**Parte 4: Preguntas de Reflexión (20 minutos)**

**Análisis de Diseño**

1. **¿Cuándo usar interfaces vs clases abstractas?**
   * Analiza los ejercicios realizados
   * Justifica tus decisiones de diseño
   * ¿Qué cambiarías en retrospectiva?
2. **¿Cómo mejora el polimorfismo la extensibilidad?**
   * Propón nuevas funcionalidades para los ejercicios
   * ¿Cómo las implementarías sin romper código existente?
3. **¿Qué patrones de diseño identificas?**
   * Strategy Pattern
   * Template Method Pattern
   * Factory Pattern
   * Observer Pattern

Solución

1. Interfaces vs Clases Abstractas

Se usaron clases abstractas cuando varias clases compartían lógica base, como en DocumentProcessor, que definía la estructura general del procesamiento y dejaba detalles a las subclases (PDF, Word, Excel).

Las interfaces son útiles cuando distintas clases deben compartir un mismo comportamiento sin herencia común, por ejemplo para unificar métodos como enviar() en diferentes tipos de notificaciones.

2. Polimorfismo y Extensibilidad

El polimorfismo permitió agregar nuevas funcionalidades sin modificar código existente.

3. Patrones de Diseño Identificados

Strategy: Cada clase concreta aplica su propia estrategia (procesamiento o envío).

Template Method: DocumentProcessor define un flujo fijo con pasos personalizables.

Factory: Se puede usar para crear el procesador adecuado según el tipo de archivo.

Observer: Podría aplicarse en notificaciones para avisar a varios receptores automáticamente.