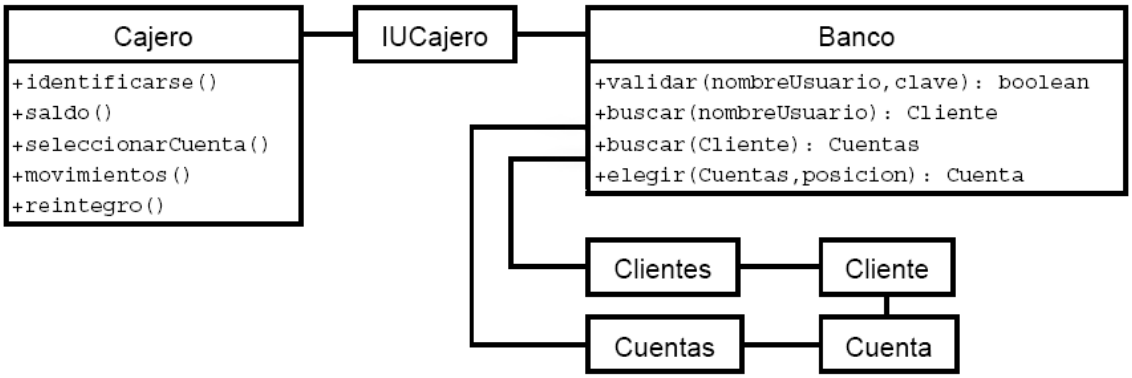
Desde el cajero de un banco un usuario puede realizar las siguientes peticiones:

- Identificarse
- Consultar el saldo total de sus cuentas
- Seleccionar una cuenta
- Consultar los últimos movimientos
- Sacar dinero

Las peticiones serán llevadas a cabo por el banco. Cuando éste realice las comprobaciones y operaciones pertinentes, devolverá el control al cajero para devolver lo necesario al usuario en función de la petición.

Es importante tener registradas las peticiones enviadas desde el cajero y en el orden en que se realizaron para poder reproducir la operativa en caso de fallo en cajero o para futuras auditorías.

La siguiente figura muestra las principales entidades que intervienen en el proceso, con las clases e interfaces relevantes para el problema:




Se pide:

1. El diagrama de clases resultado de aplicar el patrón de comportamiento que permita gestionar las peticiones enviadas de forma que, si en el futuro se desea incluir nuevas peticiones, se produzca el menor impacto posible en el resto de la aplicación.

2. El diagrama de secuencia que representa la situación en la que un usuario se identifica, consulta su saldo y retira dinero. Se supondrá que el sistema ya está inicializado con todas las instancias que toman parte en la acción. En el diagrama de secuencia se mostrará la interacción desde IUCajero hasta Banco, no se incluirá la interacción desde Banco hasta el resto de clases internas que forman parte del sistema como Cuentas, Clientes u otras.
3. Si se desea que un usuario pueda agrupar un conjunto de peticiones para que se ejecuten secuencialmente ¿qué patrón estructural se podría recomendar para tales peticiones? Explicar brevemente su funcionamiento.

## Ejercicio 2 (examen de junio año 2007/08)

Se quiere desarrollar una aplicación que permita jugar al dado (muestra un número aleatorio del 1 al 6) y a la moneda (muestra aleatoriamente cara o cruz). En ambos casos, la realización de una jugada debería ajustarse a la siguiente interfaz:

```
public interface Jugar {
   public void lanzar();
}
```

Se pide:

1. Especifica el diagrama de clases de la aplicación utilizando el patrón *Factory Method*. Identifica las clases participantes y qué roles desempeñan. ¿Qué ventajas proporciona el uso del patrón?
2. Especifica el diagrama de clases de la aplicación utilizando el patrón de diseño *Abstract Factory*. Identifica las clases participantes y qué roles desempeñan. ¿Qué ventajas proporciona el uso del patrón? ¿Qué similitudes y diferencias existen con la solución dada por el patrón *Factory Method*?
3. Especifica el diagrama de secuencia de la creación de un juego de la moneda y el lanzamiento de una moneda en el mismo, para los dos diseños propuestos en los apartados previos.