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.

**Abstract Factory:** Este patrón maneja familias de objetos que pertenecen a una categoría en particular. Para este caso las únicas distinciones en objetos son los ATM que pueden ser de diferentes países con distintas monedas por lo que no requiere su uso.

**Factory method:** Este patron no es nesario ya que no se necesita alternar entre tipos de monedas ya que a un cajero le corresponde una moneda en particular y tiene los mismos métodos que otros cajeros con otras monedas.

**Singleton:** Necesaria para el registro de las transacciones de los clientes en una base de datos.

* 1. Estructurales.

**Composite:** No sirve porque no es necesario crear objetos complejos a mediante la composición recursiva de objetos similares. La cuenta no posee demasiados atributos y sus tipos de cuenta no son tan difíciles de crear.

**Adapter:** No se sirve, porque no es necesario adaptar una nueva clase con los métodos que usa el ATM.

**Decorator:** Si sirve porque me permite asignar funcionalidades a un objeto dinámicamente, de esta forma se evita la creación de subclases por herencia. Se lo usaría en caso de que se quiera agregar nuevas funcionalidades a cada una a de los tipo de cuenta que posee el banco, sin afectar la clase cuenta.

* 1. De Comportamiento.

**Chain of Responsibility:** Este patrón de comportamiento sería muy útil al momento de manejar las transacciones de dinero con el cliente debido a que podría manejar una cadena al momento de entregar o recibir efectivo por lo que calcularía el retiro o depósito dentro de una pipeline donde los nodos serían los tipos de billetes y monedas.

**Strategy:** No sirve porque las funcionalidades son iguales para todos los objetos, ya que solo se maneja dinero en efectivo.

**Memento:** No sirve porque no es necesario capturar el estado del objeto, y tampoco se necesita restaurarlo posteriormente.

**Iterator:** No sirve porque no es necesario acceder a los elementos de la cuenta agregada de forma secuencial, no se debe saber los detalles de la cuenta.

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