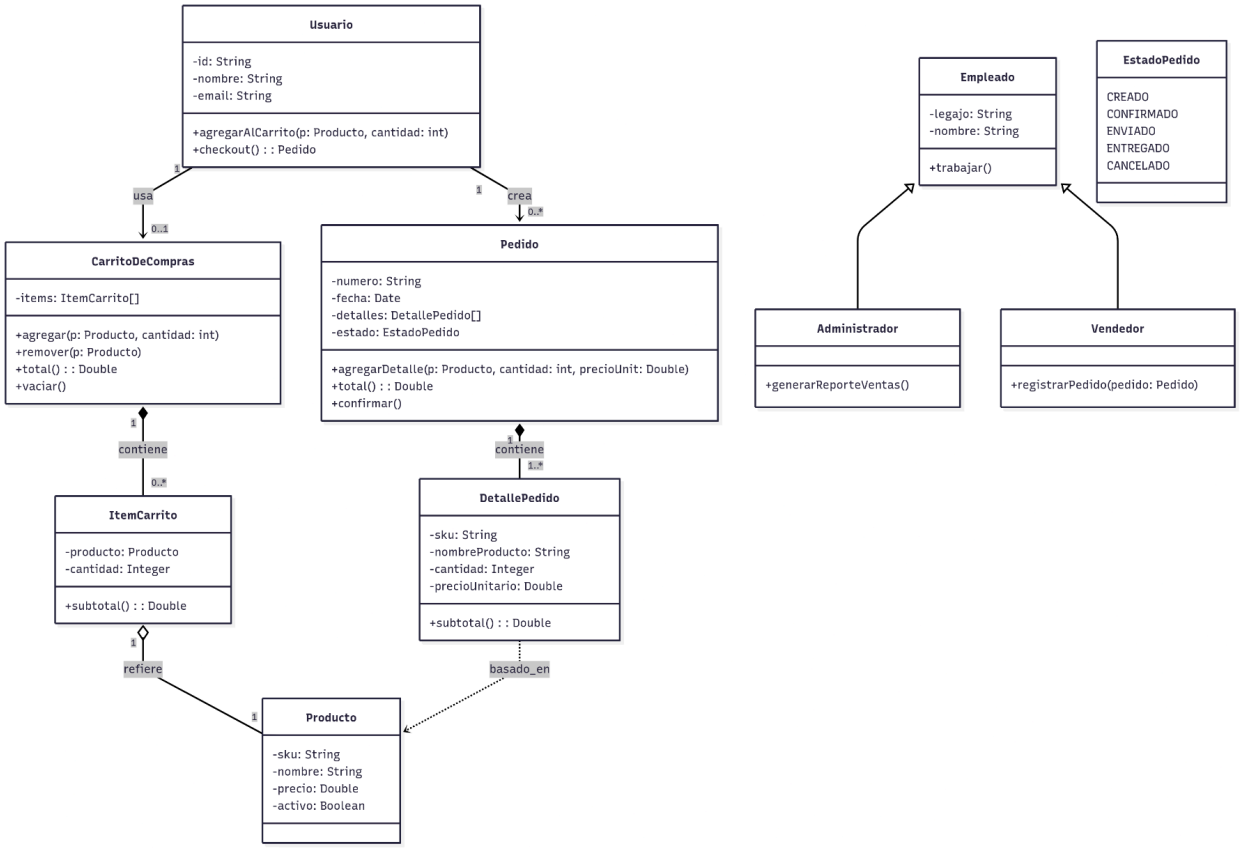
Trabajo Práctico — Implementación del modelo (Java)

Objetivo: Implementar en Java el modelo del diagrama provisto, usando solo modelo, enumerados y clases de gestión con métodos estáticos que almacenen datos en List.



Requisitos

- Java 17, Maven, IntelliJ IDEA.
- Package base: ar.edu.unju.fi.ecommerce
- GitLab: repositorio con rama main y README.md obligatorio. Branches opcionales.

1) Modelo a implementar

Usuario / Carrito / Producto / ItemCarrito

- Usuario { id, nombre, email}
  - agregarAlCarrito(p: Producto, cantidad:int)
  - checkout(): Pedido

- CarritoDeCompras { items: List<ItemCarrito> }
  - agregar(p: Producto, cantidad:int)
  - remover(p: Producto)
  - total(): BigDecimal
  - vaciar()

- ItemCarrito { producto: Producto, cantidad: Integer }
  - subtotal(): BigDecimal

- Producto { sku:String, nombre:String, precio:BigDecimal, activo:Boolean }

Pedido / Detalle / Estado

- Pedido { numero:String, fecha:Date, detalles:List<DetallePedido>, estado:EstadoPedido }
  - agregarDetalle(p: Producto, cantidad:int, precioUnit: BigDecimal)
  - total(): BigDecimal
  - confirmar() (solo transición CREADO → CONFIRMADO)

- DetallePedido { sku:String, nombreProducto:String, cantidad:Integer, precioUnitario:BigDecimal }
  - subtotal(): BigDecimal

- EstadoPedido { CREADO, CONFIRMADO, ENVIADO, ENTREGADO, CANCELADO }

- Nota: En checkout() se copia al detalle el sku, nombre y precioUnit del Producto al momento de la compra.

Herencia

- Empleado { legajo, nombre, trabajar() }

- Administrador extends Empleado { generarReporteVentas() }

- Vendedor extends Empleado { registrarPedido(pedido: Pedido) }

2) Clases de gestión (estáticas, almacenamiento en List)

Crear un paquete gestor con listas estáticas internas. Sin Map ni otras estructuras.

GestorUsuarios

- crear(Usuario u)
- buscarPorId(String id): Usuario
- buscarPorNombre(String nombre): List<Usuario>
- modificarEmail(String id, String nuevoEmail)
- listar(): List<Usuario>

GestorProductos

- crear(Producto p)
- buscarPorNombre(String nombre): List<Producto>
- modificarPrecio(String sku, BigDecimal nuevoPrecio)
- activar(String sku) / desactivar(String sku)
- listar(): List<Producto>

GestorPedidos

- crearDesdeCarrito(Usuario u): Pedido (puede delegar en u.checkout())
- buscarPorNumero(String numero): Pedido
- confirmar(String numero)
- listarPorUsuario(String idUsuario): List<Pedido>
- listar(): List<Pedido>

GestorEmpleados

- crearAdministrador(String legajo, String nombre): Administrador
- crearVendedor(String legajo, String nombre): Vendedor
- buscarPorLegajo(String legajo): Empleado
- listar(): List<Empleado>

3) Ejecución por consola (clases main)

Crear una o más clases main (paquete ar.edu.unju.fi.ecommerce.app):

MainDemoEcommerce

- Crear productos y un usuario (gestores).
- Agregar productos al carrito, mostrar items y total del carrito.
- Generar pedido (GestorPedidos.crearDesdeCarrito o checkout) e imprimir detalles y total.
- Mostrar estado antes y después de confirmar.
- Listar pedidos del usuario.

MainEmpleados

- Crear Administrador y Vendedor (gestor).
- Invocar trabajar() y generarReporteVentas().
- Crear un Pedido de ejemplo e invocar vendedor.registrarPedido(pedido).

4) GitLab

- Crear repo (ej.: tp-ecommerce-relaciones) con rama main.
- Commits claros y frecuentes.
- README.md con requisitos y cómo compilar/ejecutar (indicando las clases main).
- Incluir .gitignore de Java/Maven.

5) Criterios de corrección (compacto)

- Puede agregar otros atributos que sean relevantes al modelo y relaciones.
- Funcionamiento de los gestores con List y métodos estáticos.
- Demostración por consola del flujo (carrito ⟶ pedido ⟶ confirmar) y de herencia (admin/vendedor).
- Organización del proyecto, nombres en español y uso de BigDecimal.
- Repo en GitLab con README.md.