

Semana 5

Desarrollo Orientado a Objetos II (PRY2203)

Formato de respuesta

| **Nombre estudiante: Sebastián Tapia** | |
| --- | --- |
| **Asignatura:** | **Carrera:** |
| **Profesor:** | **Fecha:** |

# Descripción de la actividad

## En esta quinta semana realizarás una actividad sumativa individual por encargo llamada "Utilizando herramientas de trabajo colaborativo", donde deberás implementar en tu proyecto, contenidos de ambiente del ciclo de vida, integración y calidad de software, además del uso de herramientas virtuales de trabajo colaborativas.

## Instrucciones específicas

Para realizar la actividad sumativa de esta semana, primero deberás presentar el proyecto realizado en la semana 4, aplicando las mejoras aportadas por tu docente. Adjunta tus evidencias de la siguiente manera:

1. **Fase de análisis**

|  |  |
| --- | --- |
| **Requisitos funcionales** | **Requisitos no funcionales** |
| Gestión de productos: La aplicación debe permitir a los usuarios agregar nuevos productos, actualizar información de productos existentes y realizar búsquedas en el inventario. | Usabilidad: La interfaz de usuario debe ser simple y basada en la consola para una fácil interacción. |
| Generación de informes: La aplicación debe tener la capacidad de generar informes (se recomienda el uso de colecciones). | Rendimiento: El sistema debe ser eficiente en la gestión de operaciones de inventario, como agregar y buscar productos, para proporcionar respuestas rápidas. |
| Interacción con el usuario: Se debe implementar una interfaz de usuario simple basada en la consola que permita al usuario ejecutar acciones como agregar, eliminar y buscar productos, y listar el inventario. | Seguridad: El sistema debe manejar los datos del inventario y los productos de manera segura. |
| Gestión de inventario: La aplicación debe ser capaz de gestionar una colección de productos e incluir métodos para agregar, eliminar y buscar productos. | Mantenibilidad: La estructura del código debe ser clara y modular para permitir futuras actualizaciones o correcciones con facilidad. |
| Búsqueda de productos: La aplicación debe permitir la búsqueda de productos por código y por nombre o descripción. | Integración: El sistema debe estar diseñado para permitir la correcta integración entre la clase 'Producto' y la clase 'Inventario'. |

1. **Fase de diseño**

**2.1 Clase ‘Producto’**

public class Producto {

private String codigo;

private String nombre;

private double precio;

private String descripcion;

private int cantidadEnStock;

// Constructor

public Producto(String codigo, String nombre, double precio, String descripcion, int cantidadEnStock) {

this.codigo = codigo;

this.nombre = nombre;

this.precio = precio;

this.descripcion = descripcion;

this.cantidadEnStock = cantidadEnStock;

}

// Getters

public String getCodigo() {

return codigo;

}

public String getNombre() {

return nombre;

}

public double getPrecio() {

return precio;

}

public String getDescripcion() {

return descripcion;

}

public int getCantidadEnStock() {

return cantidadEnStock;

}

// Setters

public void setCodigo(String codigo) {

this.codigo = codigo;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public void setPrecio(double precio) {

this.precio = precio;

}

public void setDescripcion(String descripcion) {

this.descripcion = descripcion;

}

public void setCantidadEnStock(int cantidadEnStock) {

this.cantidadEnStock = cantidadEnStock;

}

// Método para actualizar el precio del producto

public void actualizarPrecio(double nuevoPrecio) {

this.precio = nuevoPrecio;

}

// Método que devuelve una descripción detallada del producto

@Override

public String toString() {

return "Código: " + codigo +

", Nombre: " + nombre +

", Descripción: " + descripcion +

", Precio: $" + precio +

", Cantidad en stock: " + cantidadEnStock;

}

}

**2.2 Clase ‘Inventario’**

package com.sebastiantapia.ambiente.modelos;

import java.util.HashMap;

import java.util.List;

import java.util.stream.Collectors;

public class Inventario {

private HashMap<String, Producto> productos;

public Inventario() {

productos = new HashMap<>();

}

// Método para agregar productos al inventario

public void agregarProducto(Producto producto) {

if (producto != null && producto.getCodigo() != null) {

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

System.out.println("Producto '" + producto.getNombre() + "' agregado al inventario.");

} else {

System.out.println("Error: No se puede agregar un producto nulo o sin código.");

}

}

// Método para eliminar productos por su código

public void eliminarProducto(String codigo) {

if (productos.remove(codigo) != null) {

System.out.println("Producto con código '" + codigo + "' eliminado del inventario.");

} else {

System.out.println("Error: Producto con código '" + codigo + "' no encontrado.");

}

}

// Método para buscar productos por nombre

public List<Producto> buscarPorNombre(String nombre) {

return productos.values().stream()

.filter(p -> p.getNombre().equalsIgnoreCase(nombre))

.collect(Collectors.toList());

}

// Método para buscar productos por código (retorna un solo producto)

public Producto buscarPorCodigo(String codigo) {

return productos.get(codigo);

}

**'**

// Método para listar todos los productos

public List<Producto> listarTodosLosProductos() {

return productos.values().stream().collect(Collectors.toList());

}

// Método para generar un informe (ejemplo simple)

public void generarInforme() {

System.out.println("\n--- Informe de Inventario ---");

if (productos.isEmpty()) {

System.out.println("El inventario está vacío.");

} else {

for (Producto p : productos.values()) {

System.out.println(p.toString());

}

}

System.out.println("-----------------------------");

}

}

1. **Implementación**

**Interacción con el usuario - Gestión de productos - Búsqueda y listado de productos**

Clase ‘MenuPrincipal’

package com.sebastiantapia.ambiente.modelos;

import java.util.Scanner;

import java.util.List;

import java.util.InputMismatchException;

public class MenuPrincipal {

private Inventario inventario;

private Scanner scanner;

public MenuPrincipal(Inventario inventario, Scanner scanner) {

this.inventario = inventario;

this.scanner = scanner;

}

public void iniciar() {

int opcion = -1;

while (opcion != 0) {

mostrarMenu();

try {

opcion = scanner.nextInt();

scanner.nextLine(); // Consumir el salto de línea

ejecutarOpcion(opcion);

} catch (InputMismatchException e) {

System.out.println("Entrada no válida. Por favor, ingrese un número.");

scanner.nextLine(); // Limpiar el buffer de entrada

opcion = -1;

}

}

}

private void mostrarMenu() {

System.out.println("\n--- Menú de Gestión de Inventario ---");

System.out.println("1. Agregar un nuevo producto");

System.out.println("2. Eliminar un producto");

System.out.println("3. Buscar producto por nombre");

System.out.println("4. Buscar producto por código");

System.out.println("5. Listar todos los productos");

System.out.println("6. Generar informe de inventario");

System.out.println("0. Salir");

System.out.print("Seleccione una opción: ");

}

private void ejecutarOpcion(int opcion) {

switch (opcion) {

case 1:

agregarProducto();

break;

case 2:

eliminarProducto();

break;

case 3:

buscarPorNombre();

break;

case 4:

buscarPorCodigo();

break;

case 5:

listarProductos();

break;

case 6:

inventario.generarInforme();

break;

case 0:

System.out.println("Saliendo del sistema. ¡Hasta pronto!");

break;

default:

System.out.println("Opción no válida. Por favor, intente de nuevo.");

break;

}

}

private void agregarProducto() {

System.out.println("\n--- Agregar Nuevo Producto ---");

System.out.print("Ingrese el código del producto: ");

String codigo = scanner.nextLine();

System.out.print("Ingrese el nombre del producto: ");

String nombre = scanner.nextLine();

System.out.print("Ingrese la descripción del producto: ");

String descripcion = scanner.nextLine();

System.out.print("Ingrese el precio del producto: ");

double precio = scanner.nextDouble();

System.out.print("Ingrese la cantidad en stock: ");

int cantidad = scanner.nextInt();

scanner.nextLine(); // Consumir el salto de línea

Producto nuevoProducto = new Producto(codigo, nombre, precio, descripcion, cantidad);

inventario.agregarProducto(nuevoProducto);

}

private void eliminarProducto() {

System.out.println("\n--- Eliminar Producto ---");

System.out.print("Ingrese el código del producto a eliminar: ");

String codigo = scanner.nextLine();

inventario.eliminarProducto(codigo);

}

private void buscarPorNombre() {

System.out.println("\n--- Buscar Producto por Nombre ---");

System.out.print("Ingrese el nombre del producto a buscar: ");

String nombre = scanner.nextLine();

List<Producto> resultados = inventario.buscarPorNombre(nombre);

if (resultados.isEmpty()) {

System.out.println("No se encontraron productos con ese nombre.");

} else {

System.out.println("Productos encontrados:");

for (Producto p : resultados) {

System.out.println(p.toString());

}

}

}

private void buscarPorCodigo() {

System.out.println("\n--- Buscar Producto por Código ---");

System.out.print("Ingrese el código del producto a buscar: ");

String codigo = scanner.nextLine();

Producto producto = inventario.buscarPorCodigo(codigo);

if (producto != null) {

System.out.println("Producto encontrado:");

System.out.println(producto.toString());

} else {

System.out.println("No se encontró un producto con ese código.");

}

}

private void listarProductos() {

System.out.println("\n--- Lista de Todos los Productos ---");

List<Producto> todos = inventario.listarTodosLosProductos();

if (todos.isEmpty()) {

System.out.println("El inventario está vacío.");

} else {

for (Producto p : todos) {

System.out.println(p.toString());

}

}

}

}

1. **Fase de pruebas**

**Pruebas de integración y Pruebas unitarias**

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Una vez que adjuntes tus evidencias, deberás realizar los siguientes pasos correspondientes al uso de herramientas de trabajo colaborativo Miro y Trello:

**Paso 1: realiza un Roadmap con la herramienta Miro**

Utiliza Miro para crear el Roadmap del proyecto, ingresando al siguiente enlace:

[www.miro.com](http://www.miro.com/)

**Figura 1**

*Boards de Miro*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Selecciona el Technology Roadmap

En la construcción del Roadmap, deberás realizar las siguientes acciones:

* 1. **Definir fases del proyecto:**

Utiliza las "sticky notes" (notas adhesivas) o los "frames" (marcos) para definir cada fase del ciclo de vida del desarrollo del software: fase de análisis, fase de Diseño, fase de implementación y de pruebas.

**1.2 Desglosar tareas:**

Dentro de cada fase, desglosa las tareas específicas que necesitas completar. Por ejemplo, bajo la fase de diseño, definir tareas como "Definir clases y atributos" y "Diseñar la base de datos".

**Importante**

Si tienes dificultades con Miro y la creación del Roadmap, revisa el siguiente video:

<https://www.youtube.com/watch?v=W8M6f60kGiY&ab_channel=Miro>

**Paso 2: Implementa la herramienta colaborativa Trello**

Para incorporar Trello, debes tener tu cuenta creada en la página:

<https://trello.com/>

Luego, deberás crear un nuevo tablero seleccionando el botón “Crear” y luego “Crear tablero”:

**Figura 2**

*Menú principal de Trello*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Nombra a tu tablero “Gestión de inventario en Java” y estructúralo de la siguiente forma:

**2.1 Crea listas para cada fase del proyecto:**

Organiza el tablero creando listas que representen cada fase del ciclo de vida del desarrollo de software: "Fase de Análisis", "Fase de Diseño", "Fase de Implementación" y "Fase de Pruebas". También deberás crear listas HACIENDO y HECHO.

**2.2 Agregar tarjetas para tareas específicas:**

Dentro de cada lista, crea tarjetas para tareas específicas. Por ejemplo, en la lista "Fase de Diseño", puedes tener tarjetas como "Definir clases y atributos" o "Diseñar la base de datos".

**2.3 Detalla las tarjetas:**

Añade descripciones a cada tarjeta para explicar la tarea con más detalle. Utiliza las funciones de "checklist" para dividir las tareas en sub-tareas más pequeñas y asigna fechas límite.

Para que tu docente pueda revisar tu entrega, deberás compartir el acceso desde la respectiva herramienta y adjuntar los enlaces en esta parte:

**Miro**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Enlace Miro:**

<https://miro.com/app/board/uXjVJKNnHyw=/?share_link_id=163248217848>

**Trello**

**Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente**

**Enlace Trello:**

<https://trello.com/invite/b/68c08f3f598c042cdc33750b/ATTI8abdaaece4eb0309a3a301fd98bc3ba7AF947A89/gestion-de-inventario-en-java>

**Paso 3:** Cuando termines tu entrega, debes subir este mismo archivo al AVA para que tu docente realice la evaluación y retroalimentación.



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.