Cátedra: Desarrollo de software

¡Trabajo Práctico Épico de Patrones Creacionales con Lombok! 🚀

¡Que la fuerza de los patrones te acompañe! 🚀

¡Aquí tienes el trabajo práctico mejorado con recomendaciones para cada patrón! 🚀

¡Trabajo Práctico Épico de Patrones Creacionales con Lombok! 🚀

Objetivo Principal:

- ¡Convertirte en un maestro Jedi de los patrones creacionales en Java! 🥷
- Entender a fondo las diferencias entre crear objetos de la forma tradicional vs. ¡clonarlos como un pro! 👯
- Descubrir cómo cada patrón te da superpoderes para que tu código sea flexible, ¡a prueba de balas y fácil de mantener! 💪

El Desafío:

¡Vamos a construir una mini biblioteca online con esteroides! 📚💻 Con funcionalidades que harán temblar a Amazon:

1. **Gestión de Libros y Usuarios:** ¡Control total sobre quién lee qué! 🤓
2. **Creación de Objetos Nivel Experto:** ¡Objetos complejos creados con precisión quirúrgica! 🔪
3. **Clonación al Estilo Star Wars:** ¡Duplica objetos cuando lo necesites! ✨
4. **Familias de Objetos Personalizadas:** ¡Diferentes interfaces según el tipo de usuario! 🎭

Misiones (Patrones en Acción):

Parte 1 - ¡Singleton al Rescate! 🦸

- **Objetivo:** ¡Asegurar que la base de datos de libros sea como el Santo Grial: única e irrepetible! 🏅
- **Tareas:**
    1. ¡Crea la clase Database como un Singleton de manual! 🤓
    2. Implementa métodos para agregar y listar libros. ¡Como un bibliotecario digital! 📚
    3. ¡Demuestra que no importa cuántas veces lo llames, siempre obtendrás la misma instancia! 🎯
- **Recomendaciones para el Singleton:**
    - **Hilo Seguro:** ¡Asegúrate de que tu Singleton sea seguro para usar en entornos multi-hilo! 🧵

- **Lazy Initialization:** ¡Crea la instancia solo cuando sea necesario para ahorrar recursos! 🤔
- **Enum Singleton:** ¡Considera usar un enum para un Singleton a prueba de balas y conciso! 😎

Parte 2 - ¡Factory Method, el Mago de los Libros! 🧙

- **Objetivo:** ¡Crear diferentes tipos de libros sin enloquecer al cliente! 🤯
- **Tareas:**
  1. Crea la interfaz Libro y las clases concretas LibroFisico y LibroDigital. ¡Elige tu aventura! 🖼️
  2. ¡Implementa el Factory Method en LogisticaLibro para que devuelva la instancia correcta! ❓
  3. ¡Muestra en consola la creación de varios libros y sus tipos! 🔬
- **Recomendaciones para el Factory Method:**
  - **Jerarquía de Clases:** ¡Usa este patrón cuando tengas una jerarquía de clases con múltiples variantes! 🌳
  - **Configuración Dinámica:** ¡Permite configurar qué tipo de objeto crear en tiempo de ejecución! ⚙️
  - **Abstracción:** ¡El Factory Method es genial para abstraer el proceso de creación de objetos! 🖼️

Parte 3 - ¡Abstract Factory, el Arquitecto de la Experiencia! 🏗️

- **Objetivo:** ¡Crear familias de objetos relacionados según el tipo de usuario! 👨‍👩‍👧‍👦
- **Tareas:**
  1. ¡Define las interfaces InterfazUI y MetodoEnvio! 🎨
  2. ¡Implementa AdminUI/UsuarioUI y EnvioNormal/EnvioExpress! 🚚
  3. ¡Crea AbstractFactory y las fábricas concretas (AdminFactory, UsuarioFactory)! 🏭
  4. ¡Usa la fábrica para crear objetos correctos según el tipo de usuario y muéstralo en consola! 📣
- **Recomendaciones para el Abstract Factory:**
  - **Consistencia:** ¡Asegúrate de que todos los objetos creados por una fábrica concreta sean compatibles! 🤝
  - **Escalabilidad:** ¡Ideal para agregar nuevas familias de productos fácilmente! 📈
  - **Evita la Complejidad:** ¡No lo uses si solo tienes una única familia de productos! 🚫

Parte 4 - ¡Builder, el Constructor de Sueños! 👷

- **Objetivo:** ¡Construir objetos complejos de forma clara y flexible! 🧘
- **Tareas:**
    1. ¡Crea la clase Usuario con atributos opcionales! 🧰
    2. ¡Implementa Usuario.Builder para construir instancias como un profesional! 🏗️
    3. ¡Crea al menos 2 usuarios con distintos atributos y muestra la información! 🅸
- **Recomendaciones para el Builder:**
    - **Inmutabilidad:** ¡Considera hacer que tus objetos sean inmutables para mayor seguridad! 🔒
    - **Validación:** ¡Implementa la validación de datos en el Builder para evitar errores! ✅
    - **Legibilidad:** ¡El Builder mejora la legibilidad del código al construir objetos complejos! 🥸

Parte 5 - ¡Prototype, el Clonador Supremo! 👽

- **Objetivo:** ¡Clonar objetos existentes para crear nuevos objetos similares! 👯
- **Tareas:**
    1. ¡Crea la clase Prestamo con atributos: libro, usuario, fechalnicio, fechaFin! 🕵️
    2. ¡Implementa el método clone() (shallow o deep)! 🤿
    3. ¡Crea un préstamo prototipo y clónalo al menos dos veces, modificando algunos atributos! 🚀
    4. ¡Muestra en consola que los clones son independientes del original! 🤝
- **Recomendaciones para el Prototype:**
    - **Rendimiento:** ¡Clonar puede ser más rápido que crear un objeto desde cero! 🏎️
    - **Deep vs. Shallow:** ¡Elige cuidadosamente entre clonación profunda y superficial! 🤔
    - **Cuidado con las Referencias:** ¡Asegúrate de que las referencias a otros objetos se manejen correctamente! ⚠️

Entrega Final:

1. ¡Un proyecto completo en IntelliJ IDEA! 🏆
2. ¡Archivos .java organizados por paquetes como un verdadero ninja! 🥷
3. **¡Usa Lombok para reducir el código repetitivo y hacer que tu código sea más limpio y legible!** ✨

¡Que la fuerza de los patrones te acompañe! 🚀