Casos estudio

1: Nuevo tipo de documento (PowerPoint)

Para agregar PowerPoint al sistema de documentos:

Crear una nueva clase PowerPointProcessor que herede de DocumentProcessor.

Implementar en esa clase los métodos abstractos definidos (por ejemplo: validateDocument(), transformDocument(), etc.).

No se modifica la lógica del sistema principal, ya que este trabaja con el tipo base DocumentProcessor.

Ventaja: Se mantiene el polimorfismo y no se altera código existente.

2: Soporte para criptomonedas en sistema de pagos

Para agregar un nuevo método de pago (criptomonedas):

Crear una clase PagoCriptomoneda que implemente la misma interfaz o extienda la clase base usada por los otros métodos de pago (por ejemplo, ProcesadorPago).

Sobrescribir métodos como procesarPago() con la lógica específica para criptomonedas.

Integrar esta nueva clase en la lógica sin modificar las demás, gracias al uso polimórfico de ProcesadorPago.

Ventaja: Se pueden añadir nuevos métodos de pago sin tocar el código existente.

3: Nuevo método de autenticación biométrica

Para agregar autenticación biométrica:

Si la clase abstracta existente cubre métodos generales de autenticación (por ejemplo, autenticarUsuario()), se puede crear una nueva subclase AutenticacionBiometrica que implemente la lógica.

Si la autenticación biométrica tiene un enfoque totalmente distinto (por ejemplo, uso de hardware, sensores, etc.) que no encaja bien en la jerarquía actual, conviene crear una nueva jerarquía específica para este tipo.

Decisión: Si el comportamiento encaja en la abstracción actual → subclase nueva.

Si no encaja → nueva jerarquía.