**Trabajo 1 – Patrones y diseño de software**

**Amilkar Fernando Martínez**

**Cristian Camilo Gutiérrez**

**Erich Alexis Báez**

**Instituto tecnológico metropolitano - ITM.**

**Especialización en ingeniería de software.**

**Patrones y Diseño de Software.**

**Carlos Alejandro Ruiz Ramirez.**

# Solución Primer Ejercicio

**Problema 1: Creación de transacciones bancarias – Patrón Factory Method**

**Problema:** Un cajero automático puede manejar distintos tipos de transacciones (retiro, depósito, consulta de saldo, pago de servicios). Se necesita una forma flexible de instanciar el tipo de transacción sin depender directamente de las clases concretas.

**Solución con Factory Method:**

Se define una fábrica (TransactionFactory) que genera transacciones según el tipo solicitado, evitando el acoplamiento con las clases concretas.

**Problema 2: Configuración de un cajero automático según el proveedor – Patrón Abstract Factory**

**Problema:** Existen distintos proveedores de hardware para cajeros (NCR, Diebold, Wincor). Cada uno tiene su propia implementación de dispensadores de efectivo, lectores de tarjetas y módulos de impresión. Se requiere una forma flexible de configurar un cajero con componentes compatibles según el proveedor.

**Solución con Abstract Factory:**

Se crea una fábrica abstracta (ATMFactory) con fábricas concretas (NCRFactory, DieboldFactory) que generan los componentes adecuados para cada proveedor

**Problema 3: Control de la sesión del usuario en el ATM – Patrón Singleton.**

**Problema:** Un usuario inicia sesión en el ATM, y se debe garantizar que su sesión sea única y accesible en toda la transacción hasta que cierre sesión.

**Solución con Singleton:**

Se implementa una clase ATMSession que almacena los datos de sesión y se asegura de que solo haya una instancia activa en el sistema.

Solución segundo Ejercicio  
  
**Diseño de la solución**

Para abordar este problema, utilizaremos dos patrones de diseño:

**Patrón Factory Method:**

Permite la creación de objetos de distintos tipos de archivos (XLSX, DOCX, XML) sin que el código cliente necesite conocer los detalles de implementación.

Proporciona una forma extensible de agregar nuevos formatos sin modificar el código existente.

**Patrón Builder:**

Facilita la construcción de documentos PDF de manera flexible, permitiendo la adición de elementos como tablas, columnas, títulos y estilos en tiempo de ejecución.

Permite generar documentos de diferentes estructuras sin afectar el código cliente.

Repositorio de Git:  
<https://github.com/EbaezrITM/entregable_1.git>

# **Diagrama de Clases**

Repo: <https://drive.google.com/file/d/1S6Y4l_KhYJWzplXbbqzcAUdQVmF_IFU6/view?usp=sharing>   
  




