Mini Sistema de Administración de Productos y Localidades

Contexto:

Se debe desarrollar un sistema de administración para gestionar productos y sus localidades en un almacén. Los productos pueden estar ubicados en distintas localidades dentro del almacén, y cada localidad almacena una cierta cantidad del producto. Se espera que se implemente la solución utilizando programación orientada a objetos en Python, haciendo uso de excepciones personalizadas para un manejo adecuado de errores. Adjunto, encontrará el archivo "*test.py*" con *tests* unitarios para la funcionalidad básica de las clases, además de un diagrama visual.

El sistema tendrá tres clases principales:

- **Producto**: Representa un producto en el almacén.
- **Localidad**: Representa una ubicación física donde se almacena un producto.
- **Administracion**: Responsable de gestionar los productos y sus localidades.

Además, se deberán definir y utilizar excepciones personalizadas para manejar las siguientes situaciones:

- **ProductoNoEncontradoError**: Se lanza cuando no se encuentra un producto por su id.
- **LocalidadDuplicadaError**: Se lanza cuando se intenta agregar una localidad que ya está asignada a un producto.
- **LocalidadNoEncontradaError**: Se lanza cuando no se encuentra una localidad específica para un producto.

Detalles de las Clases y Funcionalidad Esperada:

Clase: Localidad

Descripción: Representa una ubicación física donde un producto puede estar almacenado. Tiene cinco atributos para describir la ubicación y un sexto atributo que es la cantidad de stock.

Atributos:

- **deposito** (str): El nombre o identificador del depósito.
- **isla** (str): La identificación de la isla dentro del depósito.
- **modulo** (int): El módulo dentro de la isla.
- **lado** (str): El lado del módulo.
- **estante** (int): El estante dentro del módulo.
- **cantidad** (int): Cantidad de stock en esta localidad (valor predeterminado 0).

Define el método de igualdad que compara si dos instancias de Localidad son iguales en base a sus cinco atributos (deposito, isla, modulo, lado, estante).

Clase: Producto

Descripción: Representa un producto almacenado en múltiples localidades. Cada producto tiene un ID único, una descripción y una lista de localidades donde está almacenado.

Atributos:

- **id** (int): Identificador autoincremental único para cada producto.
- **descripcion** (str): Descripción del producto.
- **localidades** (list): Lista de instancias de Localidad donde el producto está almacenado. Las localidades en esta lista deben ser distintas entre sí.

Métodos:

- **stock**: Calcula y devuelve el stock total del producto sumando la cantidad de todas las localidades donde está presente.
- **poseeLocalidad**: Devuelve True si el producto está almacenado en la localidad pasada como parámetro; de lo contrario, devuelve False.

Clase: Administracion

Descripción: Gestiona la colección de productos y sus localidades. Proporciona métodos para agregar productos, localidades, cargar stock y consultar el stock de un producto específico.

Atributos:

- **productos** (dict): Un diccionario que mapea el ID de cada producto a su correspondiente instancia de Producto.

Métodos:

- **agregarProducto**: Crea una instancia de Producto con la descripción proporcionada y la agrega a la colección de productos.
- **agregarLocalidad**: Toma un ID de producto y los datos para generar una instancia de Localidad. Agrega esa localidad al producto si no existe ya en su lista de localidades. Si el producto no existe, lanza una excepción de tipo **ProductoNoEncontradoError**. Si la localidad ya existe en el producto, lanza una excepción de tipo **LocalidadDuplicadaError**.
- **cargarStock**: Toma un ID de producto, la cantidad y los datos de una Localidad. Aumenta el stock de un producto en una localidad específica. Si el producto no existe, lanza una excepción de tipo **ProductoNoEncontradoError**. Si la localidad no existe en el producto, lanza una excepción de tipo **LocalidadNoEncontradaError**.
- **stock**: Consulta el stock total del producto con el ID proporcionado. Si el producto no existe, lanza una excepción de tipo **ProductoNoEncontradoError**.