Taller Integrador: Patrones de diseño

# Sistema de Requerimientos Académicos.

# Objetivos

* Evaluar que patrones de diseño se deben utilizar en el desarrollo de un sistema.
* Aplicar varios patrones de diseño dentro de un mismo sistema.

# Requerimientos del sistema:

Se desea desarrollar un sistema de cajero automático que permita manejar consultas de saldo, retiros y depósitos de cuentas bancarias, para esto se le provee una clase llamada Account, la cual es la única que no debe ser modificada, pero también se proveen 3 clases más, que pueden ser modificadas a conveniencia, y sobre las que se debe aplicar los patrones de diseño elegidos.

# Desarrollar

1. Indique para cada uno de los patrones estudiados si pudiera o no servir dentro del desarrollo de este sistema. (explique)
   1. Creacionales:

<http://www.cristalab.com/tutoriales/patrones-de-diseno-creacionales-c99932l/>

* + 1. **Absrtract Factory:** No es necesario utilizarlo, debido a que no hay una familia de objetos.
    2. **Factory Method:** Es necesario para poder implementar los diferentes tipos de ATM que se requieren en este caso UK y US.
    3. **Singleton:** No es necesario, ya que si bien es un cajero único tampoco se necesita encapsular diferentes tipos de éste según su currency.
  1. Estructurales:

<https://highscalability.wordpress.com/2010/04/12/patrones%C2%A0estructurales/>

* + 1. **Composite:**  No se puede usar composite, ya que este patrón busca la independencia entre todos los componentes de la aplicación, entonces ninguna de las funcionalidades será tratada igual.
    2. **Adapter:** Es necesario para la conversión de la unidad monetaria para las transacciones de dinero sobre una cuenta
    3. **Decorator:** No es necesario agregar responsabilidades adicionales a los ATMs; ya que, el funcionamiento de este cajero, se da un menú estático.
  1. De Comportamiento.

<http://modelosdeprog.blogspot.com/2009/05/comportamiento.html?m=1>

* + 1. **Chain of Responsibility:** Es necesario para el mecanismo de entrega de dinero en los ATMs, encadenando de esta manera la denominación del billete hasta el handler que maneje dicha denominación de billete.
    2. **Strategy**: No es necesario utilizarlo puesto que las acciones que realiza el ATM no requieren de alguna estrategia y resulta más conveniente otro patrón de diseño de comportamiento.
    3. **Memento:** No es necesario para verificar que los estados de las cuentas antes de realizar una transacción sobre sí misma, ya que la cadena de responsabilidad ya toma en cuenta aquello.
    4. **Iterator:** No es aplicable, aunque se haga una búsqueda de las denominaciones a dispensar, éstas son pocas y no implican que sean de un tipo.

1. Diseñe un diagrama de clases del sistema, aplicando los patrones elegidos.