

Semana 4

Desarrollo Orientado a Objetos II (PRY2203)

Formato de respuesta

Nombre estudiante: Jorge Gallardo Heck, Pedro Breit Lira	
Asignatura: Desarrollo Orientado a Objetos II	Carrera: Analista Programador Computacional
Profesor: Eithel Gonzalez Rojas	Fecha: 08/09/2025

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

```

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
}

```

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 nuevos productos (nombre, código, precio base, stock)	Se deben ingresar todos los valores para poder listarse de forma correcta un producto. Se verifica si el código existe. Si ya existe con anterioridad no puede ingresarse el nuevo producto Los requisitos a ingresar son claros para evitar errores. Una vez ingresado se actualiza la lista de productos Respuesta rápida en menos de 2 segundos.
Modificar stock	Modificar stock y actualizar la información. Estos valores deben ser numéricos.
Eliminar producto	Eliminar y actualizar la información
Cargar la lista de productos existentes una vez iniciado el programa (Utilizar Hashmap utilizando el código como la Key)	Debe almacenar la información previamente ingresada sin pérdida de datos. Da flexibilidad a la búsqueda de datos por medio de Hashmap
Listar el inventario completo de productos ingresado	Debe actualizarse de forma interna para reflejar los cambios
Se utiliza patrón de diseño Command	Da mayor flexibilidad a la hora de ingresar comandos
Buscar productos por código	Comprobar si existe el producto, devolver la descripción del producto

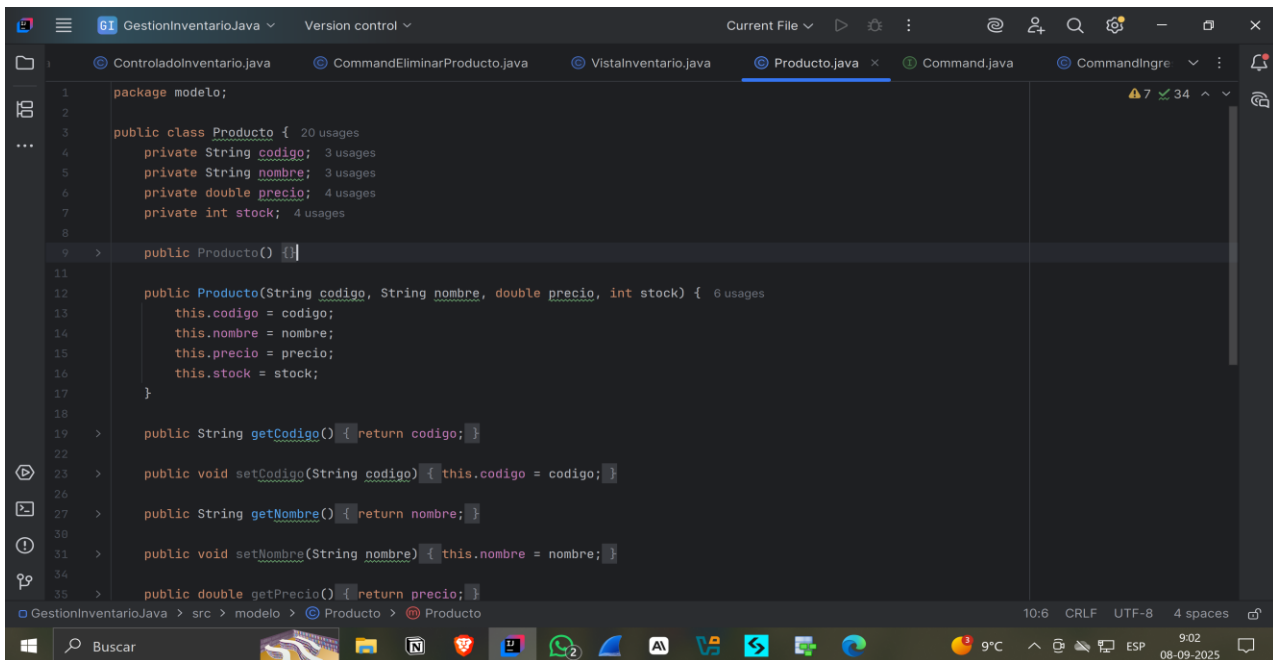
(Elimina o añade filas según sea necesario)

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:

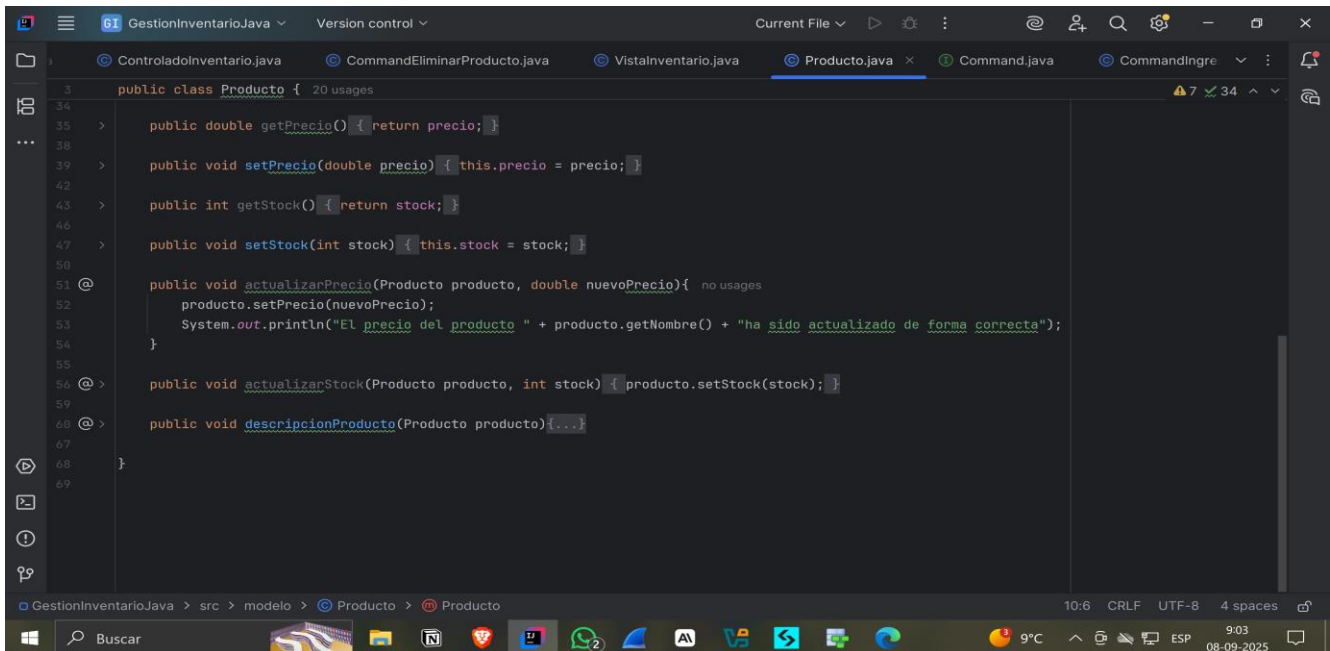
2.1 Clase Producto

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.



```
1 package modelo;
2
3 public class Producto { 20 usages
4     private String codigo; 3 usages
5     private String nombre; 3 usages
6     private double precio; 4 usages
7     private int stock; 4 usages
8
9     public Producto() {}
10
11     public Producto(String codigo, String nombre, double precio, int stock) { 6 usages
12         this.codigo = codigo;
13         this.nombre = nombre;
14         this.precio = precio;
15         this.stock = stock;
16     }
17
18     public String getCodigo() { return codigo; }
19
20     public void setCodigo(String codigo) { this.codigo = codigo; }
21
22     public String getNombre() { return nombre; }
23
24     public void setNombre(String nombre) { this.nombre = nombre; }
25
26     public double getPrecio() { return precio; }
```

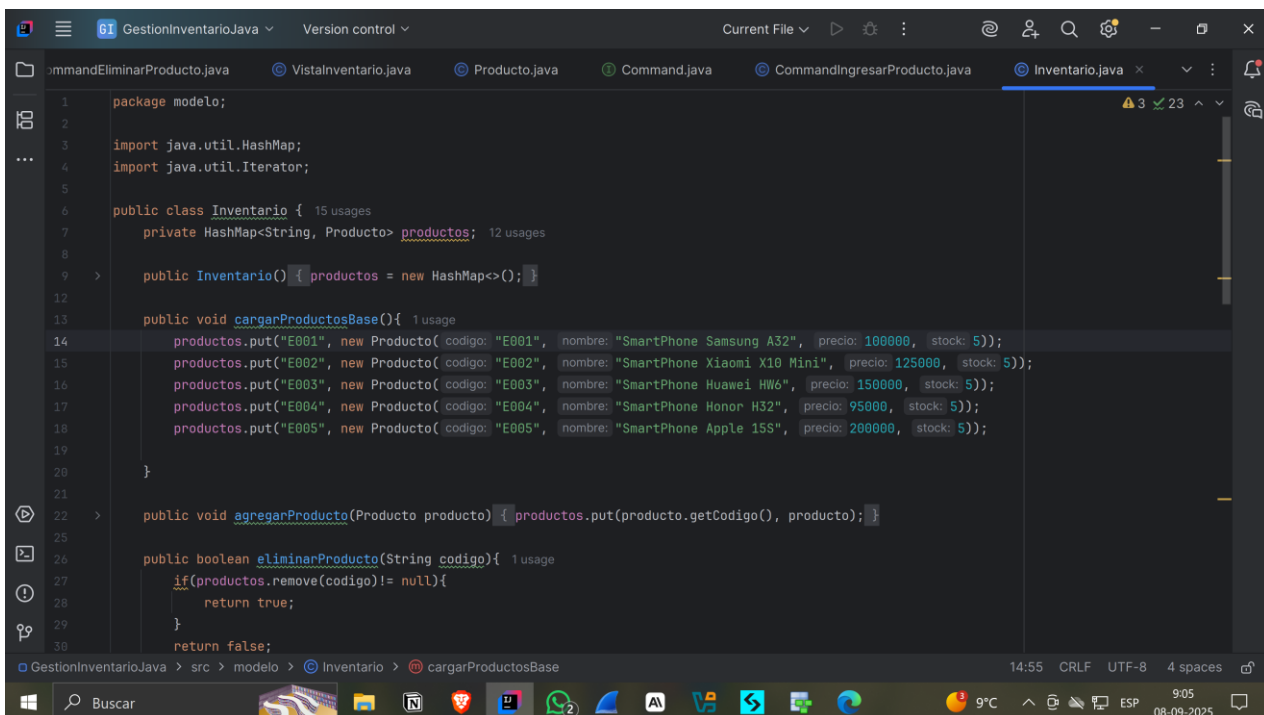
The screenshot shows an IDE window titled 'GestionInventarioJava' with a 'Version control' dropdown. The 'Current File' dropdown is set to 'Producto.java'. The code editor displays the 'Producto' class with its attributes and methods. The attributes are 'codigo' (String), 'nombre' (String), 'precio' (double), and 'stock' (int). The methods include a no-argument constructor, a parameterized constructor, and getters/setters for each attribute. The IDE interface includes a sidebar with a file explorer showing the project structure: 'GestionInventarioJava' > 'src' > 'modelo' > 'Producto'. The status bar at the bottom shows the file encoding as 'UTF-8' and the line ending as 'CRLF'.



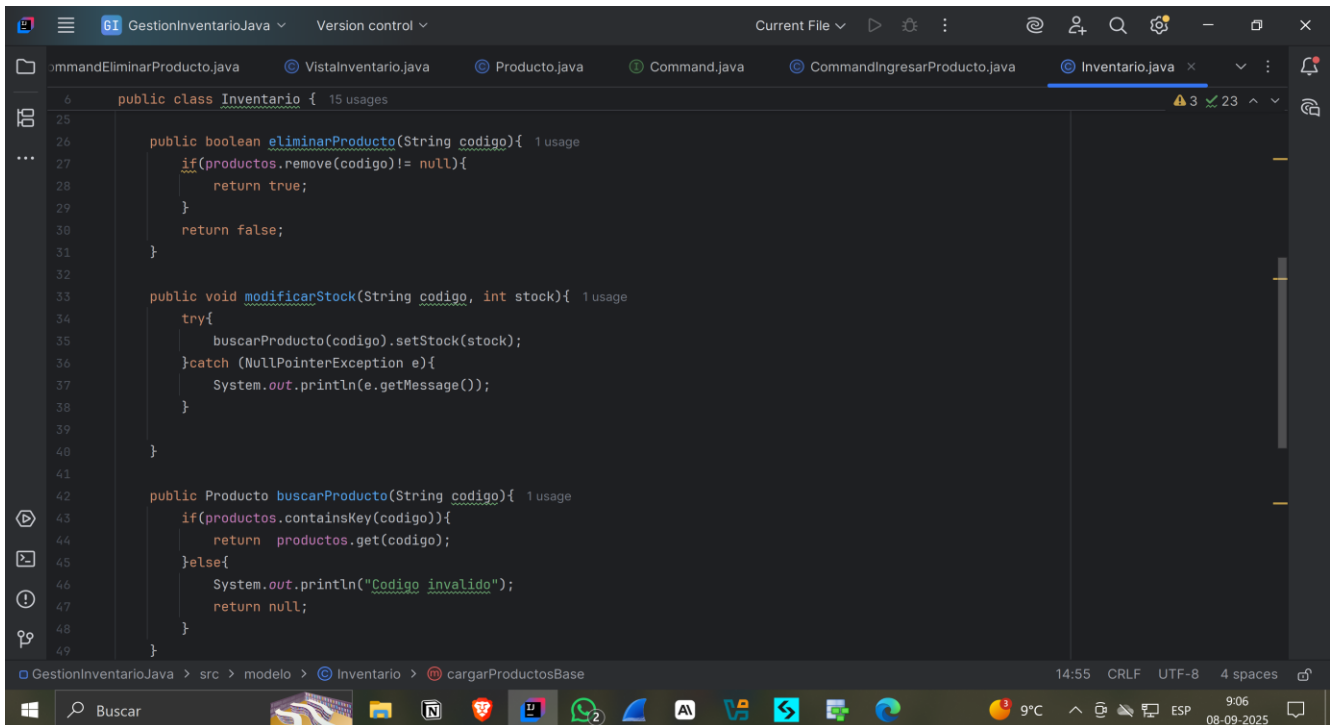
```
3 public class Producto { 20 usages
34
35 > public double getPrecio() { return precio; }
38
39 > public void setPrecio(double precio) { this.precio = precio; }
42
43 > public int getStock() { return stock; }
46
47 > public void setStock(int stock) { this.stock = stock; }
50
51 @ public void actualizarPrecio(Producto producto, double nuevoPrecio){ no usages
52     producto.setPrecio(nuevoPrecio);
53     System.out.println("El precio del producto " + producto.getNombre() + "ha sido actualizado de forma correcta");
54 }
55
56 @ public void actualizarStock(Producto producto, int stock){ producto.setStock(stock); }
59
60 @ public void descripcionProducto(Producto producto){...}
67
68 }
69
```

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.

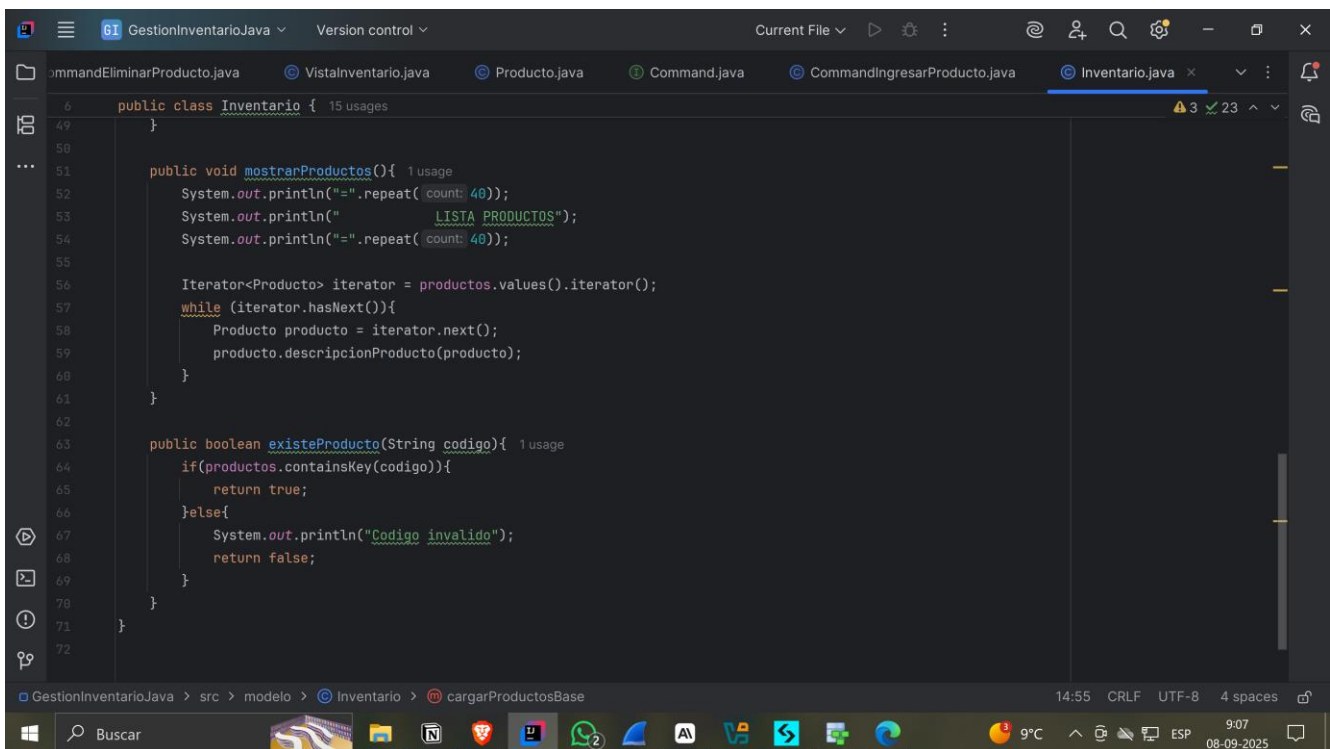


```
1 package modelo;
2
3 import java.util.HashMap;
4 import java.util.Iterator;
5
6 public class Inventario { 15 usages
7     private HashMap<String, Producto> productos; 12 usages
8
9 > public Inventario() { productos = new HashMap<>(); }
12
13 public void cargarProductosBase(){ 1 usage
14     productos.put("E001", new Producto( codigo: "E001", nombre: "SmartPhone Samsung A32", precio: 100000, stock: 5));
15     productos.put("E002", new Producto( codigo: "E002", nombre: "SmartPhone Xiaomi X10 Mini", precio: 125000, stock: 5));
16     productos.put("E003", new Producto( codigo: "E003", nombre: "SmartPhone Huawei HW6", precio: 150000, stock: 5));
17     productos.put("E004", new Producto( codigo: "E004", nombre: "SmartPhone Honor H32", precio: 95000, stock: 5));
18     productos.put("E005", new Producto( codigo: "E005", nombre: "SmartPhone Apple 15S", precio: 200000, stock: 5));
19
20 }
21
22 > public void agregarProducto(Producto producto) { productos.put(producto.getCodigo(), producto); }
25
26 public boolean eliminarProducto(String codigo){ 1 usage
27     if(productos.remove(codigo)!= null){
28         return true;
29     }
30     return false;
31 }
```



This screenshot shows the 'eliminarProducto' method in the 'Inventario' class. The method takes a 'codigo' string as input and attempts to remove it from the 'productos' map. If the removal is successful (i.e., the map was not null), it returns true; otherwise, it returns false. Below this, the 'modificarStock' method is shown, which uses 'buscarProducto' to find an item and then updates its stock. The 'buscarProducto' method checks if the 'codigo' exists in the map; if it does, it returns the corresponding 'Producto' object; if not, it prints an error message and returns null. The IDE interface includes a sidebar with project files, a top toolbar with various icons, and a bottom status bar showing file encoding (UTF-8) and line endings (CRLF).

```
6 public class Inventario { 15 usages
25
26     public boolean eliminarProducto(String codigo){ 1 usage
27         if(productos.remove(codigo)!= null){
28             return true;
29         }
30         return false;
31     }
32
33     public void modificarStock(String codigo, int stock){ 1 usage
34         try{
35             buscarProducto(codigo).setStock(stock);
36         }catch (NullPointerException e){
37             System.out.println(e.getMessage());
38         }
39     }
40
41
42     public Producto buscarProducto(String codigo){ 1 usage
43         if(productos.containsKey(codigo)){
44             return productos.get(codigo);
45         }else{
46             System.out.println("Codigo invalido");
47             return null;
48         }
49     }
}
```



This screenshot shows the 'mostrarProductos' and 'existeProducto' methods in the 'Inventario' class. The 'mostrarProductos' method prints a separator line, a header 'LISTA PRODUCTOS', and another separator line. It then uses an 'Iterator' to loop through the 'productos' map, printing the description of each product. The 'existeProducto' method checks if a 'codigo' exists in the map and returns a boolean result. The IDE interface is consistent with the previous screenshot, showing the same project structure and status bar information.

```
49
50
51     public void mostrarProductos(){ 1 usage
52         System.out.println("=".repeat(40));
53         System.out.println("LISTA PRODUCTOS");
54         System.out.println("=".repeat(40));
55
56         Iterator<Producto> iterator = productos.values().iterator();
57         while (iterator.hasNext()){
58             Producto producto = iterator.next();
59             producto.descripcionProducto(producto);
60         }
61     }
62
63     public boolean existeProducto(String codigo){ 1 usage
64         if(productos.containsKey(codigo)){
65             return true;
66         }else{
67             System.out.println("Codigo invalido");
68             return false;
69         }
70     }
71
72 }
```

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.

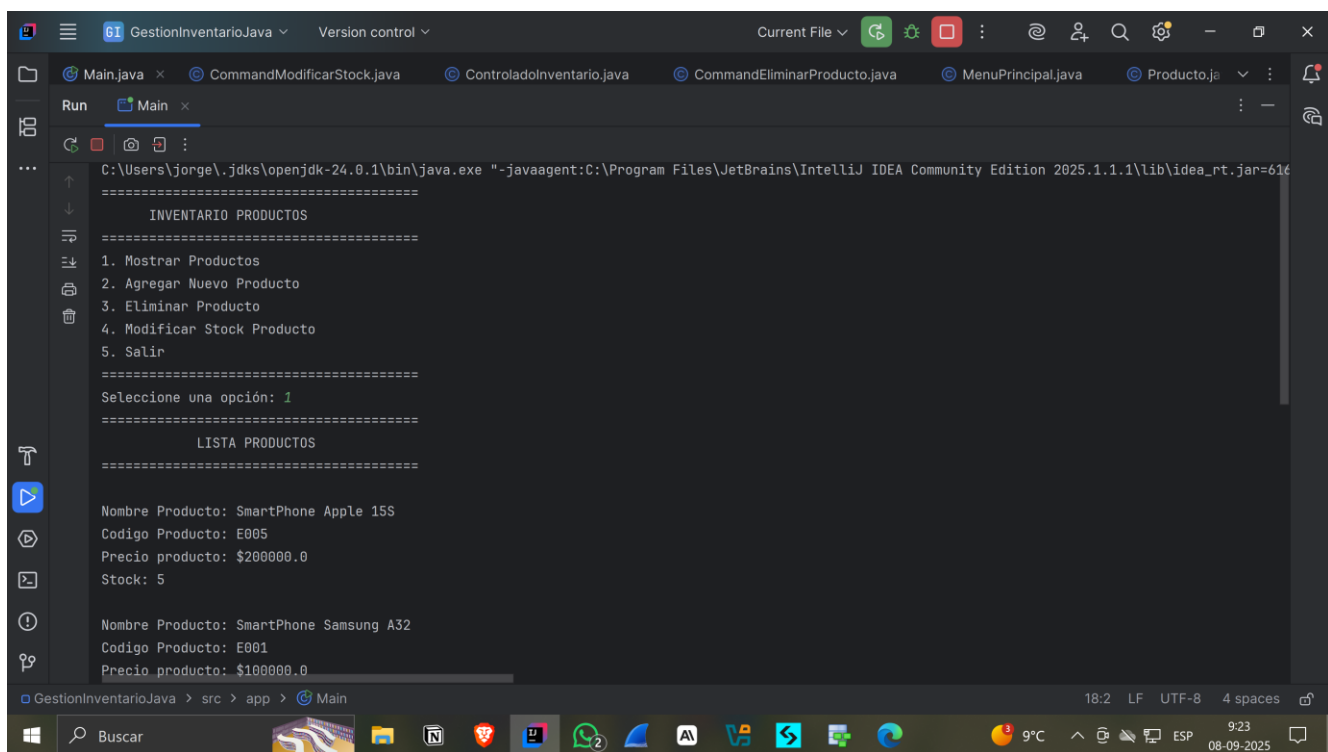
Paso 4: Fase de pruebas

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.

Se muestra la lista de productos:



```
C:\Users\jorge\.jdk\openjdk-24.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2025.1.1.1\lib\idea_rt.jar=616...

=====
INVENTARIO PRODUCTOS
=====
1. Mostrar Productos
2. Agregar Nuevo Producto
3. Eliminar Producto
4. Modificar Stock Producto
5. Salir
=====
Seleccione una opción: 1
=====
LISTA PRODUCTOS
=====
Nombre Producto: SmartPhone Apple 15S
Codigo Producto: E005
Precio producto: $200000.0
Stock: 5

Nombre Producto: SmartPhone Samsung A32
Codigo Producto: E001
Precio producto: $100000.0
```

Búsqueda de producto por código:

```
Nombre Producto: SmartPhone Honor H32
Codigo Producto: E004
Precio producto: $95000.0
Stock: 5
=====
INVENTARIO PRODUCTOS
=====
1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir
=====
Seleccione una opción: 2
Indica el CODIGO: E004

Nombre Producto: SmartPhone Honor H32
Codigo Producto: E004
Precio producto: $95000.0
Stock: 5
=====
```

Agregar producto al Inventario:

```
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir
=====
Seleccione una opción: 3
Indica el CODIGO: C007
Indica el NOMBRE del Producto: Tablet Samsung A7s
Ingresa el Precio: 89000
Ingresa el Stock: 6
Producto Agregado de forma correcta al Inventario

=====
INVENTARIO PRODUCTOS
=====
1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir
=====
Seleccione una opción: |
```

```
1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir
=====
Seleccione una opción: 1
=====
LISTA PRODUCTOS
=====
Nombre Producto: SmartPhone Apple 15S
Codigo Producto: E005
Precio producto: $200000.0
Stock: 5

Nombre Producto: TABLET SAMSUNG A7S
Codigo Producto: C007
Precio producto: $89000.0
Stock: 6

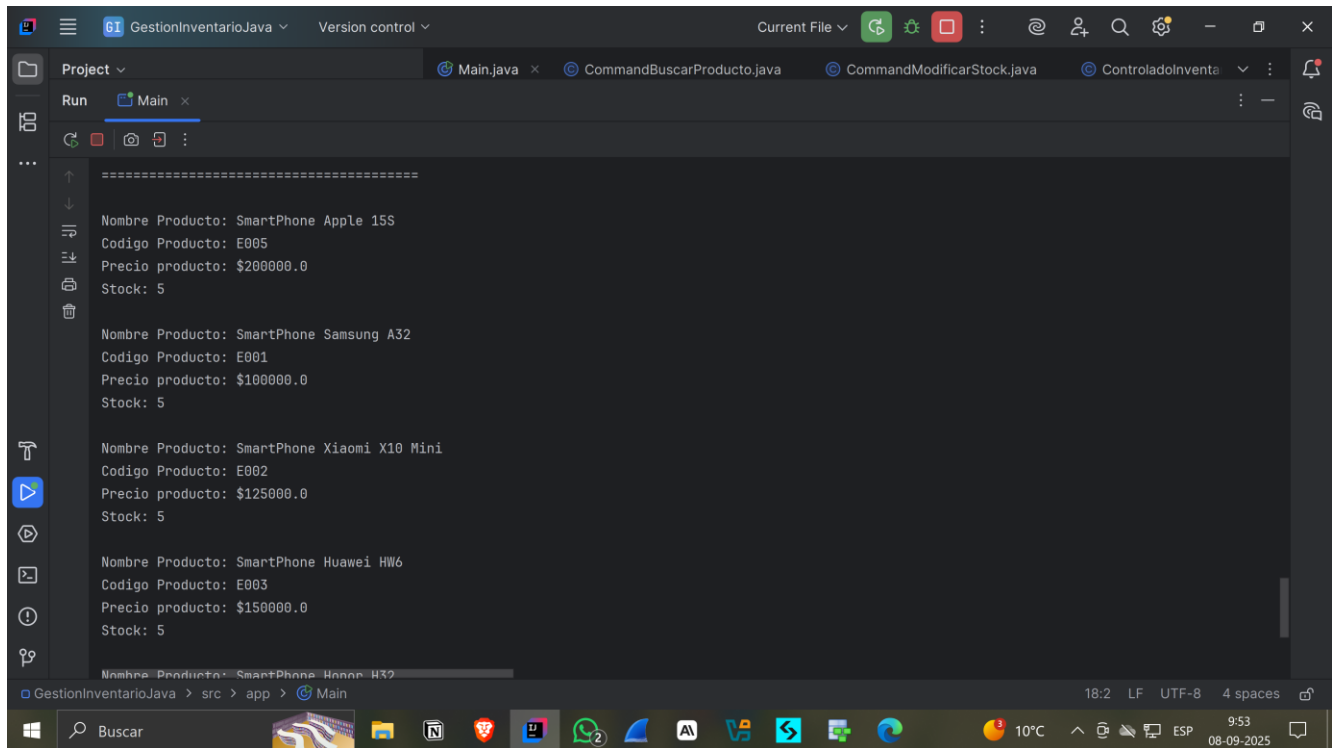
Nombre Producto: SmartPhone Samsung A32
```

Eliminar Producto por código ingresado (HashMap):

```
1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir
=====
Seleccione una opción: 4
Indica el CODIGO: C007
Producto codigo: C007 ha sido eliminado de forma correcta

=====
INVENTARIO PRODUCTOS
=====
1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir
=====
Seleccione una opción: |
```

(No se presenta en la lista de productos ingresados. Se ha eliminado de forma correcta)



```
=====
Nombre Producto: SmartPhone Apple 15S
Codigo Producto: E005
Precio producto: $200000.0
Stock: 5

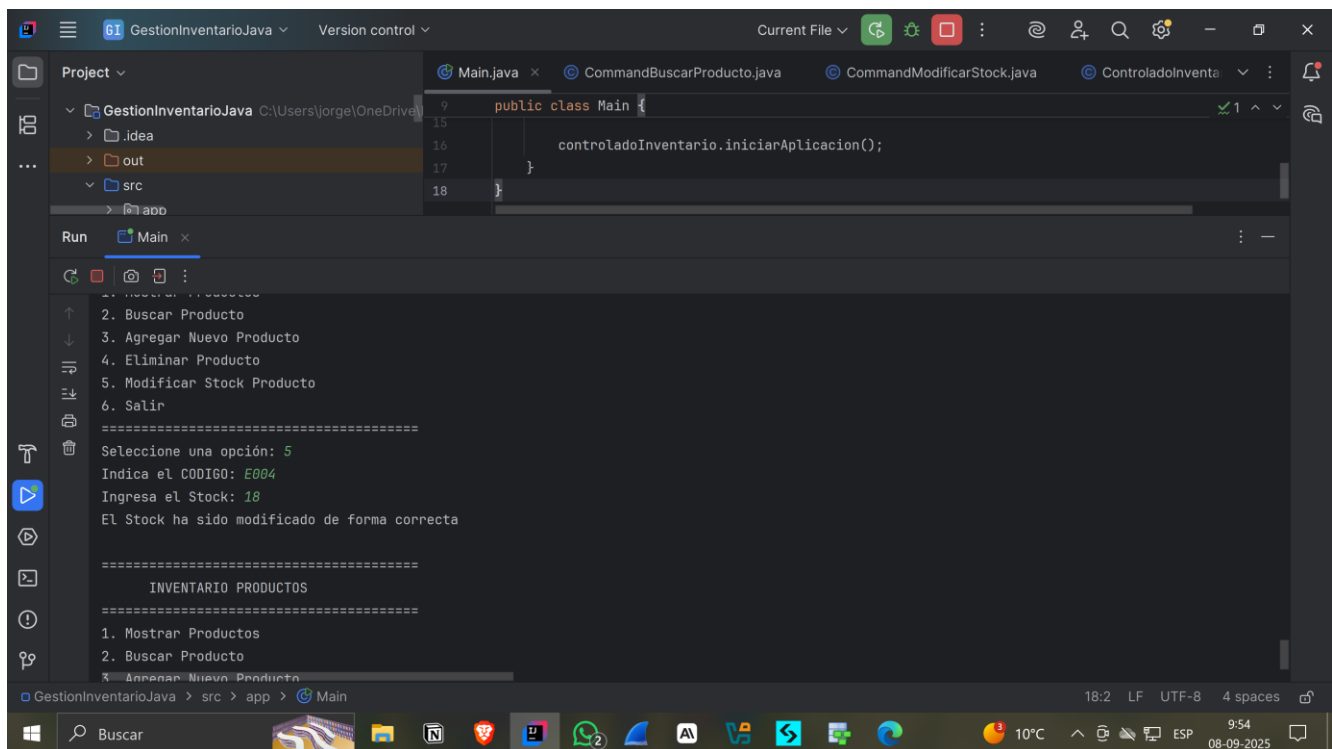
Nombre Producto: SmartPhone Samsung A32
Codigo Producto: E001
Precio producto: $100000.0
Stock: 5

Nombre Producto: SmartPhone Xiaomi X10 Mini
Codigo Producto: E002
Precio producto: $125000.0
Stock: 5

Nombre Producto: SmartPhone Huawei HW6
Codigo Producto: E003
Precio producto: $150000.0
Stock: 5

Nombre Producto: SmartPhone Honor H32
```

Modificar Stock:



```
9 public class Main {
15
16     controladoInventario.iniciarAplicacion();
17 }
18 }
```

```
=====
1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir

Seleccione una opción: 5
Indica el CODIGO: E004
Ingresa el Stock: 18
El Stock ha sido modificado de forma correcta

=====
INVENTARIO PRODUCTOS
=====
1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
```

```
public class Main {  
    15  
    16        controladoInventario.iniciarAplicacion();  
    17    }  
    18}
```

Run Main x

Nombre Producto: SmartPhone Xiaomi X10 Mini
Codigo Producto: E002
Precio producto: \$125000.0
Stock: 5

Nombre Producto: SmartPhone Huawei HW6
Codigo Producto: E003
Precio producto: \$150000.0
Stock: 5

Nombre Producto: SmartPhone Honor H32
Codigo Producto: E004
Precio producto: \$95000.0
Stock: 18

=====

INVENTARIO PRODUCTOS

=====

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’:

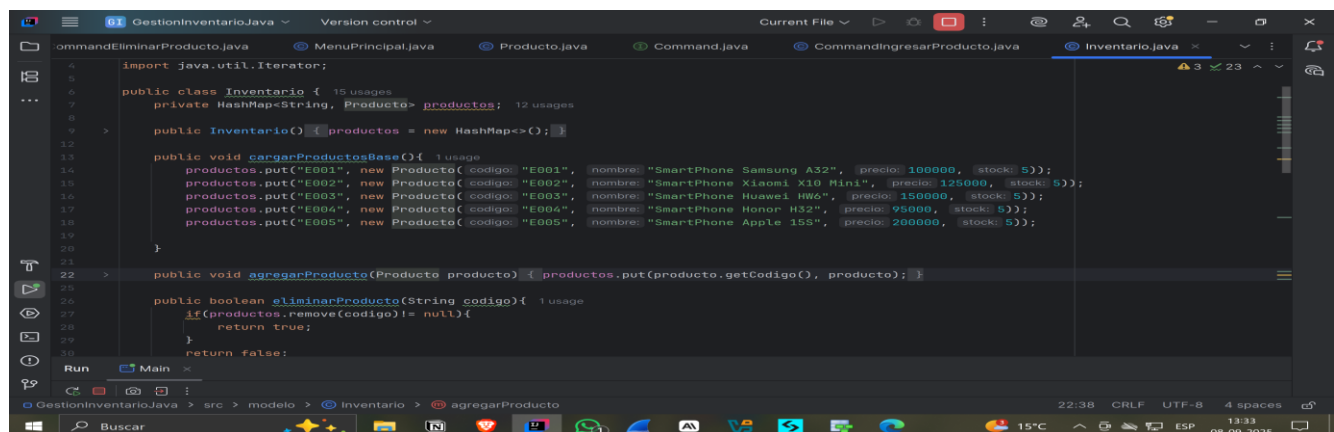
- a) Creación de producto: verificar que se pueda crear un objeto ‘Producto’ con todos los atributos esperados.
- Comprobar que los valores asignados a los atributos del producto, como nombre, descripción, precio y cantidad en stock, son correctos tras la inicialización.
- b) Actualización de atributos: 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’:

- a) 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.
- b) 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.
- c) 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.
- d) 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:

Existe respuesta de la “Base de Datos”:



```
import java.util.Iterator;

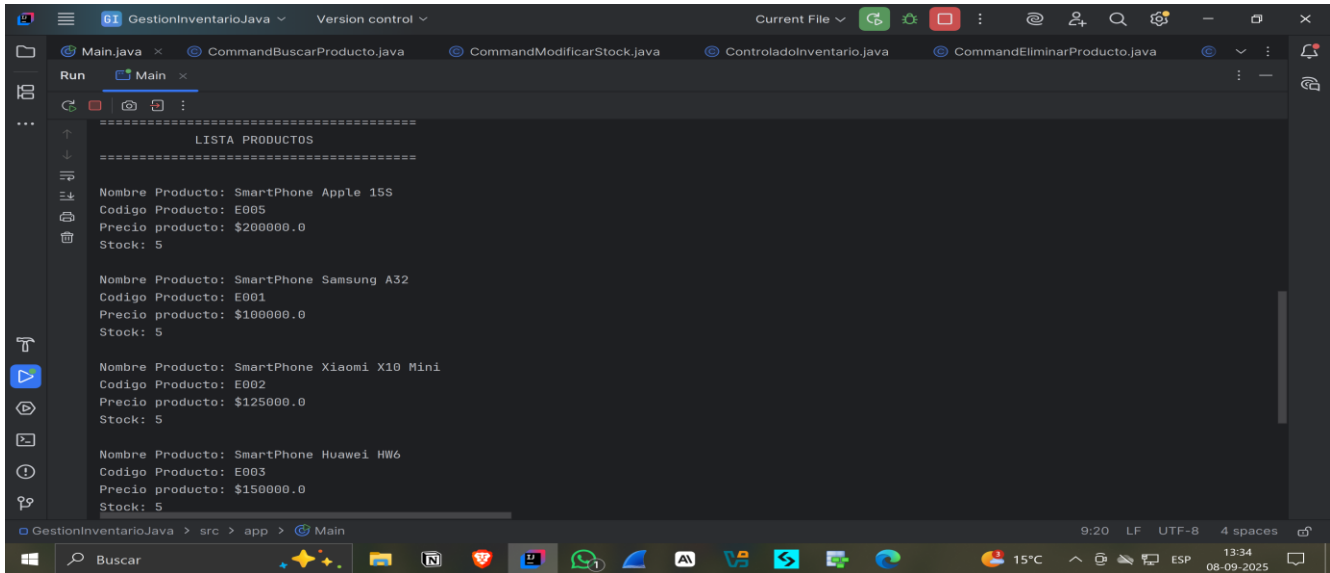
public class Inventario {
    private HashMap<String, Producto> productos;

    public Inventario() { productos = new HashMap<>(); }

    public void cargarProductosBase() {
        productos.put("E001", new Producto( "E001", nombre: "SmartPhone Samsung A32", precio: 100000, stock: 5));
        productos.put("E002", new Producto( "E002", nombre: "SmartPhone Xiaomi X10 Mini", precio: 125000, stock: 5));
        productos.put("E003", new Producto( "E003", nombre: "SmartPhone Huawei HW6", precio: 150000, stock: 5));
        productos.put("E004", new Producto( "E004", nombre: "SmartPhone Honor H32", precio: 95000, stock: 5));
        productos.put("E005", new Producto( "E005", nombre: "SmartPhone Apple 155", precio: 200000, stock: 5));
    }

    public void agregarProducto(Producto producto) { productos.put(producto.getCodigo(), producto); }

    public boolean eliminarProducto(String codigo) {
        if (productos.remove(codigo) != null) {
            return true;
        }
        return false;
    }
}
```



```
=====
LISTA PRODUCTOS
=====

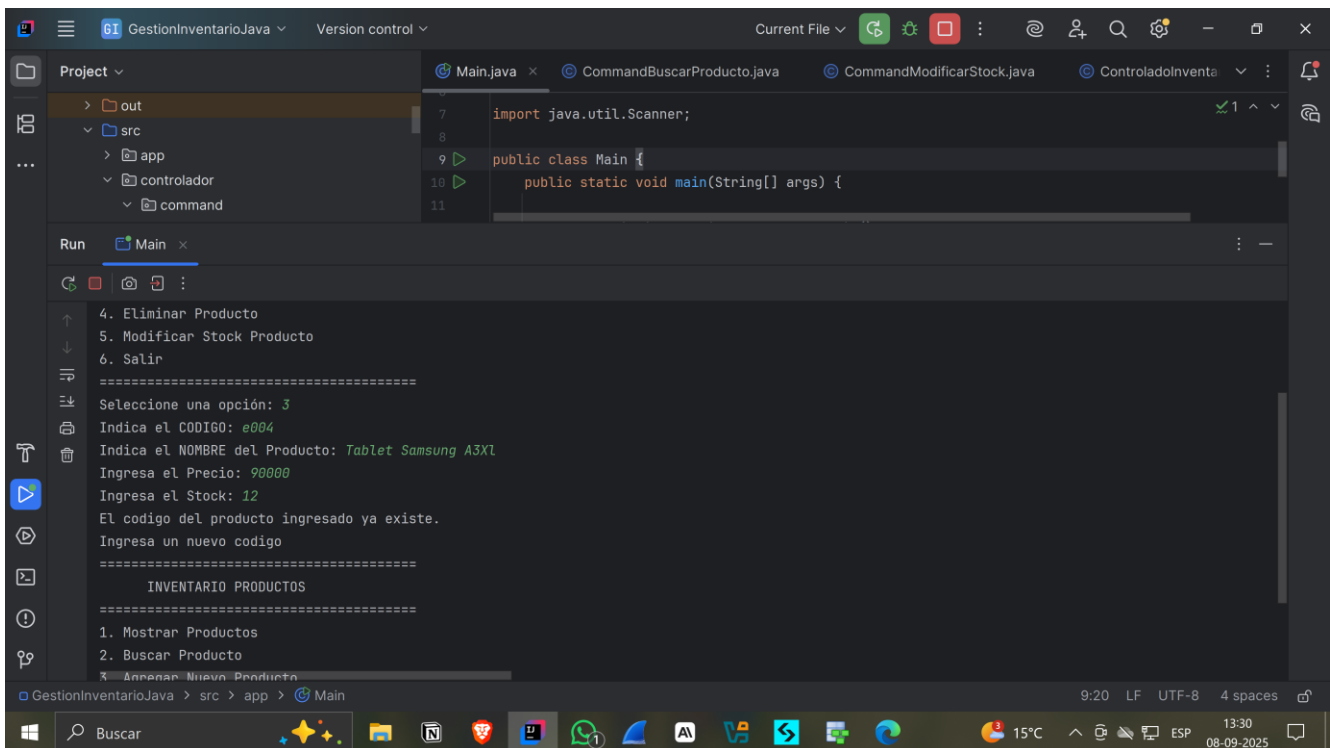
Nombre Producto: SmartPhone Apple 15S
Codigo Producto: E005
Precio producto: $200000.0
Stock: 5

Nombre Producto: SmartPhone Samsung A32
Codigo Producto: E001
Precio producto: $100000.0
Stock: 5

Nombre Producto: SmartPhone Xiaomi X10 Mini
Codigo Producto: E002
Precio producto: $125000.0
Stock: 5

Nombre Producto: SmartPhone Huawei HW6
Codigo Producto: E003
Precio producto: $150000.0
Stock: 5
```

Verifica la existencia de un código existente:



```
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir

=====
Seleccione una opción: 3
Indica el CODIGO: e004
Indica el NOMBRE del Producto: Tablet Samsung A3X1
Ingresa el Precio: 90000
Ingresa el Stock: 12
El código del producto ingresado ya existe.
Ingresa un nuevo código

=====
INVENTARIO PRODUCTOS
=====

1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
```

Stock superior a 0:

```
6
7 import java.util.Scanner;
8
9 public class Main {
10     public static void main(String[] args) {
11
```

Run Main x

```
=====
↑
Seleccione una opción: 5
↓
Indica el CODIGO: e001
Ingresa el Stock: -2
El Stock no puedo ser menor a 0
=====
INVENTARIO PRODUCTOS
=====
1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir
=====
Seleccione una opción:
```

GestionInventarioJava > src > app > Main 9:20 LF UTF-8 4 spaces

Eliminar producto Código invalido:

```
1 package app;
2
3 import controlador.ControladoInventario;
4 import modelo.Inventario;
5 import vista.MenuPrincipal;
6
7 import java.util.Scanner;
8
9 public class Main {
```

Run Main x

```
=====
1. Mostrar Productos
2. Buscar Producto
3. Agregar Nuevo Producto
4. Eliminar Producto
5. Modificar Stock Producto
6. Salir
=====
Seleccione una opción: 4
Indica el CODIGO: R001
Código invalido
=====
INVENTARIO PRODUCTOS
=====
```

GestionInventarioJava > src > app > Main 9:20 LF UTF-8 4 spaces

Eliminacion Producto:

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project structure with folders `out`, `src`, and `app`.
- Editor:** Displays `Main.java` with the following code:

```
import java.util.Scanner;  
  
public class Main {
```
- Run Console:** Shows the output of the program:

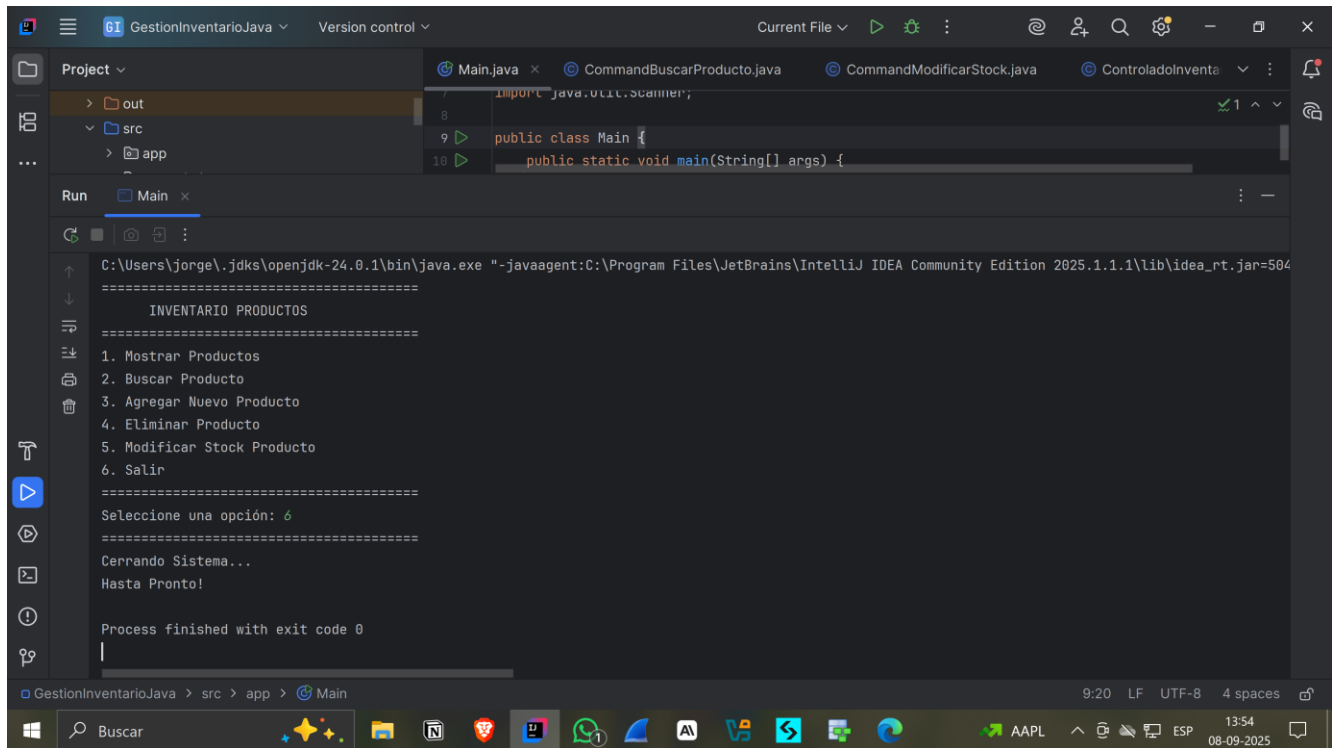
```
Nombre Producto: SmartPhone Honor H32  
Codigo Producto: E004  
Precio producto: $95000.0  
Stock: 5  
=====  
INVENTARIO PRODUCTOS  
=====  
1. Mostrar Productos  
2. Buscar Producto  
3. Agregar Nuevo Producto  
4. Eliminar Producto  
5. Modificar Stock Producto  
6. Salir  
=====  
Seleccione una opción: 4  
Indica el CODIGO: e004  
Producto codigo: E004 ha sido eliminado de forma correcta  
=====
```
- Taskbar:** Shows the Windows taskbar with various application icons and system information (16°C, 13:49, 08-09-2025).

The screenshot shows the same IDE with the following components:

- Project Explorer:** Same project structure as the first screenshot.
- Editor:** Displays `Main.java` with the same code as the first screenshot.
- Run Console:** Shows the output of the program:

```
Nombre Producto: SmartPhone Apple 15S  
Codigo Producto: E005  
Precio producto: $200000.0  
Stock: 5  
  
Nombre Producto: SmartPhone Samsung A32  
Codigo Producto: E001  
Precio producto: $100000.0  
Stock: 5  
  
Nombre Producto: SmartPhone Xiaomi X10 Mini  
Codigo Producto: E002  
Precio producto: $125000.0  
Stock: 5  
  
Nombre Producto: SmartPhone Huawei HW6  
Codigo Producto: E003  
Precio producto: $150000.0  
Stock: 5  
=====  
INVENTARIO PRODUCTOS  
=====
```
- Taskbar:** Same Windows taskbar as the first screenshot.

Cierre Sistema:



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.