Lenguaje natural:

La intención es agregar un nuevo producto a la lista de productos del almacén. Primero se verifica que el código del producto no esté ya siendo utilizado por otro producto ya existente. Si el código es único y todos los datos del producto son válidos, entonces se procede a incorporarlo a la lista del almacén.

Precondiciones:

El producto no puede ser null.

El código de identificación puede ser repetido, pero no va a ser agregado a la lista.

El control de los atributos de los productos (evitando que el código y la descripción sean nulos y que el stock y la cantidad sea negativos) son controlados en el constructor de la clase producto.

Postcondiciones:

Si ya existía un producto con el mismo código la lista no se modifica. Retorna false

Si el producto es null la lista no se modifica. Retorna False

Si el código de producto no está en uso y lo anterior no se cumple, el producto es incorporado a la lista. Retorna True

Pseudocodigo:

```
insertarProducto (objeto IProducto)
Inicio
  Si producto es nulo entonces
    retornar falso
  FinSi
  Si buscarPorCodigo(producto.getCodProducto()) ≠ nulo entonces
    retornar falso
  FinSi
  listaProductos.insertar(producto, producto.getCodProducto())
  cantidadProductos ← cantidadProductos + 1
  montoTotal ← montoTotal + (producto.getPrecioUnitario() × producto.getStock())
  retornar verdadero
Fin
```

Lenguaje natural

Se debe permitir aumentar el stock de un producto dentro de la lista. Primero se debe recibir el código del producto y la cantidad a aumentar. Debemos que saber que tipos de datos son válidos (el stock no puede ser negativo, por ejemplo). Hay que buscar el producto dentro de la lista según su código. Si el producto existe se le agrega el stock correspondiente sino no se realizar nada.

Precondiciones:

El código de producto no debe ser nulo.

La cantidad a agregar debe ser un número entero positivo, no vamos a tomar a 0 como válido.

Debe existir un producto con el código de la lista para poder sumarle el stock

Postcondiciones:

Si todas las precondiciones se cumplen se puede agregar el stock y el producto aumenta su cantidad de stock.

Se actualiza el monto total del inventario si cumplimos todas las precondiciones

Si llega a no cumplirse alguna de las postcondiciones debemos retornar false

Pseudocodigo:

```
agregarStock(codigo, cantidad)
  Si codigo es nulo o cantidad es nulo o cantidad <= 0
    retornar false
  FinSi
  producto ← buscarPorCodigo(codigo)
  Si producto es nulo //Si no se encuentra el producto en la lista
    retornar false
  FinSi
  agregado ← producto.agregarCantidadStock(cantidad)

  Si agregado es true
    montoTotal ← montoTotal + (producto.getPrecioUnitario() * cantidad)
```

FinSi

retornar agregado

FinFuncion

Lenguaje natural:

Queremos simular que vendemos un producto y según la cantidad reduciremos su stock. Debemos saber que producto se va a vender (el código) y que cantidad (cuanto stock vamos a reducir. Primero, hay que verificar que la cantidad no se nula ni sea cero o menor y además hay que verificar que el código no sea nulo. Posteriormente verificamos que el producto esté en la lista. Si existe dentro de nuestra lista y tiene suficiente stock, se le descuenta la cantidad dada. Si alguna de estas condiciones no se cumple se retorna falso, mientras que si se pasan todas las condiciones se indicará con un true

Precondiciones:

El código del producto no debe ser nulo.

La cantidad para reducir debe ser un número entero positivo y mayor a 0.

Debe haber un producto con el código dado.

El stock actual debe ser mayor o igual a la cantidad a reducir.

Postcondiciones:

Si todas las condiciones se cumplen, se descuenta la cantidad del stock, se actualiza el monto total del inventario y se retorna true.

Si alguna condición falla, no se modifica el stock ni el monto total. Se retorna false.

Pseudocódigo:

reducirStock(codigo, cantidad)

Inicio

Si codigo es nulo o cantidad es nulo o cantidad <= 0

retornar false

FinSi

```
  producto ← buscarPorCodigo(codigo)
  Si producto es nulo
    retornar false
  FinSi

  Si producto.getStock() < cantidad
    retornar false
  FinSi

  producto.reducirStock(cantidad)
  montoTotal ← montoTotal - (producto.getPrecioUnitario() * cantidad)
  retornar true
Fin
```

Lenguaje natural:

Queremos listar todos los productos que tiene el almacen, ordenados alfabéticamente por su nombre (descripción) y dar su stock. Para obtener esta lista primero debemos obtener la lista base de los productos (sin ordenar), creamos una nueva lista ordenada utilizando la descripción de cada producto como comparador para así ordenar. Luego, recorremos la lista ordenada y damos su descripción junto con su stock.

Precondiciones:

- La lista de productos no debe ser nula.
- Debe haber al menos un producto registrado para que el resultado no sea una cadena vacía.

Postcondiciones:

- Se retorna una cadena de texto con los productos ordenados por nombre y su stock.
- Si no hay productos registrados, puede devolverse una cadena vacía o null.

Pseudocódigo:

```
listarOrdenadoPorNombre()
Inicio
  Si listaProductos es nula o listaProductos.esVacia es true
    retornar “No hay productos”
  FinSi
  productos ← listaProductos.toArray()
  listaOrdenada ← nueva ListaOrd
  Por cada producto en productos
    listaOrdenada.insertar(producto, producto.getDescripcion())
  FinPara
  resultado ← “” //String vacío, en java se utilizará un StringBuilder
  productosOrdenados ← listaOrdenada.toArray() //Array para recorrer
  Por cada producto en productosOrdenados
    resultado ← resultado + producto.getDescripcion() + " - Stock: " +
producto.getStock() + "\n"
  FinPara
retornar resultado
Fin
```