Taller Integrador: Patrones de diseño

# Sistema de transacciones de cuentas.

# 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.
      * Singleton. Para el desarrollo de este sistema si puede ser útil, ya que solo deberíamos crear un único cajero automático, así que usando este patrón se garantizará que se tenga una única instancia y un punto da acceso global a ella. También, debido a que se necesita una única instanciación al momento de realizar un depósito y/o retiro así como consulta de saldos de la cuenta bancaria
      * Factory Method. No se utiliza porque ya estamos definiendo los procesos a realizar: retiro de dinero, depósito de dinero y consulta de saldo en la cuenta.
      * Abstract Factory. No sirve porque no necesitamos o tener familias de diferentes clases para resolver el problema, en este caso para la moneda.
   2. Estructurales.
      * Adapter. No sirve, porque en este sistema no necesitamos conectar dos interfaces incompatibles por lo tanto un “wrapper” no es de mucha utilidad.
      * Composite. No sirve por que el usuario no podrá tratar objetos individuales y composiciones de objetos de forma uniforme, puesto que cada uno tiene su respectiva funcion.
      * Decorator. No sirve debido a que no queremos añadir más funcionalidades, solo se debe usar las que se necesitan que son: depositar, retirar y consultar el saldo.
   3. De Comportamiento.
      * Chain of Responsibility. Sí sirve, para el uso de los manejadores. Más específicamente, porque un depósito o retiro sólo involucra una denominación de moneda a la vez.
      * Memento: No sirve porque por motivos de seguridad, el comportamiento tipo last checkpoint no es recomendable para una transacción bancaria ya que no es tan segura.
      * Strategy. No sirve, porque Strategy se enfoca en proveer varias alternativas “para moverse de un punto A a un punto B” (resolver un requerimiento). Las transacciones especificadas sólo tienen un modo de efectuarse.
      * Iterator: No sirve, porque para el requerimiento en cuestión no se requiere recorrer una colección completa (en este caso, los manejadores. Es menester que el recorrido termina cuando encuentre el manejador apropiado).
2. Diseñe un diagrama de clases del sistema, aplicando los patrones elegidos.