Texto

Descripción generada automáticamente

**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**FACULTAD DE ESTUDIOS SUPERIORES ARAGON**

**PROYECTO FINAL**

**P R E S E N T A**

Alexis Hernández Zamudio

**APROFESOR**

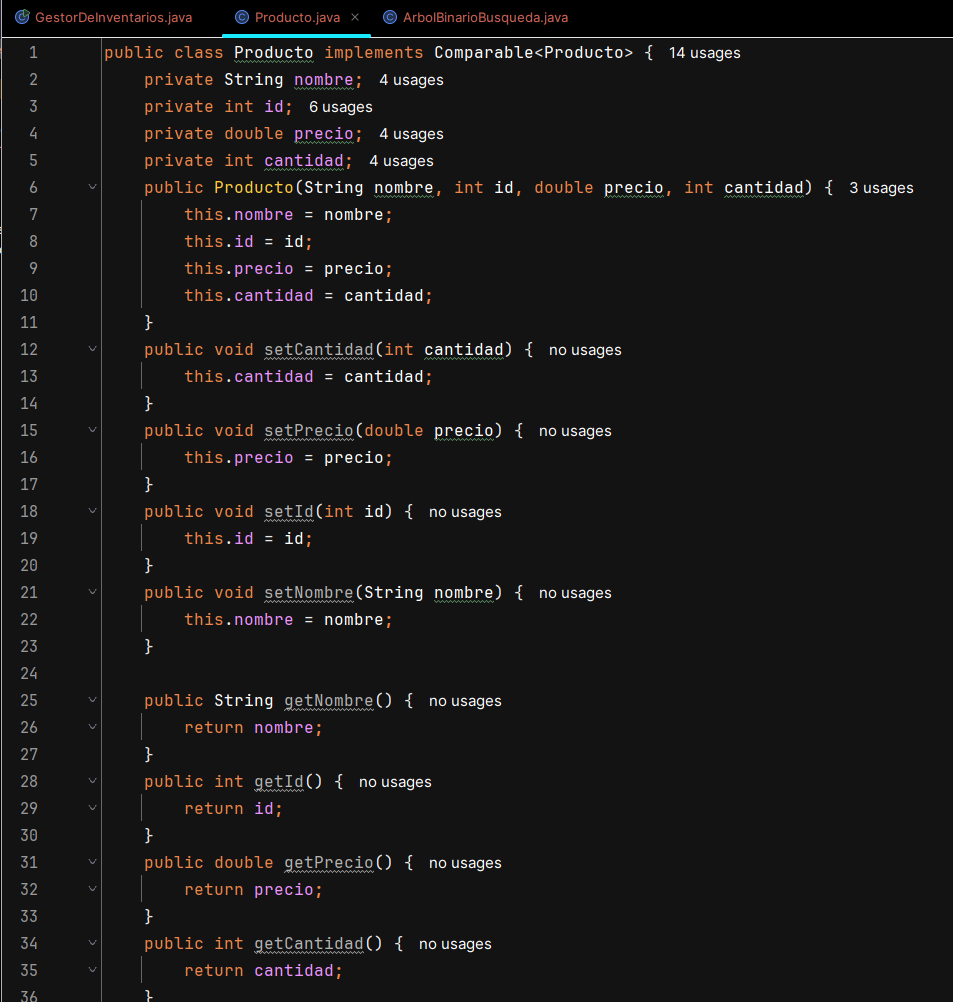
Jesús Hernández Cabrera

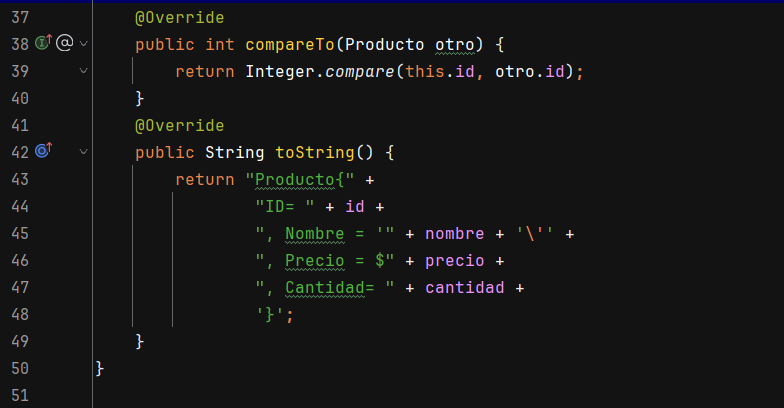
**Gpo:1158**

**URL del repositorio:**

**https://github.com/TyrBalder1439/Estructura-de-Datos-**

**Ciudad Nezahualcóyotl, EDOMEX. 19 DE NOVIEMBRE DEL 2024**

----------------------------------------------------------Clase Producto----------------------------------------------------------



## CLASE PRODUCTO

La clase Producto es una implementación de la interfaz Comparable<Producto>, por lo que permite saber si los objetos de esta clase son capaces de ser comparados entre sí, con lo que es posible ordenarlos.

Atributos: La clase contiene cuatro atributos privados:

* nombre: un String que almacena el nombre del producto.
* id: un int que indica un identificador único para el producto.
* precio: un double que indica el precio del producto.
* cantidad: un int que indica cuántas unidades del producto se encuentran disponibles.

Constructor: El constructor de la clase permite instanciar un objeto Producto inicializando las propiedades de la forma indicada cuando se crea el objeto.

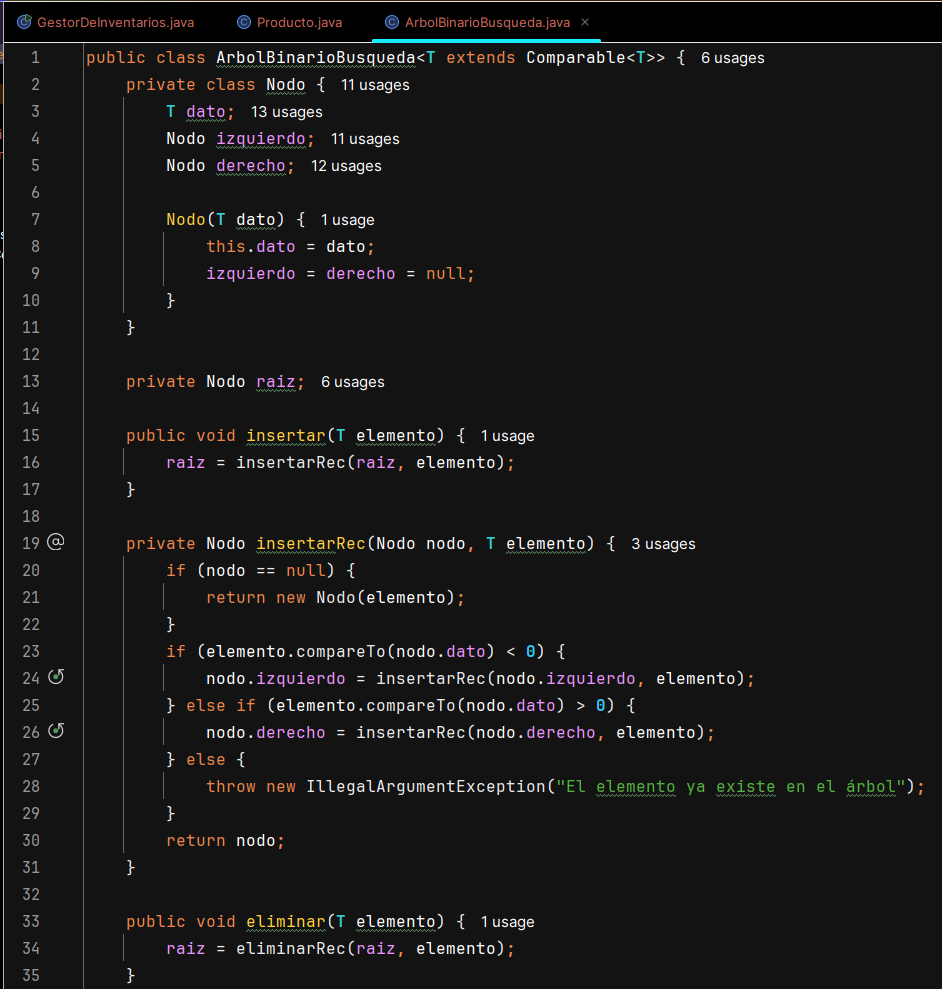
Métodos set y get: Se incluyen métodos para fijar (set) y conseguir (getter) los valores de los atributos. Con esto se consigue encapsular los datos y controlar los accesos a los mismos.

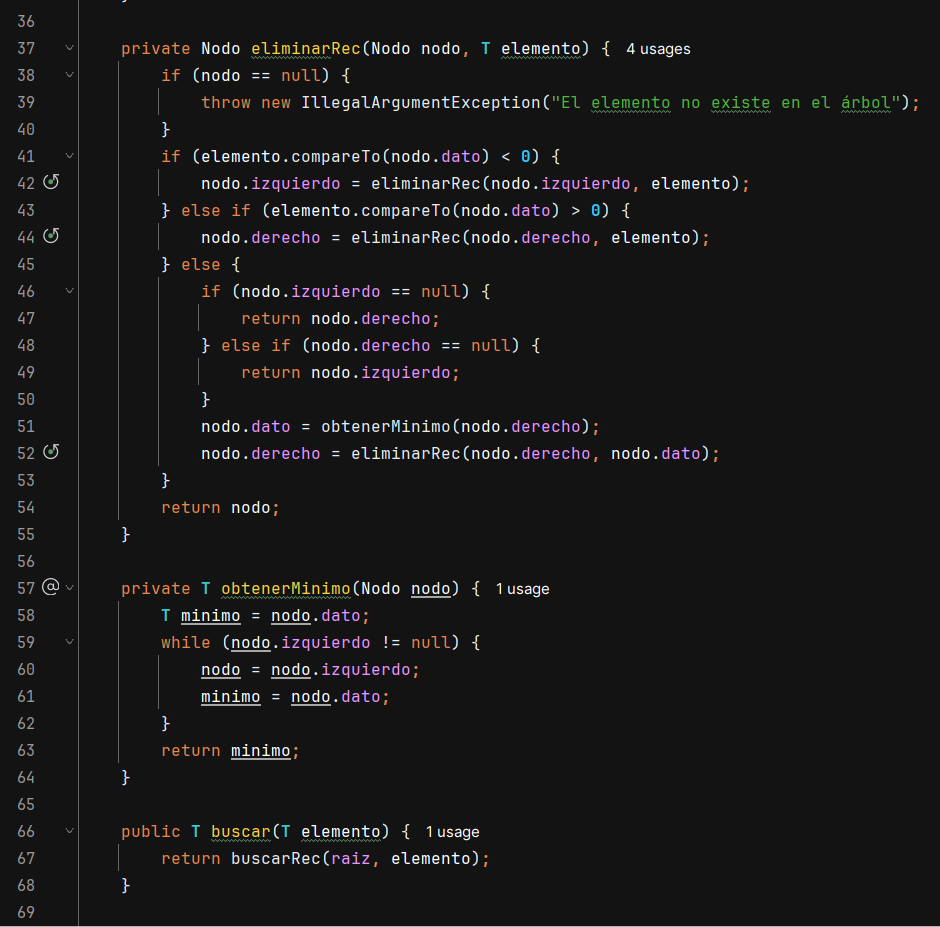
Método compareTo: Este método permite comparar dos objetos Producto mediante el atributo id. El método devuelve un valor negativo, cero o positivo dependiendo de si el id del objeto actual es menor, igual o mayor que el id del objeto pasado como argumento.

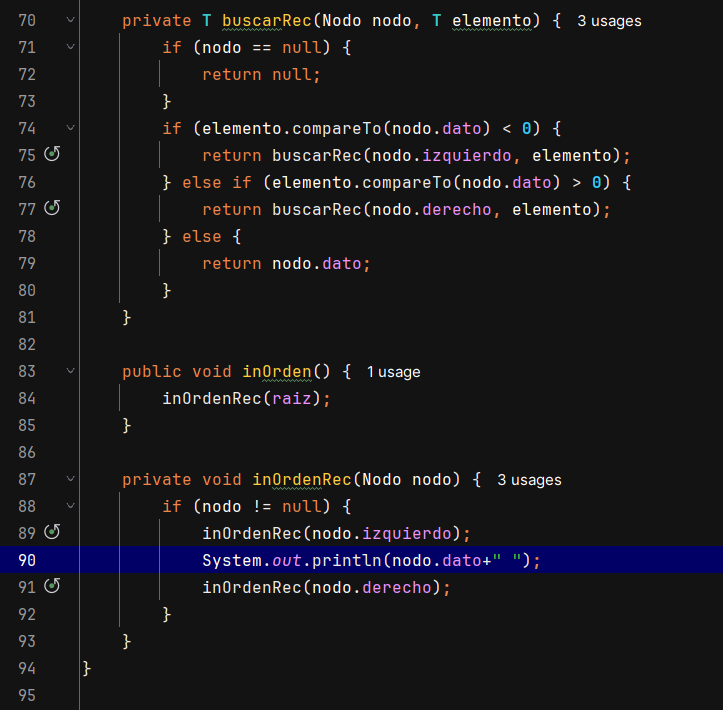
Método toString: El método sobrescribe el método toString de la clase Object y proporciona una representación en forma de cadena del objeto Producto mostrando los atributos de forma legible.

En resumen, esta clase es una representación simple de un producto que incluye atributos, métodos para los atributos y la posibilidad de compararse con otros productos. Esto puede ser útil en aplicaciones que gestionan inventarios o catálogos de productos.

------------------------------------------------Clase ArbolBinarioBusqueda--------------------------------------------







## CLASE ARBOL BINARIO DE BUSQUEDA (ABB)

La clase ArbolBinarioBusqueda es una clase propia que tiene un tipo genérico T que implementa la interfaz Comparable, lo que significa que aquellos elementos que se almacenan dentro del árbol deben ser comparables entre si.

La clase ArbolBinarioBusqueda contiene un nodo interno que es de tipo Nodo y que representa un nodo del árbol. Cada uno de los nodos tiene un valor dato y, además, tiene tanto un hijo izquierdo como un hijo derecho.

La clase ArbolBinarioBusqueda tiene varios métodos, que son:

* insertar(T elemento): que inserta en el árbol un nuevo elemento. Si el elemento ya existe, lanza una excepción("El elemento ya existe en el árbol");
* eliminar(T elemento): que elimina del árbol un elemento. Si el elemento no existe, lanza una excepción ("El elemento no existe en el árbol");
* buscar(T elemento): que busca en el árbol un elemento y devuelve el valor si existe, o devuelve null si no existe;
* inOrden(): que imprime los elementos del árbol en orden de elementos inorden (izquierdo, raíz y derecho).

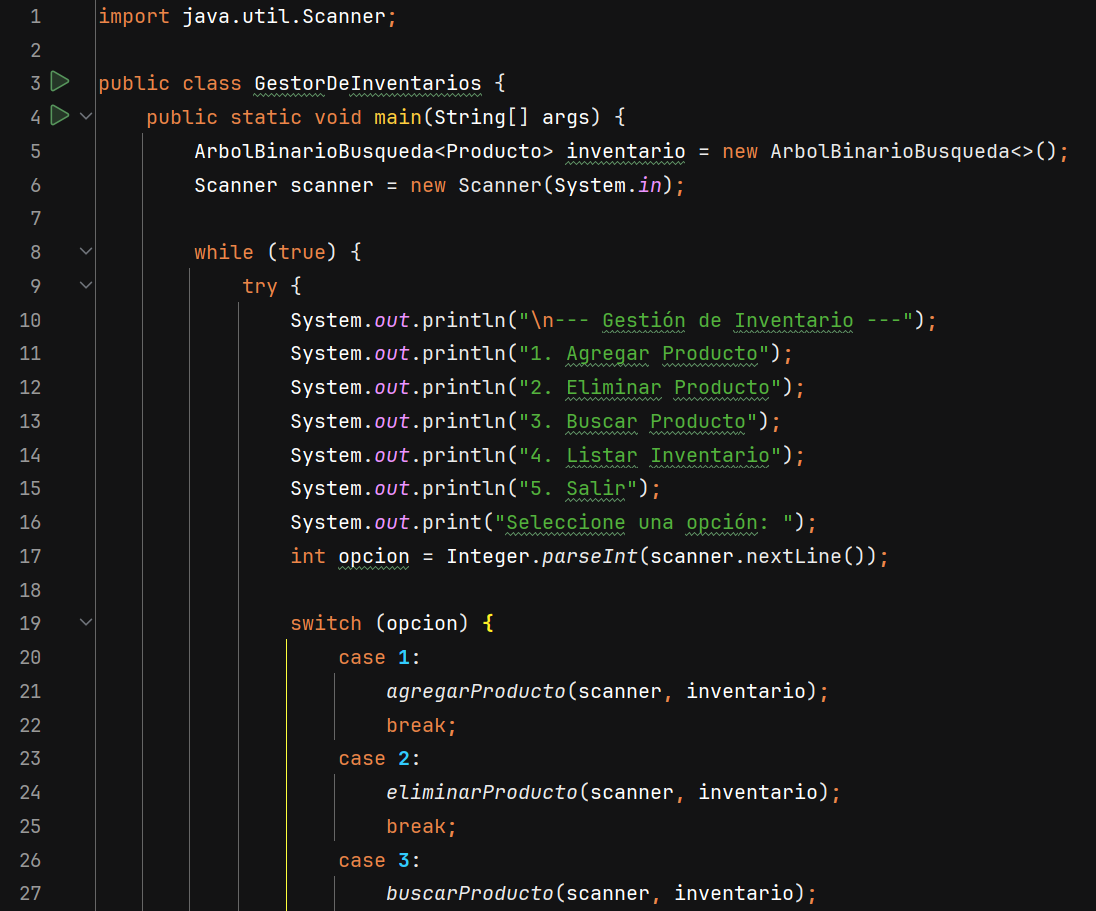
El método insertarRec es un método de carácter recursivo que permite insertar un nuevo elemento dentro del árbol, si el nodo es null crea un nuevo nodo cuyo valor será el de este elemento. En el caso que el elemento sea menor que el valor del nodo actual ejecuta insertarRec al hijo izquierdo. Por el contrario en el caso de que el elemento sea mayor que el valor del nodo actual ejecuta insertarRec al hijo derecho.

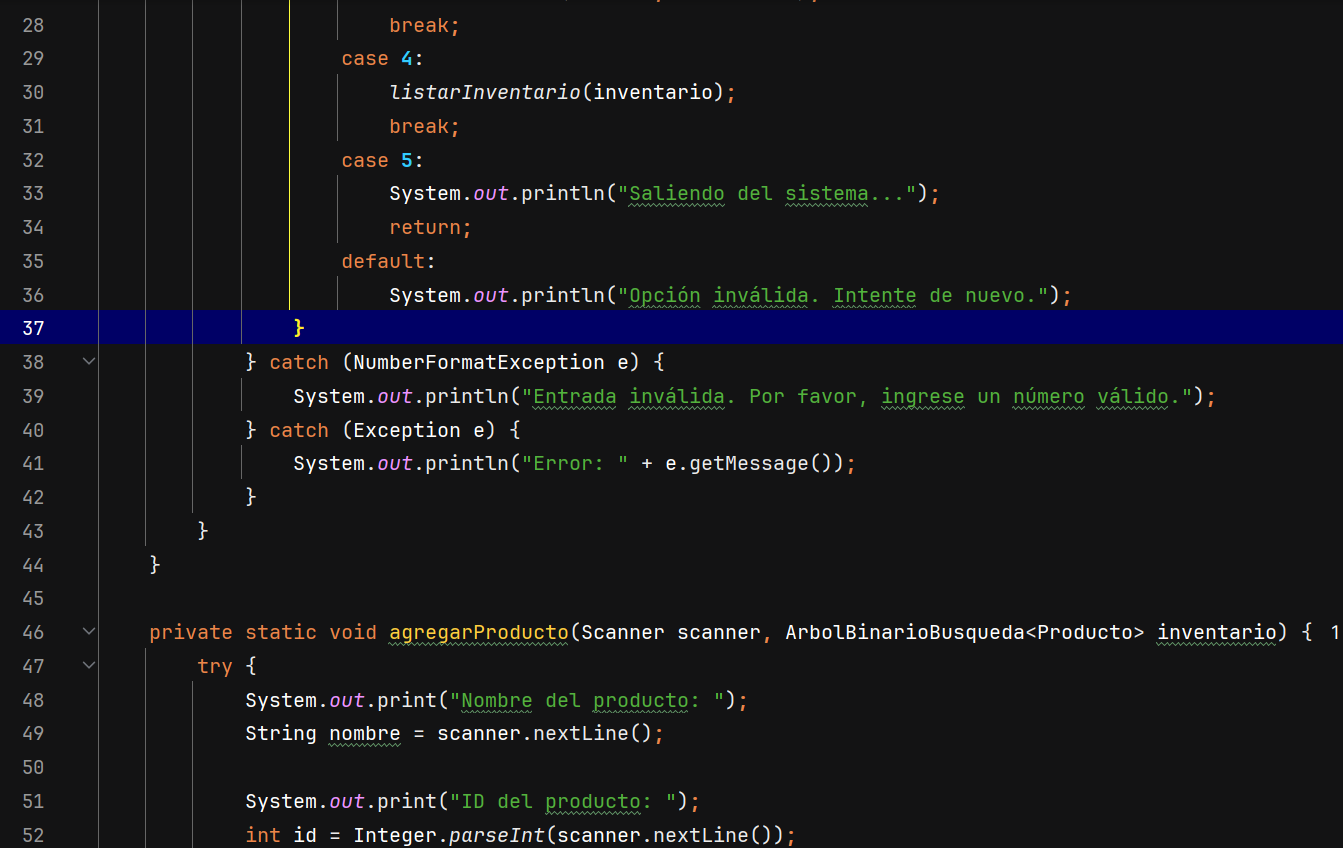
El método eliminarRec es un método de carácter recursivo que permite eliminar un elemento del árbol en caso de que el nodo en cuestión sea null se lanza una excepción, en caso de que el elemento sea menor que el del nodo actual se ejecuta eliminarRec al hijo izquierdo, si el elemento es mayor que el valor del nodo actual ejecuta eliminarRec a su hijo derecho, y en el caso en que el elemento es igual que el del nodo actual se elimina el nodo y se sustituye su valor por el mínimo valor del subárbol derecho.

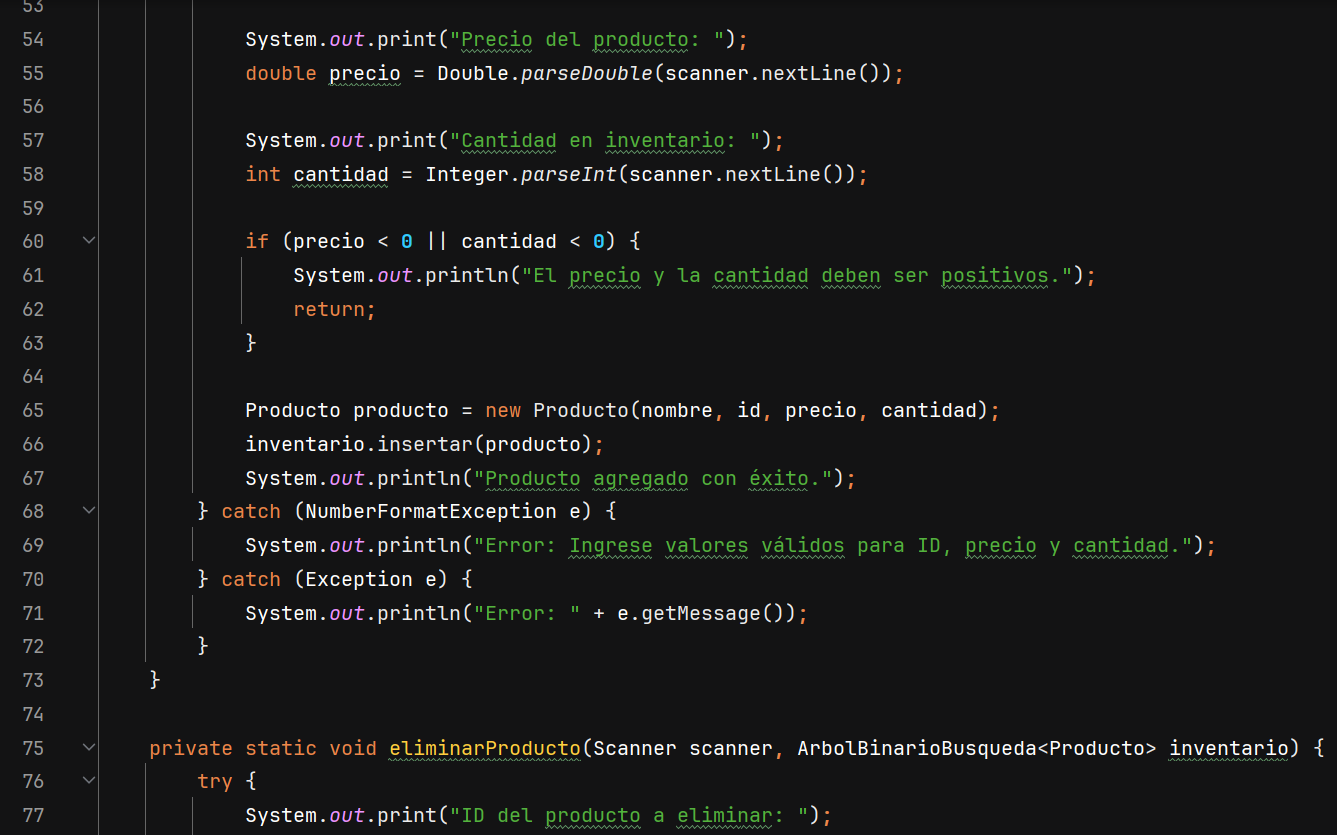
El método obtenerMinimo devuelve el mínimo valor de un subárbol.

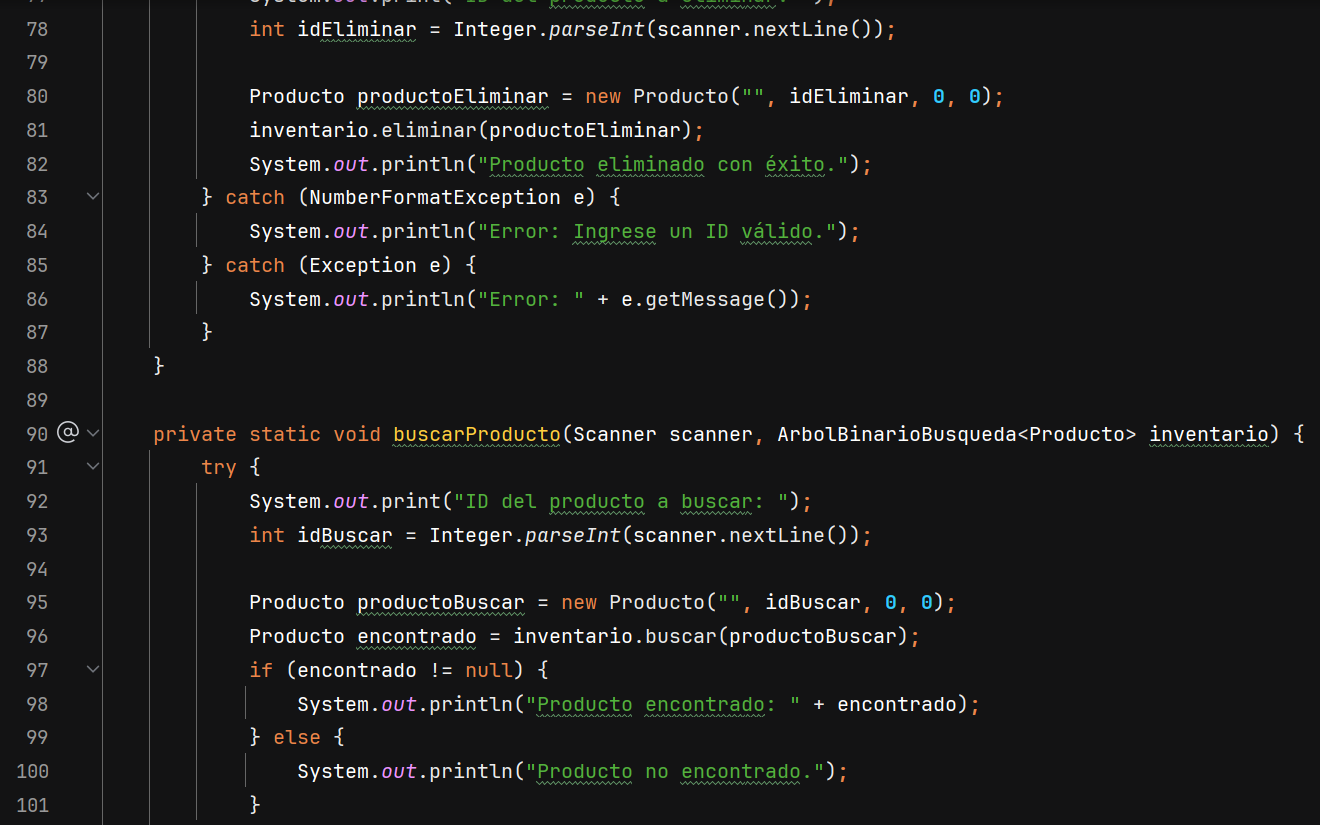
El método buscarRec es un método recursivo que busca un elemento en el árbol. Si el nodo es null devolverá null, si el elemento es menor que el del nodo actual ejecuta buscarRec al hijo izquierdo si el elemento es mayor que el del nodo actual ejecuta buscarRec al hijo derecho, si el elemento es igual que el dato del nodo devuelvo su valor.

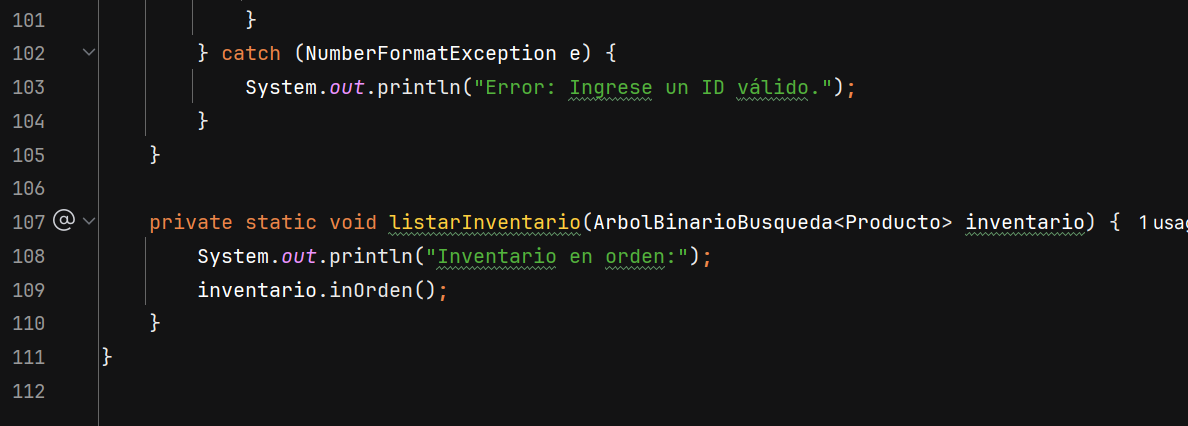
El método inOrdenRec: Este método recursivo realiza el recorrido en orden. Si el nodo no es null, primero llama a sí mismo para el subárbol izquierdo, luego imprime el dato del nodo actual y finalmente llama a sí mismo para el subárbol derecho. Esto resulta en una salida ordenada de los elementos del árbol.

------------------------------------------CLASE GESTION DE INVENTARIO-------------------------------------------









## GESTOR DE INVENTARIO

Importaciones y Clase Principal:

Importamos java.util.Scanner para poder recibir datos por teclado

La clase GestorDeInventarios contiene el método main que es donde inicia el programa.

Estructura del Menú:

Creamos un bucle while(true) para poder estar interactuando de forma continua hasta que el usuario se decida por salir.Dentro de ese bucle se le presenta al usuario un menú con opciones para añadir, eliminar, buscar y listar productos así como para salir del programa.

Manejo de Opciones: Se utiliza un switch para gestionar la opción seleccionada por el usuario. Según la opción elegida, se invocan diferentes métodos (agregarProducto, eliminarProducto, buscarProducto, listarInventario).

Método agregarProducto: Este método permite al usuario ingresar los detalles de un nuevo producto (nombre, ID, precio y cantidad). Se verifica que el precio y la cantidad sean positivos antes de crear un objeto Producto y añadirlo al inventario.

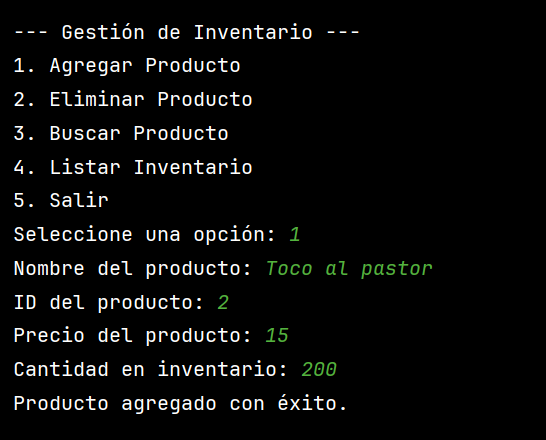
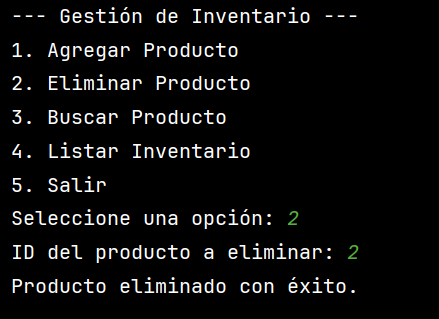
Método eliminarProducto: Permite al usuario eliminar un producto del inventario ingresando su ID. Se crea un objeto Producto con el ID proporcionado y se llama al método eliminar del árbol binario de búsqueda.

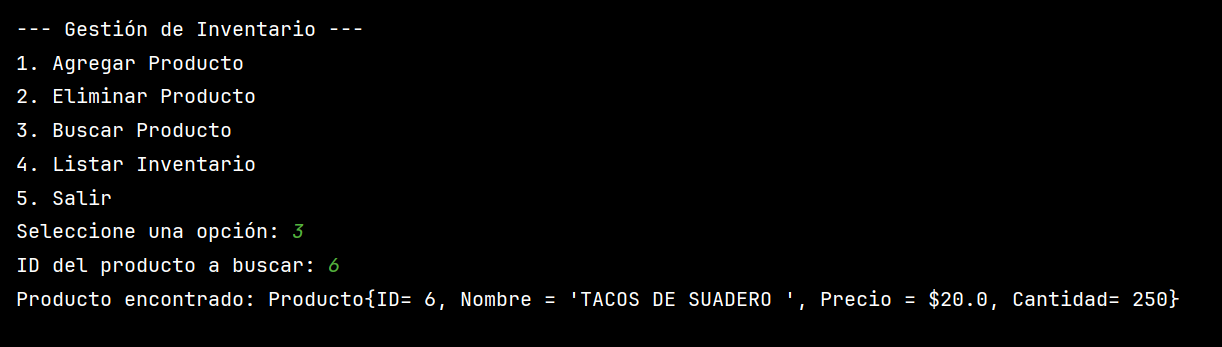
Método buscarProducto: Facilita la búsqueda de un producto en el inventario por su ID. Si se encuentra el producto, se muestra su información; de lo contrario, se notifica que no se encontró.

Método listarInventario: Este método imprime todos los productos en el inventario en orden, utilizando el método inOrden del árbol binario de búsqueda.

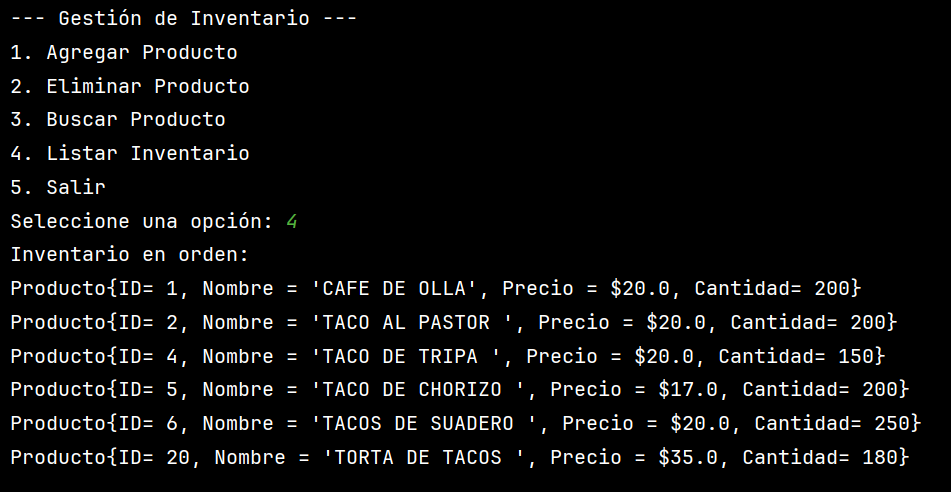
Manejo de Errores: Se implementan bloques try-catch para gestionar excepciones, como entradas no válidas, asegurando que el programa no se detenga ante errores.

## EJECUCION DEL PROGRAMA

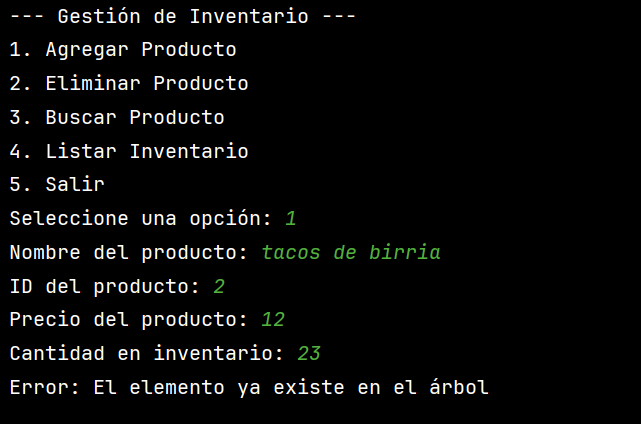
* AGREGAR PRODUCTO
* ELIMINAR PRODUCTO
* BUSCAR PRODUCTO

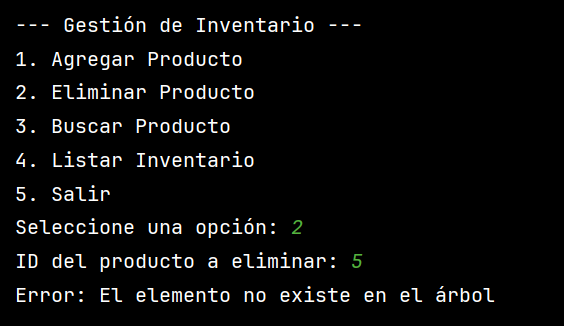
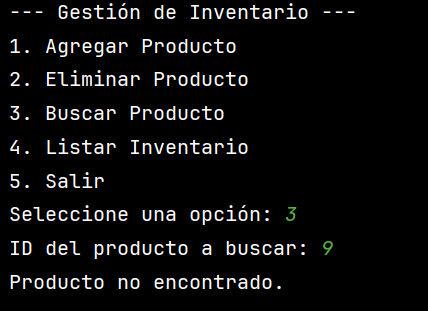
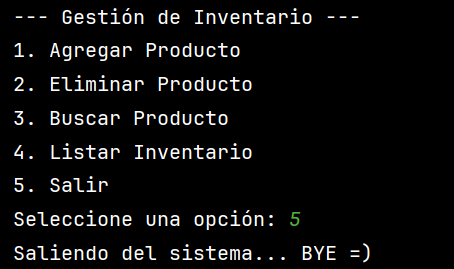


* LISTAR INVENTARIO



* Ejemplo de duplicación de ID



*  Eliminación de productos inexistentes.
* Intentos de búsqueda de ID inexistentes
* Salir