Sistema de Gerenciamento de Pedidos Online

Arthur Mazzardo Naue - SI 2024

Descrição:

Desenvolva um sistema de gerenciamento de pedidos online que permita a administração de clientes, produtos e pedidos. O sistema deve ser implementado em Python.

Requisitos Funcionais:

Clientes:

- Cada cliente deve ter um nome, um endereço e um e-mail.
- Os clientes podem ser adicionados, removidos, atualizados e consultados no sistema.

Produtos:

- Cada produto deve ter um nome, uma descrição, um preço e uma quantidade em estoque.
- Os produtos podem ser adicionados, removidos, atualizados e consultados no sistema.

Pedidos:

- Cada pedido deve ter um cliente associado, uma lista de produtos solicitados e um status (por exemplo, "pendente", "entregue", "cancelado").
- Os pedidos podem ser criados, atualizados (por exemplo, adicionar/remover produtos), consultados e marcados como entregues ou cancelados.

Operações:

- Implemente operações para adicionar, remover, atualizar e consultar clientes, produtos e pedidos no sistema.

Clientes:

- Cada cliente deve ter um nome, um endereço e um e-mail.
- Os clientes podem ser adicionados, removidos, atualizados e consultados no sistema.

```python
from cObjetoListavel import Objeto_Listavel

class Cliente(Objeto_Listavel):
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, sNome: str, sEndereco: str, sEmail: str) -> None:
        self.__sNome = sNome
        self.__sEndereco = sEndereco
        self.__sEmail = sEmail

    Tabnine | Edit | Test | Explain | Document
    def __str__(self) -> str:
        return self.__sNome
```

Clientes:

- Cada cliente deve ter um nome, um endereço e um e-mail.
- Os clientes podem ser adicionados, removidos, atualizados e consultados no sistema.

```
sTipoOperacao = input("O QUE VOCE GOSTARIA DE FAZER? (MOSTRAR, CONSULTAR, CRIAR, ATUALIZAR OU REMOVER)\n")
match sTipoOperacao:

    case "mostrar":
        match eTipoObjeto:
            case Objetos.CLIENTE:
                for cliente in lListaClientes:
                    print(cliente)
                main()
            case Objetos.PRODUTO:
                iInputFiltro = input("ADICIONAR FILTRO DE ESTOQUE? (SIM OU NAO)\n")
                match iInputFiltro:
                    case "sim":
                        for produto in lListaProdutos:
                            if produto.quantidade > 0:
                                print(produto)
                    case "nao":
                        for produto in lListaProdutos:
                            print(produto)
                main()
```

```
case "consultar":
    match eTipoObjeto:
        case Objetos.CLIENTE:
            bClienteEncontrado = False
            iNomeObjeto = input("QUAL O NOME DO CLIENTE?\n")
            for cliente in lListaClientes:
                if cliente.nome == iNomeObjeto:
                    print(cliente.info)
                    bClienteEncontrado = True
            if bClienteEncontrado == False:
                print("CLIENTE NAO ENCONTRADO\n")
            main()
```

```
case "criar":
    match eTipoObjeto:
        case Objetos.CLIENTE:
            lListaClientes.append(Cliente(input("NOME DO CLIENTE\n"), input("ENDERECO DO CLIENTE\n"), input("EMAIL DO CLIENTE\n")))
            print("CRIADO COM SUCESSO!\n")
            main()
```

```
case "atualizar":
    match eTipoObjeto:
        case Objetos.CLIENTE:
            cCliente: Cliente
            bClienteAchado = False
            iNomeCliente = input("QUAL O NOME DO CLIENTE A SER ATUALIZADO?\n")
            for cliente in lListaClientes:
                if iNomeCliente == cliente.nome:
                    cCliente = cliente
                    bClienteAchado = True

            if bClienteAchado:
                iAtributoAtualizado = input("QUAL ATRIBUTO VOCE GOSTARIA DE ATUALIZAR? (NOME, ENDERECO, EMAIL)\n")
                match iAtributoAtualizado:
                    case "nome":
                        iNovoAtributo = input("QUAL O NOVO NOME?\n")
                        cCliente.set_nome = iNovoAtributo
                        print("CLIENTE ATUALIZADO COM SUCESSO\n")
                    case "endereco":
                        iNovoAtributo = input("QUAL O NOVO ENDERECO?\n")
                        cCliente.set_endereco = iNovoAtributo
                        print("CLIENTE ATUALIZADO COM SUCESSO\n")
                    case "email":
                        iNovoAtributo = input("QUAL O NOVO EMAIL\n")
                        cCliente.set_email = iNovoAtributo
                        print("CLIENTE ATUALIZADO COM SUCESSO\n")
                main()
            else:
                print("CLIENTE NAO ENCONTRADO\n")
                main()
            main()
```

```
case "remover":
    match eTipoObjeto:
        case Objetos.CLIENTE:
            cCliente: Cliente
            bClienteEncontrado = False
            iClienteDesejado = input("QUAL O NOME DO CLIENTE A SER REMOVIDO?\n")

            for cliente in lListaClientes:
                if cliente.nome == iClienteDesejado:
                    cCliente = cliente
                    bClienteEncontrado = True

            if bClienteEncontrado:
                lListaClientes.remove(cCliente)
                print("CLIENTE REMOVIDO\n")
                main()
            else:
                print("CLIENTE NAO ENCONTRADO\n")
                main()
```

Produtos:

- Cada produto deve ter um nome, uma descrição, um preço e uma quantidade em estoque.
- Os produtos podem ser adicionados, removidos, atualizados e consultados no sistema.

```python
from cObjetoListavel import Objeto_Listavel

class Produto(Objeto_Listavel):
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, sNome: str, sDescricao: str, fPreco: float, iQuantidade: int) -> None:
        self.__sNome = sNome
        self.__sDescricao = sDescricao
        self.__fPreco = fPreco
        self.__iQuantidade = iQuantidade
```

Produtos:

- Cada produto deve ter um nome, uma descrição, um preço e uma quantidade em estoque.

- Os produtos podem ser adicionados, removidos, atualizados e consultados no sistema.

```
case Objetos.PRODUTO:
    iInputFiltro = input("ADICIONAR FILTRO DE ESTOQUE? (SIM OU NAO)\n")
    match iInputFiltro:
        case "sim":
            for produto in lListaProdutos:
                if produto.quantidade > 0:
                    print(produto)
        case "nao":
            for produto in lListaProdutos:
                print(produto)
    main()
```

MOSTRAR

```
case Objetos.PRODUTO:
    bProdutoEncontrado = False
    iNomeObjeto = input("QUAL O NOME DO PRODUTO?\n")
    for produto in lListaProdutos:
        if produto.nome == iNomeObjeto:
            print(produto.info)
            bProdutoEncontrado = True
    if bProdutoEncontrado == False:
        print("PRODUTO NAO ENCONTRADO")
    main()
```

CONSULTAR

```
case Objetos.PRODUTO:
    lListaProdutos.append(Produto(input("NOME DO PRODUTO\n"), input("DESCRICAO DO PRODUTO\n"), float(input("PRECO DO PRODUTO\n")), int(input("QUANTIDADE DO PRODUTO\n"))
    print("CRIADO COM SUCESSO!\n")
    main()
```

CRIAR

ATUALIZAR

```
case Objetos.PRODUTO:
    cProduto: Produto
    bProdutoAchado = False
    iNomeProduto = input("QUAL O NOME DO PRODUTO A SER ATUALIZADO?\n")
    for produto in lListaProdutos:
        if iNomeProduto == produto.nome:
            cProduto = produto
            bProdutoAchado = True

    if bProdutoAchado:
        iAtributoAtualizado = input("QUAL ATRIBUTO VOCE GOSTARIA DE ATUALIZAR? (NOME, DESCRICAO, PRECO, QUANTIDADE)\n")
        match iAtributoAtualizado:
            case "nome":
                iNovoAtributo = input("QUAL O NOVO NOME?\n")
                cProduto.set_nome = iNovoAtributo
                print("PRODUTO ALTERADO COM SUCESSO\n")
            case "descricao":
                iNovoAtributo = input("QUAL A NOVA DESCRICAO?\n")
                cProduto.set_descricao = iNovoAtributo
                print("PRODUTO ALTERADO COM SUCESSO\n")
            case "preco":
                iNovoAtributo = input("QUAL O NOVO PRECO?\n")
                cProduto.set_preco = float(iNovoAtributo)
                print("PRODUTO ALTERADO COM SUCESSO\n")
            case "quantidade":
                iNovoAtributo = input("QUAL A NOVA QUANTIDADE?\n")
                cProduto.set_quantidade = int(iNovoAtributo)
                print("PRODUTO ALTERADO COM SUCESSO\n")
        main()
    else:
        print("PRODUTO NAO ENCONTRADO\n")
        main()
```

REMOVER

```
case Objetos.PRODUTO:
    cProduto: Produto
    bProdutoEncontrado = False
    iProdutoDesejado = input("QUAL O NOME DO PRODUTO A SER REMOVIDO?\n")

    for produto in lListaProdutos:
        if produto.nome == iProdutoDesejado:
            cProduto = produto
            bProdutoEncontrado = True

    if bProdutoEncontrado:
        lListaProdutos.remove(cProduto)
        print("PRODUTO REMOVIDO\n")
        main()
    else:
        print("PRODUTO NAO ENCONTRADO\n")
        main()
```

Pedidos:

- Cada pedido deve ter um cliente associado, uma lista de produtos solicitados e um status (por exemplo, "pendente", "entregue", "cancelado").
- Os pedidos podem ser criados, atualizados (por exemplo, adicionar/remover produtos), consultados e marcados como entregues ou cancelados.

```python
from cObjetoListavel import Objeto_Listavel
from enum import Enum
from cCliente import Cliente

class StatusPedido(Enum):
    PENDENTE = 1
    ENTREGUE = 2
    CANCELADO = 3

class Pedido(Objeto_Listavel):
    Tabnine | Edit | Test | Explain | Document
    def __init__(self, sCod: str, cCliente: Cliente, lListaProdutos: list, eStatus: StatusPedido) -> None:
        self.__sCod = sCod
        self.__cCliente = cCliente
        self.__lListaProdutos = lListaProdutos
        self.__eStatus = eStatus

    Tabnine | Edit | Test | Explain | Document
    def __str__(self) -> str:
        return self.__sCod
```

Pedidos:

- Cada pedido deve ter um cliente associado, uma lista de produtos solicitados e um status (por exemplo, "pendente", "entregue", "cancelado").
- Os pedidos podem ser criados, atualizados (por exemplo, adicionar/remover produtos), consultados e marcados como entregues ou cancelados.

MOSTRAR

```
case Objetos.PEDIDO:
    iInputFiltro = input("ADICIONAR FILTRO? (SIM OU NAO)\n")
    match iInputFiltro:
        case "sim":
            iInputTipoFiltro = input("QUAL FILTRO? (CLIENTE OU STATUS)\n")
            match iInputTipoFiltro:
                case "cliente":
                    iInputFiltroCliente = input("QUAL O NOME DO CLIENTE?\n")
                    for pedido in lListaPedidos:
                        if pedido.cliente.nome == iInputFiltroCliente:
                            print(pedido)
                case "status":
                    iInputFiltroStatus = input("QUAL? (PENDENTE, ENTREGUE, CANCELADO)\n")
                    match iInputFiltroStatus:
                        case "pendente":
                            for pedido in lListaPedidos:
                                if pedido.status == StatusPedido.PENDENTE:
                                    print(pedido)
                        case "entregue":
                            for pedido in lListaPedidos:
                                if pedido.status == StatusPedido.ENTREGUE:
                                    print(pedido)
                        case "cancelado":
                            for pedido in lListaPedidos:
                                if pedido.status == StatusPedido.CANCELADO:
                                    print(pedido)
        case "nao":
            for pedido in lListaPedidos:
                print(pedido)
    main()
```

CONSULTAR

```
case Objetos.PEDIDO:
    bPedidoEncontrado = False
    iNomeObjeto = input("QUAL O CODIGO DO PEDIDO?\n")
    for pedido in lListaPedidos:
        if pedido.cod == iNomeObjeto:
            print(pedido.info)
            bPedidoEncontrado = True
    if bPedidoEncontrado == False:
        print("PEDIDO NAO ENCONTRADO")
    main()
```

```
case Objetos.PEDIDO:
    iInputCliente = input("NOME DO CLIENTE\n")
    cCliente = None
    for cliente in lListaClientes:
        if cliente.nome == iInputCliente:
            cCliente = cliente

    if cCliente != None:
        lListaProdutosPedido = []
        bListaPronta = False
        cProdutoEscolhido: str

        while bListaPronta == False:
            cProdutoEscolhido = input("NOME DO PRODUTO\n")
            for produto in lListaProdutos:
                if cProdutoEscolhido == produto.nome:
                    lListaProdutosPedido.append(produto)
            iNewProduct = input("DESEJA ADICIONAR MAIS UM PRODUTO? (SIM OU NAO)\n")
            match iNewProduct:
                case "sim":
                    pass
                case "nao":
                    bListaPronta = True
        else:
            iStatusPedido = input("STATUS DO PEDIDO (PENDENTE, ENTREGUE OU CANCELADO)\n")
            eStatusPedido: StatusPedido
            match iStatusPedido:
                case "pendente":
                    eStatusPedido = StatusPedido.PENDENTE
                case "entregue":
                    eStatusPedido = StatusPedido.ENTREGUE
                case "cancelado":
                    eStatusPedido = StatusPedido.CANCELADO

            lListaPedidos.append(Pedido(input("CODIGO DO PEDIDO\n"), cCliente, lListaProdutosPedido, eStatusPedido))
            print("CRIADO COM SUCESSO!\n")
```

CRIAR

ATUALIZAR

```python
case Objetos.PEDIDO:
    cPedido: Pedido
    bPedidoAchado = False
    iNomePedido = input("QUAL O CODIGO DO PEDIDO A SER ATUALIZADO?\n")
    for pedido in lListaPedidos:
        if iNomePedido == pedido.cod:
            cPedido = pedido
            bPedidoAchado = True

    if bPedidoAchado:
        iAtributoAtualizado = input("QUAL ATRIBUTO VOCE GOSTARIA DE ATUALIZAR? (COD, CLIENTE, LISTA PRODUTOS, STATUS)\n")
        match iAtributoAtualizado:
            case "cod":
                iNovoAtributo = input("QUAL O NOVO COD?\n")
                cPedido.set_cod = iNovoAtributo
                print("PEDIDO ALTERADO COM SUCESSO\n")
            case "cliente":
                iNovoAtributo = input("QUAL O NOME DO NOVO CLIENTE?\n")
                for cliente in lListaClientes:
                    if cliente.nome == iNovoAtributo:
                        iNovoAtributo = cliente
                cPedido.set_client = iNovoAtributo
                print("PEDIDO ALTERADO COM SUCESSO\n")
            case "lista produtos":
                lNovaListaProdutos = []
                bListaProdutosPronta = False
                while bListaProdutosPronta == False:
                    iNovoAtributo = input("QUAL O NOME DO PRODUTO?\n")
                    for produto in lListaProdutos:
                        if produto.nome == iNovoAtributo:
                            lNovaListaProdutos.append(produto)
                    iNovoAtributo = input("ADICIONAR MAIS UM PRODUTO? (SIM OU NAO)\n")
                    match iNovoAtributo:
                        case "sim":
                            pass
                        case "nao":
                            bListaProdutosPronta = True
                cPedido.set_lista_produtos = lNovaListaProdutos
                print("PEDIDO ALTERADO COM SUCESSO\n")
            case "status":
                iNovoAtributo = input("QUAL O NOVO STATUS? (PENDENTE, ENTREGUE, CANCELADO)\n")
                match iNovoAtributo:
                    case "pendente":
                        iNovoAtributo = StatusPedido.PENDENTE
                    case "entregue":
                        iNovoAtributo = StatusPedido.ENTREGUE
                    case "cancelado":
                        iNovoAtributo = StatusPedido.CANCELADO
                cPedido.set_status = iNovoAtributo
                print("PEDIDO ALTERADO COM SUCESSO\n")
main()
```

```
case Objetos.PEDIDO:
    cPedido: Pedido
    bPedidoEncontrado = False
    iPedidoDesejado = input("QUAL O CODIGO DO PEDIDO A SER REMOVIDO?\n")

    for pedido in lListaPedidos:
        if pedido.cod == iPedidoDesejado:
            cPedido = pedido
            bPedidoEncontrado = True

    if bPedidoEncontrado:
        lListaPedidos.remove(cPedido)
        print("PEDIDO REMOVIDO\n")
        main()
    else:
        print("PEDIDO NAO ENCONTRADO\n")
        main()
```

REMOVER

Operações:

- Implemente operações para adicionar, remover, atualizar e consultar clientes, produtos e pedidos no sistema.
- Permita a criação de novos pedidos, associando clientes e produtos.
- Forneça a capacidade de consultar informações sobre pedidos de um cliente específico, produtos em estoque e pedidos com status específico.

Operações:

- Implemente operações para adicionar, remover, atualizar e consultar clientes, produtos e pedidos no sistema.
- Permita a criação de novos pedidos, associando clientes e produtos.
- Forneça a capacidade de consultar informações sobre pedidos de um cliente específico, produtos em estoque e pedidos com status específico.

Operações:

- Implemente operações para adicionar, remover, atualizar e consultar clientes, produtos e pedidos no sistema.
- Permita a criação de novos pedidos, associando clientes e produtos.
- Forneça a capacidade de consultar informações sobre pedidos de um cliente específico, produtos em estoque e pedidos com status específico.

Requisitos Técnicos:

- Utilize princípios de herança, encapsulamento e polimorfismo em seu projeto.
- Utilize coleções (listas, arrays, etc.) para armazenar dados e evite o uso excessivo de variáveis globais.
- Implemente tratamento de exceções para lidar com situações como tentativas de consultar clientes ou produtos inexistentes, adicionar produtos com informações incompletas, etc.
- Crie uma interface de usuário simples para interação com o sistema (pode ser uma interface de linha de comando inicialmente).

```python
class Objeto_Listavel():
    Tabnine | Edit | Test | Explain | Document
    def __init__(self):
        pass

    Tabnine | Edit | Test | Explain | Document
    def info(self):
        return ''
```

```python
class Cliente(Objeto_Listavel):
```

```python
@property
def info(self) -> list:
    return [self.__sNome, self.__sEndereco, self.__sEmail]
```

```python
class Produto(Objeto_Listavel):
```

```python
@property
def info(self) -> list:
    return [self.__sNome, self.__sDescricao, str(self.__fPreco), str(self.__iQuantidade)]
```

```python
class Pedido(Objeto_Listavel):
```

```python
@property
def info(self) -> list:
    return [str(self.__cCliente), str(self.__lListaProdutos), str(self.__eStatus)]
```

Requisitos Técnicos:

- Utilize princípios de herança, encapsulamento e polimorfismo em seu projeto.
- Utilize coleções (listas, arrays, etc.) para armazenar dados e evite o uso excessivo de variáveis globais.
- Implemente tratamento de exceções para lidar com situações como tentativas de consultar clientes ou produtos inexistentes, adicionar produtos com informações incompletas, etc.
- Crie uma interface de usuário simples para interação com o sistema (pode ser uma interface de linha de comando inicialmente).

```
lListaClientes = [
    Cliente("arthur", "Sc 434 s/n", "arthur_naue@hotmail.com"),
    Cliente("vitor", "Rua Jorge Lacerda n32", "vitorxxxpasa@gmail.com")
]
lListaProdutos = [
    Produto("mesa", "Uma mesa para jantar", 49.99, 10),
    Produto("cadeira", "Uma cadeira para sentar", 24.99, 0)
]
lListaPedidos = [
    Pedido("001", lListaClientes[0], lListaProdutos, StatusPedido.PENDENTE),
    Pedido("002", lListaClientes[1], lListaProdutos, StatusPedido.CANCELADO),
    Pedido("003", lListaClientes[0], lListaProdutos, StatusPedido.ENTREGUE)
]
```

Requisitos Técnicos:

- Utilize princípios de herança, encapsulamento e polimorfismo em seu projeto.
- Utilize coleções (listas, arrays, etc.) para armazenar dados e evite o uso excessivo de variáveis globais.
- Implemente tratamento de exceções para lidar com situações como tentativas de consultar clientes ou produtos inexistentes, adicionar produtos com informações incompletas, etc.
- Crie uma interface de usuário simples para interação com o sistema (pode ser uma interface de linha de comando inicialmente).

```
bClienteEncontrado = False
iNomeObjeto = input("QUAL O NOME DO CLIENTE?\n")
for cliente in lListaClientes:
    if cliente.nome == iNomeObjeto:
        print(cliente.info)
        bClienteEncontrado = True
if bClienteEncontrado == False:
    print("CLIENTE NAO ENCONTRADO\n")
main()
```

```
bProdutoEncontrado = False
iNomeObjeto = input("QUAL O NOME DO PRODUTO?\n")
for produto in lListaProdutos:
    if produto.nome == iNomeObjeto:
        print(produto.info)
        bProdutoEncontrado = True
if bProdutoEncontrado == False:
    print("PRODUTO NAO ENCONTRADO")
main()
```

Requisitos Técnicos:

- Utilize princípios de herança, encapsulamento e polimorfismo em seu projeto.
- Utilize coleções (listas, arrays, etc.) para armazenar dados e evite o uso excessivo de variáveis globais.
- Implemente tratamento de exceções para lidar com situações como tentativas de consultar clientes ou produtos inexistentes, adicionar produtos com informações incompletas, etc.
- Crie uma interface de usuário simples para interação com o sistema (pode ser uma interface de linha de comando inicialmente).

```
print("BEM VINDO AO GERENCIAMENTO DE PEDIDOS ONLINE\n")
```

```
sTipoOperacao = input("O QUE VOCE GOSTARIA DE FAZER? (MOSTRAR, CONSULTAR, CRIAR, ATUALIZAR OU REMOVER)\n")
```

```
iTipoObjetoInput = input("QUAL TIPO DE OBJETO VOCE GOSTARIA DE ACESSAR (CLIENTE, PRODUTO OU PEDIDO)\n")
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Projects\SeminarioSchalata> py main.py
BEM VINDO AO GERENCIAMENTO DE PEDIDOS ONLINE

QUAL TIPO DE OBJETO VOCE GOSTARIA DE ACESSAR (CLIENTE, PRODUTO OU PEDIDO)
```