Edison García-202200031

MANUAL TÉCNICO: Sistema de Inventario

El siguiente manual técnico describe el funcionamiento y las características del sistema de inventario implementado en el lenguaje de programación Python. El sistema se encarga de gestionar un inventario de productos, realizar operaciones como agregar stock y vender productos, y generar informes de inventario.

A. Clases Principales

1. Clase "Producto"

La clase "Producto" representa un artículo en el inventario con los siguientes atributos:

- "nombre": El nombre del producto.
- "cantidad": La cantidad disponible del producto en el inventario.
- "precio_unitario": El precio unitario del producto.
- "ubicación": La ubicación del producto en el inventario.

1.1 Métodos:

```python
def __init__(self, nombre, cantidad, precio_unitario, ubicacion):
    self.nombre = nombre
    self.cantidad = cantidad
    self.precio_unitario = precio_unitario
    self.ubicacion = ubicacion
```

Constructor de la clase, inicializa los atributos del producto.

```python
def actualizar_cantidad(self, cantidad):
    self.cantidad += cantidad
```

Actualiza la cantidad del producto en el inventario sumándole la cantidad especificada.

```python
def vender(self, cantidad):
    if cantidad <= self.cantidad:
        self.cantidad -= cantidad
        return True
    else:
        return False
```

Vende una cantidad especificada del producto. Si la cantidad es suficiente, resta la cantidad vendida; de lo contrario, devuelve "False".

2. Clase “Inventario”

La clase “Inventario” maneja la colección de productos y las operaciones en el inventario. Contiene los siguientes métodos:

```python
def __init__(self):
    self.productos = {}
```

Constructor de la clase, inicializa el diccionario de productos.

```python
def cargar_inventario_inicial(self, archivo):
    with open(archivo, "r", encoding="utf-8") as f:
        for linea in f:
            instruccion, detalles = linea.strip().split(' ', 1)
            if instruccion == 'crear_producto':
                nombre, cantidad, precio, ubicacion = detalles.split(';')
                producto = Producto(nombre, int(cantidad), float(precio), ubicacion)
                self.productos[nombre + '_' + ubicacion] = producto
```

Carga el inventario inicial desde un archivo. Lee las líneas del archivo y crea objetos “Producto” correspondientes.

```python
def cargar_instrucciones_movimientos(self, archivo):
    with open(archivo, "r", encoding="utf-8") as f:
        for linea in f:
            instruccion, detalles = linea.strip().split(' ', 1)
            if instruccion == 'agregar_stock':
                nombre, cantidad, ubicacion = detalles.split(';')
                key = nombre + '_' + ubicacion
                if key in self.productos:
                    self.productos[key].actualizar_cantidad(int(cantidad))
                else:
                    print(f"Error: Producto '{nombre}' no exite en la ubicación '{ubicacion}'.")
            elif instruccion == 'vender_producto':
                nombre, cantidad, ubicacion = detalles.split(';')
                key = nombre + '_' + ubicacion
                if key in self.productos:
                    if self.productos[key].vender(int(cantidad)):
                        print(f"Venta de {cantidad} unidades de '{nombre}' en '{ubicacion}' realizada.")
                    else:
                        print(f"Error: Cantidad insuficiente de '{nombre}' en '{ubicacion}'.")
                else:
                    print(f"Error: Producto '{nombre}' no existe en la ubicación '{ubicacion}'.")
```

Carga las instrucciones de movimientos desde un archivo. Ejecuta operaciones como agregar stock o vender productos según las instrucciones.

```python
def ordenar_por_ubicacion(self):
    sorted_products = sorted(self.productos.values(), key=lambda producto: producto.ubicacion)
    return sorted_products
```

Ordena los productos en el inventario según su ubicación y devuelve una lista ordenada.

Edison García-202200031

```python
def crear_informe_inventario(self, archivo):
    productos_ordenados = self.ordenar_por_ubicacion()
    with open(archivo, 'w', encoding="UTF-8") as f:
        f.write("Informe de Inventario:\n")
        f.write("Ordenado por Ubicación\n")
        f.write("{:<15} {:<10} {:<15} {:<15} {:<10}\n".format("Producto", "Cantidad", "Precio Unitario", "Valor Total", "Ubicación"))
        f.write("-" * 70 + "\n")

        for producto in productos_ordenados:
            valor_total = producto.cantidad * producto.precio_unitario
            f.write("{:<15} {:<10} ${:<15} ${:<15} {:<10}\n".format(producto.nombre, producto.cantidad, producto.precio_unitario, valor_t
```

Crea un informe de inventario ordenado por ubicación en un archivo. Muestra información detallada de los productos y sus cantidades, precios y valores totales.

B. Función “main()”

La función “main()” inicia la ejecución del programa y presenta un menú interactivo al usuario. El menú permite al usuario realizar las siguientes acciones:

1. **Cargar Inventario Inicial:** Carga información sobre los productos del inventario desde un archivo.
2. **Cargar Instrucciones de Movimientos:** Carga instrucciones desde un archivo para agregar stock o vender productos.
3. **Crear Informe de Inventario:** Genera un informe de inventario ordenado por ubicación y lo guarda en un archivo.
4. **Salir:** Termina la ejecución del programa.

USO DEL PROGRAMA

1. **Ejecución:** Ejecuta el programa en un entorno de Python.
2. **Menú:** Selecciona una opción del menú principal (1, 2, 3 o 4).
3. Según la opción seleccionada:
   - **Opción 1:** Ingresa el nombre del archivo (.inv) que contiene información inicial de productos en el inventario.
   - **Opción 2:** Ingresa el nombre del archivo (.mov) que contiene instrucciones para agregar stock o vender productos.
   - **Opción 3:** Ingresa el nombre del archivo (.txt) en el cual se guardará el informe de inventario.
   - **Opción 4:** Finaliza el programa.
4. Las opciones 1 y 2 procesarán la información de los archivos y ejecutarán las operaciones correspondientes.

5. La opción 3 generará un informe de inventario en el archivo especificado.

6. El programa puede ser finalizado seleccionando la opción 4.

CONSIDERACIONES:

- Se asume que los archivos de entrada (.inv, .mov) y el archivo de informe (.txt) están ubicados en el mismo directorio que el script, es decir, en la misma carpeta donde se encuentra el archivo que contiene al código fuente.

- El código maneja situaciones de errores como productos inexistentes, cantidades insuficientes, nombres de archivo incorrectos, entre otros. Y muestra mensajes adecuados al usuario.

Este manual técnico proporciona una descripción detallada del sistema de inventario implementado en el código. Proporciona instrucciones sobre cómo interactuar con el programa y realizar diversas operaciones relacionadas con la gestión de inventario y generación de informes.