

Semana 4

Desarrollo Orientado a Objetos II (PRY2203)

Formato de respuesta

| **Nombre estudiante:** | Carlos Barria Valdvenito |
| --- | --- |
| **Asignatura: DOO2** | **Carrera: Desarrollo de Aplicaciones** |
| **Profesor: Alberto Campos Vidal** | **Fecha: 02/09/2024** |

# Descripción de la actividad

En esta cuarta semana realizarás una actividad formativa en pareja por encargo llamada "Creando un ambiente de ciclo de vida de software", donde deberás estructurar un ambiente de ciclo de vida de software al caso planteado, considerando la integración de sistemas y el aseguramiento de la calidad, de manera de proporcionar un marco estructurado y sistemático a través de las diferentes fases de creación del producto de software.

## Instrucciones específicas

Para realizar la actividad formativa de esta semana, presta atención al caso planteado:

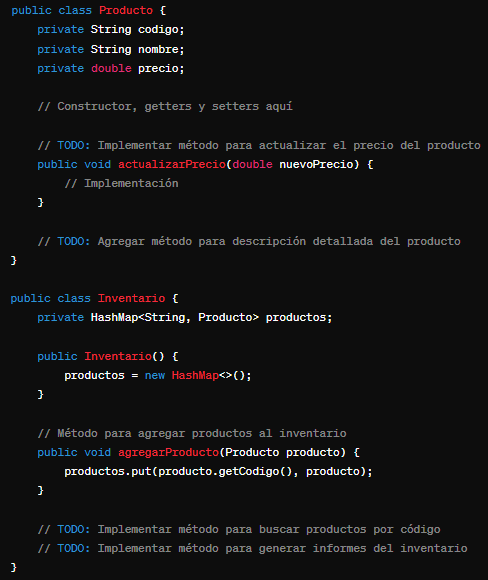
**Desarrollo de una aplicación de gestión de inventario en Java**

El desafío se centra en desarrollar una aplicación de gestión de inventario utilizando el lenguaje de programación Java. La aplicación debe permitir a los usuarios realizar operaciones como agregar nuevos productos, actualizar información de productos existentes, realizar búsquedas en el inventario y generar informes (se recomienda utilizar colecciones).

Bajo este contexto, y previo a los pasos correspondientes al ciclo de vida del software, te brindaremos un código de partida del proyecto “GestionInventarioJava”, el que deberás gestionar en Apache NetBeans:

**Figura 1**

*Código de partida en Java*

Este código es una estructura básica del proyecto mencionado con clases vacías y parcialmente completadas para modelos de datos (‘Producto’, ‘Inventario’). Deberás guiarte por los comentarios que sugieren métodos a implementar y mejorar.

*Nota.* Proyecto *“GestionInventarioJava”*. The Apache Software Foundation. (2023). *Apache NetBeans (16).* [Software]. Apache NetBeans.<https://netbeans.apache.org/front/main/>

**Importante**

El enfoque principal del caso de estudio será la integración de sistemas y el aseguramiento de la calidad a lo largo del ciclo de vida del software.

Para la realización de la actividad, ahora deberás desarrollar los siguientes pasos:

**Paso 1: Fase de análisis**

Deberás identificar los requisitos funcionales y no funcionales de la aplicación, así como también definir las interfaces y las interacciones con otros sistemas. Haz el registro en la siguiente tabla:

|  |  |
| --- | --- |
| **Requisitos funcionales** | **Requisitos no funcionales** |
| Agregar Productos:  El sistema debe permitir al usuario incorporar productos al inventario. | Usabilidad:  El sistema es fácil de usar e intuitivo, gracias a su menú principal, en el cual se presentan las opciones disponibles. |
| Actualizar Precio Producto:  La actualización del precio de los productos existentes; debe estar disponible a través del Sistema. | Rendimiento:  El sistema debe manejar un tiempo de respuesta rápido (razonable), al momento de entregar resultados. |
| Buscar Productos por código:  El Sistema permitir la búsqueda de productos por código. | Seguridad:  El sistema debe proteger la integridad de los datos. Se debe implementar medidas para prevenir accesos no autorizados y/o ataques maliciosos (ciberataques). |
| Generar Informes:  El sistema permitir la generación de informes de los productos existentes. | Disponibilidad:  El sistema debe estar disponible 7/24hrs. Consiente del tiempo de inactividad (programado) por Mantención del sistema. |
|  |  |

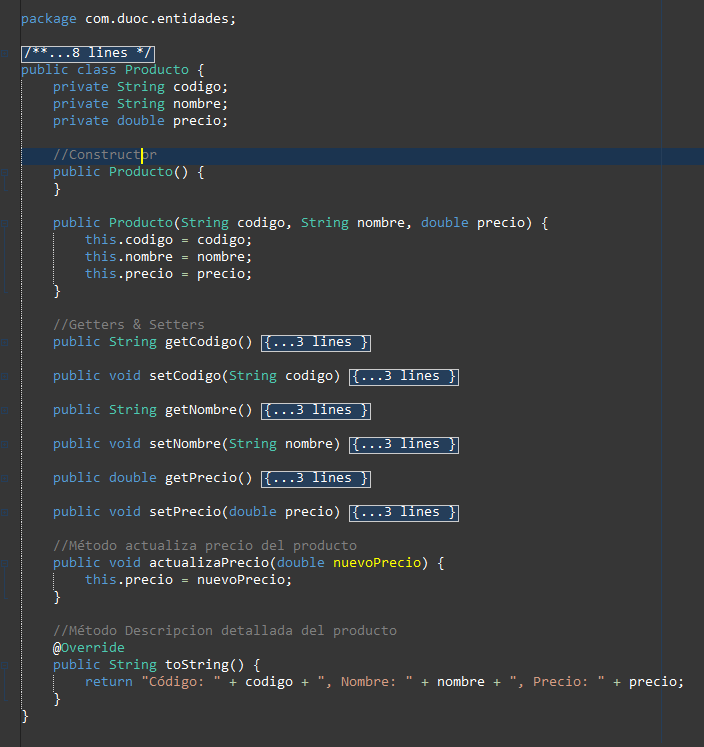
**(Elimina o añade filas según sea necesario)**

**Paso 2: Fase de diseño**

Tendrás que definir las clases, atributos y métodos claves para implementar las funcionalidades esenciales del sistema. Adjunta la imagen de los componentes de código correspondiente a:

**2.1 Clase Producto**

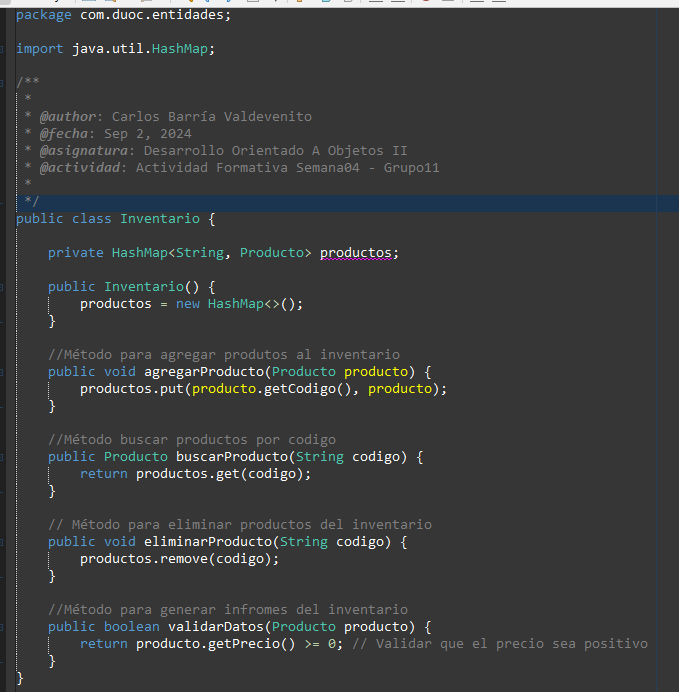
La clase ‘Producto’ representa los items individuales que serán almacenados en el inventario. Esta clase incluye atributos básicos como el ID, nombre, descripción, precio y cantidad en stock.



**(Ajusta el cuadro de respuesta según sea necesario)**

**2.2 Clase Inventario**

La clase ‘Inventario’ gestiona una colección de productos. Incluye métodos para agregar, eliminar, y buscar productos por nombre, así como listar todos los productos disponibles.



**Paso 3: Implementación**

Deberás desarrollar la aplicación en Java, enfocando la codificación en implementar las funcionalidades clave del sistema, como la gestión de productos e inventario.

**3.1 Interacción con el usuario:**

* Se implementará una interfaz de usuario simple basada en la consola para interactuar con el sistema. Esta interfaz permitirá al usuario ejecutar acciones como agregar, eliminar, buscar productos y listar el inventario.
* Se creará una clase ‘MenuPrincipal’ que mostrará las opciones disponibles y gestionará la entrada del usuario.

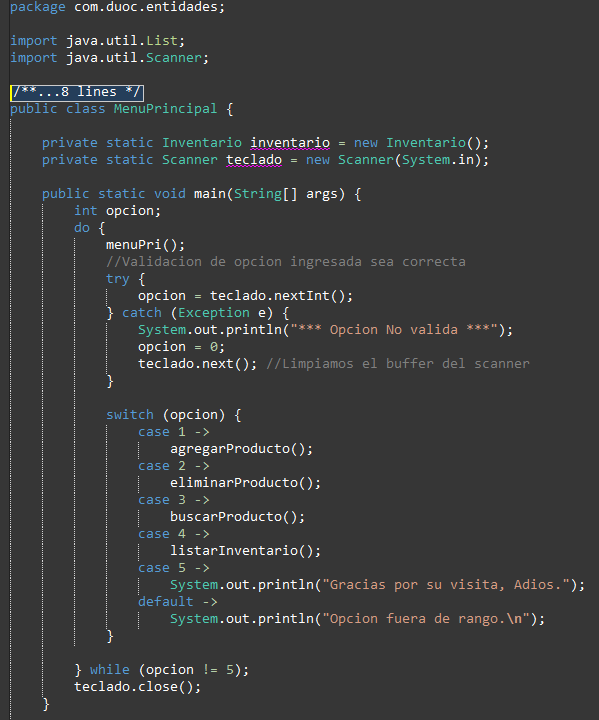
**3.2 Gestión de productos:**

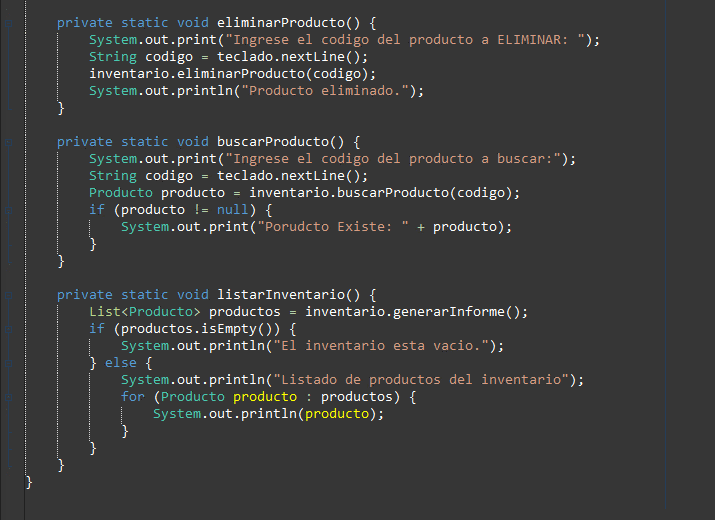
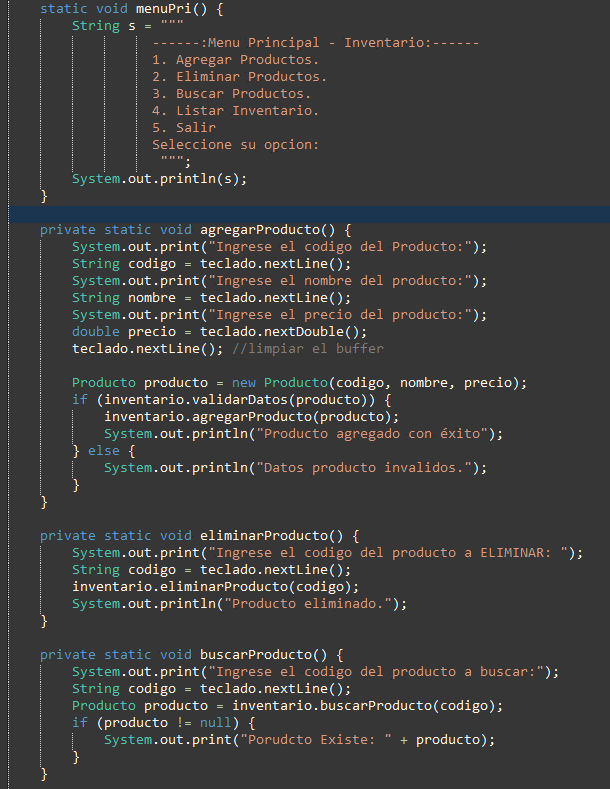
Basándonos en la clase ‘Producto’ ya definida, implementaremos métodos en la clase ‘Inventario’ para agregar nuevos productos al inventario, actualizar la información de los productos existentes y eliminar productos del inventario.

**3.3 Búsqueda y listado de productos:**

* Implementaremos una función de búsqueda en la clase ‘Inventario’ que permita encontrar productos por nombre o descripción.
* Además, se añadirá un método para listar todos los productos disponibles en el inventario.

Clase ‘MenuPrincipal’



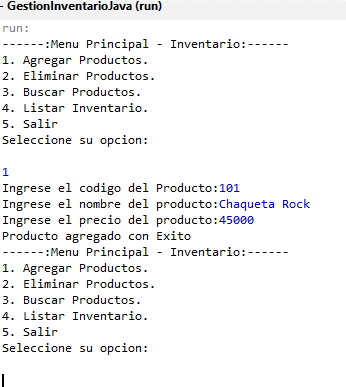


**Paso 4: Fase de pruebas**

**4.1 Pruebas de integración:** tendrás que realizar pruebas de integración para asegurarte que la aplicación se integra correctamente con otros sistemas y cumple con los requisitos establecidos. Las pruebas de integración deberían incluir:

Integración entre ‘Producto’ e ‘Inventario’: verifica que los productos se agregan, buscan y eliminan correctamente del inventario, comprobando la coherencia del estado del inventario después de cada operación.

Documenta en esta parte los resultados observados, incluyendo capturas de pantalla, registros o cualquier otro dato que evidencie el comportamiento del sistema bajo prueba.



**4.2 Pruebas unitarias:** tendrás que realizar pruebas unitarias para garantizar que cada componente del sistema funcione correctamente de forma aislada antes de integrarlo con otros componentes. Las pruebas unitarias a realizar serán:

**Para la clase ‘Producto’:**

1. Creación de producto: verificar que se pueda crear un objeto ‘Producto’ con todos los atributos esperados.

* Comprobar que los valores asignados a los atributos del producto, como nombre, descripción, precio y cantidad en stock, son correctos tras la inicialización.

1. Actualización de atributos: probar los métodos setters para asegurar que los atributos del producto se actualizan correctamente.

* Cambiar y recuperar el precio de un producto.
* Cambiar y verificar la cantidad en stock de un producto.

**Para la clase ‘Inventario’:**

1. Agregar producto al inventario: asegurar que un producto se agrega correctamente al inventario.

* Agregar un producto y verificar que el inventario lo contiene.
* Probar agregar un producto nulo y manejar adecuadamente la excepción o el caso de error.

1. Eliminar producto del inventario: confirmar que la eliminación de productos funcione correctamente.

* Eliminar un producto por su ID y comprobar que ya no esté en el inventario.
* Intentar eliminar un producto con un ID inexistente y verificar el manejo del caso.

1. Buscar producto por nombre: verificar que la búsqueda por nombre retorna los productos correctos.

* Buscar por un nombre específico y comprobar que se devuelven todos los productos que coinciden.
* Realizar una búsqueda con un nombre que no existe en el inventario y verificar que se devuelve una lista vacía o un valor nulo.

1. Listar todos los productos: asegurar que el inventario pueda listar todos los productos correctamente.

* Verificar que la lista devuelta contiene todos los productos agregados previamente.

Adjunta la evidencia de tus pruebas unitarias aquí:

**Cuando termines tu entrega, debes subir este mismo archivo al AVA para ser retroalimentado/a.**

## Anexo

Aquí tienes disponible el código de la Figura 1:

public class Producto {

private String codigo;

private String nombre;

private double precio;

// Constructor, getters y setters aquí

// TODO: Implementar método para actualizar el precio del producto

public void actualizarPrecio(double nuevoPrecio) {

// Implementación

}

// TODO: Agregar método para descripción detallada del producto

}

public class Inventario {

private HashMap<String, Producto> productos;

public Inventario() {

productos = new HashMap<>();

}

// Método para agregar productos al inventario

public void agregarProducto(Producto producto) {

productos.put(producto.getCodigo(), producto);

}

// TODO: Implementar método para buscar productos por código

// TODO: Implementar método para generar informes del inventario

}



Reservados todos los derechos Fundación Instituto Profesional Duoc UC. No se permite copiar, reproducir, reeditar, descargar, publicar, emitir, difundir, de forma total o parcial la presente obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de Fundación Instituto Profesional Duoc UC La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.