Marca V (Verdadero) o F (Falso) según corresponda:

Patrones Creacionales:

\_V\_\_ El patrón Singleton garantiza que solo exista una instancia de una clase en toda la aplicación.

\_F\_\_ En el patrón Singleton, el constructor debe ser público para poder crear la instancia.

\_V\_\_ El patrón Prototype se utiliza para clonar objetos existentes en lugar de crear nuevos desde cero.

\_V\_\_ El patrón Factory Method define una interfaz para crear objetos, pero deja que las subclases decidan qué clase instanciar.

\_V\_\_ El patrón Abstract Factory sirve para crear familias de objetos relacionados sin especificar sus clases concretas.

\_V\_\_ El patrón Builder es útil cuando necesitamos construir objetos complejos paso a paso.

\_F\_\_ En el patrón Factory, siempre es necesario crear una clase abstracta.

\_V\_\_ El patrón Singleton puede tener problemas en aplicaciones multihilo si no se implementa correctamente.

Patrones Estructurales:

\_V\_\_ El patrón Adapter permite que dos interfaces incompatibles trabajen juntas.

\_F\_\_ El patrón Facade hace que un sistema complejo sea más difícil de usar.

\_V\_\_ El Adapter actúa como un "traductor" entre dos clases incompatibles.

\_V\_\_ El patrón Facade proporciona una interfaz simplificada a un conjunto de interfaces en un subsistema