Objetivo:

Desarrollar una aplicación de consola que permita crear y confirmar pedidos aplicando estrategias de envío, construyendo el pedido con Builder y notificando con Observer, incluyendo persistencia en JSON y uso de Inyección de Dependencias (DI) y Repository para desacoplar la lógica.
Todo los contenidos en este examen es el resultado de todas las guías y trabajos realizados disponibles en la plataforma ETNA.

Consignas:

1. Creá un proyecto de consola en C#. Implementá una arquitectura organizada. (Modelo–Controlador–Vista o Facade).
2. Utilizá inyección de dependencias: las clases deben recibir por constructor los objetos que utilizan (repositorio, estrategias, servicios, etc.).
3. Toda la lógica de negocio debe estar fuera del Program.cs. El programa principal solo se encarga de mostrar el menú y delegar acciones al controlador o fachada.

Estructura del proyecto

Debés implementar las siguientes clases e interfaces mínimas:

- Producto: con propiedades Nombre, Precio y Cantidad.
- Pedido: contiene una lista de productos, datos del cliente, dirección y totales.
- IPedidoBuilder y PedidoBuilder: implementan el patrón Builder para construir pedidos paso a paso.
- IEnvioStrategy y tres estrategias concretas: Moto, Correo y Retiro.
- PedidoService: dispara un evento (Observer) al confirmar un pedido. Los observadores (Cliente, Logística) deben reaccionar mostrando mensajes en consola.
- IRepositorio<Pedido> y RepositorioJson: permite guardar los pedidos confirmados en un archivo pedidos.json usando System.Text.Json.
- PedidoController o CheckoutFacade: coordina toda la lógica, recibe por constructor el repositorio, el servicio y la estrategia de envío.

Funcionalidad obligatoria

- ❖ Agregar productos al pedido (nombre, precio, cantidad). Validar entradas numéricas.
- ❖ Seleccionar tipo de envío (Strategy) y recalcular total.
- ❖ Confirmar pedido: validar datos mínimos (cliente, dirección, al menos un ítem), disparar notificaciones (Observer) y guardar el pedido en JSON.
- ❖ Implementar manejo de errores básico con try/catch y mensajes legibles al usuario.

Criterios de evaluación

Eje	Descripción	Ponderación
Patrones de diseño	Uso correcto de Strategy, Builder y Observer.	40%
Arquitectura	Controlador/Fachada con DI; separación de lógica y presentación.	30%
Persistencia y errores	Guardado en JSON, manejo de errores y validaciones básicas.	20%
Claridad y buenas prácticas	Código ordenado, nombres claros, comentarios precisos.	10%

Entrega

- Código compilable.
- Archivo pedidos.json generado al menos una vez.
- Archivo .zip con Apellido y nombre. Ej: FulanitoDeTal.zip