PROGRAMAÇÃO ESTRUTURADA
AP1 – parte 2 – TRABALHO EM GRUPO
PROF. CLAYTON JONES ALVES DA SILVA

Condições gerais:

1. O trabalho (parte 2 da AP1) perfaz 50% da nota da primeira avaliação bimestral.
2. O trabalho deve ser realizado e **submetido em grupo**, conforme definido em sala.
3. A entrega do pedido será realizada por e-mail para o endereço de e-mail clayton.silva@professores.ibmec.edu.br.
4. **Somente o representante do grupo submeterá o trabalho por e-mail**.
5. É **obrigatório** que a entrega contemple a **auto-avaliação do grupo,** em cada arquivo, quanto à participação de cada membro, identificando nome – matricula – e a escala - TA (trabalhou ativamente) ou TP (trabalhou parcialmente) ou NT (não trabalhou).
6. Data de entrega do trabalho: **5 de maio de 2023**.

Especificações do sistema:

1. Descrição geral

Uma loja está automatizando seus processos de negócios e irá desenvolver um sistema, designado ***SisLoja***. Em uma fase inicial, irá implantar os **módulos** do sistema *SisLoja* para:

1) gerenciar o estoque (***GerEstoque***);

2) gerenciar clientes (***GerClientes***); e

3) gerenciar o fluxo de caixa (***GerCaixa***).

Após o cliente escolher uma opção e realizar a operação selecionada no módulo principal, o **sistema deverá retornar à tela principal**.

1.1 Funcionalidades do Módulo *GerEstoque*

O módulo ***GerEstoque*** deve permitir que o usuário possa **incluir ou excluir itens** no estoque, selecionando uma das duas opções. Portanto, o uso do módulo *GerEstoque* requer dois tipos de transação:

- **inclusão** de itens; ou

- **exclusão** de itens.

Em um acesso **para inclusão,** o usuário **poderá incluir vários itens** ao estoque em uma mesma transação, informando preliminarmente quantos itens serão incluídos. Para incluir **cada item** no estoque o usuário deverá digitar:

1. **o código do item (*cod_item*)** – um número inteiro;
2. **uma descrição** do item **(*desc_item*);** e
3. **o valor do item**, em reais **(*valor_item*)**.

Todos os itens deverão ser armazenados em uma lista nomeada **listaEstoque**.

Após realizada toda a transação de inclusão, o sistema **apresentará** um relatório da transação com a **média dos valores** dos itens cadastrados; e o **código do item** de **maior valor** com o seu respectivo valor.

Em um acesso **para exclusão,** o usuário poderá excluir vários componentes, **item por item**, digitando o código do item a ser excluído.

Após o usuário **digitar o código do item a ser excluído** o sistema deverá **emitir uma mensagem**: "TEM CERTEZA QUE DESEJA EXCLUIR O ITEM?".

Se o usuário digitar "não", a transação será finalizada. Caso o usuário digite "sim", o sistema deverá:

1. **verificar se o código é válido** (códigos válidos possuem a soma dos dígitos superior a 30 e inferior a 100).
2. Se o código for inválido, o sistema deverá **apresentar uma mensagem** "CÓDIGO INVÁLIDO" (códigos não cadastrados devem ser tratados como códigos inválidos) e encerrar a transação; e
3. se o código for válido, o sistema deverá ler a quantidade de itens a excluir e, se **houver saldo em estoque**, deverá excluir o item solicitado e **apresentar uma mensagem** "OPERAÇÃO REALIZADA COM SUCESSO".
4. **Caso não haja saldo em estoque**, o sistema deverá apresentar uma mensagem "NÃO HÁ SALDO EM ESTOQUE" e encerrar a transação.
5. Após a exclusão bem sucedida de um item **apresentar uma mensagem** perguntando se o usuário deseja excluir um novo item.

Após realizada toda a transação de exclusão de todos os itens, o sistema deverá **apresentar um relatório com a relação** dos itens excluídos e o respectivo saldo em estoque.

1.2 Funcionalidades do Módulo *GerClientes*

O módulo ***GerClientes*** deve permitir que o **usuário possa cadastrar vários clientes**, digitando o **CPF do cliente (*cpf*)** e a **renda** (***renda_cliente***).

A operação poderá ser interrompida **quando o usuário digitar um CPF de cliente igual a 0**. O sistema deverá **verificar se o CPF do cliente é um número válido**. Utilizar o critério definido em <A Matemática nos Documentos: CPF – Clubes de Matemática da OBMEP>.

Após cadastrar os clientes, o módulo deverá **apresentar uma mensagem** "OPERAÇÃO REALIZADA COM SUCESSO", **seguida de uma mensagem** com o número de clientes cadastrados e com o **percentual de clientes** com renda superior a R$ 10.000,00; entre R$ 5.000,00 e R$ 10.000,00; e inferior a R$ 5.000,00.

Todos os clientes deverão ser armazenados em uma lista nomeada **listaClientes**.

1.3 Funcionalidades do Módulo *GerCaixa*

O módulo ***GerCaixa*** deve permitir que o usuário **registre a movimentação financeira de um determinado dia**.

O módulo deve permitir que o usuário **digite no início do lançamento dos dados:**

1. a data (*data*),
2. o saldo inicial do caixa no dia (*saldo_inicial*) e
3. o número total (*N*) de vendas realizadas.

O lançamento dos dados de vendas pelo usuário no sistema *GerCaixa* deve ser realizado pela ordem das operações realizadas naquele dia. Em cada lançamento de venda, o usuário do sistema **deverá lançar:**

1. O CPF do cliente (*cpf*),
2. o código de cada item (*cod_item*),
3. a quantidade de itens (*quant_item*) e
4. o valor unitário, em reais, (*valor_un*) do item.
5. O sistema deverá **calcular o** valor total (*tot_venda*) da venda.

Após o lançamento de todas as vendas, o sistema deverá **apresentar um relatório** com:

1. a **data** da movimentação,
2. o **saldo** final do caixa,
3. o **valor médio das vendas** no dia e
4. o **total de itens vendidos**.

Não é necessário dar baixa no estoque em cada venda realizada. Essa transação está no módulo de gestão do estoque.

2. Interface com o usuário

O sistema poderá possuir interfaces gráficas ou não. A critério do grupo. **Não é obrigatório o uso de interface gráfica**. Alguns *templates* para construção de interface gráfica estão apresentados no anexo. **Obs**. Se o grupo utilizar interface gráfica as mesmas informações exigidas nestas especificações devem ser apresentadas.

2.1 Interface do módulo principal

O **módulo principal** sem uso de interface gráfica deverá possuir a seguinte forma de tela

```
----------------------------
SISTEMA DE GESTÃO DE LOJA (SisLoja)
Selecionar a opção desejada:
Gestão de estoque (1)
Gestão de clientes (2)
Gestão de fluxo de caixa (3)
----------------------------
```

2.2 Interface do módulo *GerEstoque*

Cada transação **de inclusão** do módulo *GerEstoque* deverá gerar um relatório com a seguinte interface:

```
----------------------------
Valor médio dos itens cadastrados: R$ x,xx
Item de maior valor cadastrado: código xxx, valor R$ x,xx
----------------------------
Teclar 0 para retornar à tela principal
```

Cada transação **de exclusão** do módulo *GerEstoque* deverá gerar um relatório com a seguinte interface:

```
--------------------------------------
RELATÓRIO DE ITENS EXCLUÍDOS
ITEM          SALDO
12678         22
345699        12
--------------------------------------
Teclar 0 para retornar à tela principal
```

2.3 Interface do módulo *GerCliente*

Cada transação **de cadastro de clientes** do módulo *GerCliente* deverá gerar um relatório com a seguinte interface:

```
--------------------------------
OPERAÇÃO REALIZADA COM SUCESSO
Total de clientes cadastrados: 12
-----------------------------------------------
FAIXA                              PORCENTAGEM
Abaixo de R$ 5.000,00                  15%
Entre R$ 5.000,00 e R$ 10.000,00       70%
Acima de R$ 5.000,00                   15%
-----------------------------------------------
Teclar 0 para retornar à tela principal
```

2.3 Interface do módulo *GerCaixa*

Cada transação **de cadastro de vendas** do módulo *GerCaixa* deverá gerar um relatório com a seguinte interface:

```
--------------------------
RELATÓRIO DE VENDAS
Data da movimentação: 12/07/2023
Saldo: R$ x,xx
Valor médio das vendas: R$ x,xx
Total das vendas: 34 unidades
--------------------------
Teclar 0 para retornar à tela principal
```

3. Pedido

1. Elaborar o *script* na linguagem Python **para cada um dos módulos** descritos. **Cada módulo deve possuir um arquivo**. Os arquivos **deverão** ser designados *GerEstoque.py*; *GerClientes.py*; e *GerCaixa.py*.
2. Elaborar o *script* em Python para o módulo principal.
3. A avaliação considerará o uso apropriado dos comentários, conforme tratado em aula. **A designação do grupo assim como a auto-avaliação deverão ser lançados como comentário em cada um dos arquivos**.
4. A avaliação considerará:
    a. a entrega ou não das **funcionalidades** pedidas para cada um dos módulos da forma correta;
    b. a **clareza** do código; e
    c. a **estruturação** de acordo com o pedido.

ANEXO

Template de interface gráfica

1. Criar uma janela principal utilizando o pacote *Tkinter*

```
from tkinter import *  # Importa todos os módulos do pacote Tkinter
from tkinter import ttk  # A biclioteca ttk é uma extensão da
biblioteca tkinter

root = Tk()  # Cria uma janela principal
root.title("Módulo Principal")  # Cria o título da janela principal
frm = ttk.Frame(root, padding=10)  # Cria um container para outros
widgets e instancia frm
frm.grid()  # Cria uma grade de células
ttk.Label(frm, text="Oi, Mundo!!!!").grid(column=0, row=0)
ttk.Button(frm, text="Exit", command=root.destroy).grid(column=1,
row=0)
root.geometry("400x300")  # Ajusta a largura e altura do container
root.mainloop()  # Atualiza a interface gráfica após ocorrer um evento
                 # Trata-se de um loop infinito que espera por eventos
```

Obs: Um *container* é um *widget* que contém outros *widgets*. Ele é usado para organizar e agrupar *widgets* relacionados em uma janela.

2. Criar um botão de *ok* e uma *messagebox*

Uma *messagebox* em *tkinter* é uma janela pop-up que exibe uma mensagem para o usuário.

```
from tkinter import *  # Importa todos os módulos do pacote Tkinter
from tkinter import ttk  # A biclioteca ttk é uma extensão da
biblioteca tkinter
from tkinter import messagebox


def OkBotao():  # Cria a função que define a resposta ao evento quando
o botão Ok é acionado
    MsgBox = messagebox.showwarning(title='Alerta!', message='A
resposta está aqui')
    if MsgBox == "Ok":
        root.destroy()  # O método destroy remove a janela ou widget
da tela e libera todos os recursos associados a ele


root = Tk()  # Cria uma janela principal
root.title("Módulo Principal")  # Cria o título da janela principal
frm = ttk.Frame(root, padding=10)  # Cria um container para outros
widgets e instancia frm
frm.grid()  # Cria uma grade de células
ttk.Label(frm, text="Oi, Mundo!!!!").grid(column=0, row=0)
ttk.Button(frm, text="Exit", command=root.destroy).grid(column=1,
row=1)
ttk.Button(frm, text="Ok", command=OkBotao).grid(column=0, row=1)
root.geometry("400x100")  # Ajusta a largura e altura do container
root.mainloop()  # Atualiza a interface gráfica após ocorrer um evento
                 # Trata-se de um loop infinito que espera por eventos
```

3. Criar uma caixa de texto e uma *Combobox*

Uma *combobox* em *tkinter* é um *widget* de interface gráfica do usuário que permite ao usuário **selecionar um valor** de uma lista suspensa.

Uma caixa de texto em *tkinter* é um *widget* que permite ao usuário **inserir texto** em um aplicativo Python com interface gráfica.

Observe que foram criados dois *containers*.

```
from tkinter import *  # Importa todos os módulos do pacote Tkinter
from tkinter import ttk  # A biclioteca ttk é uma extensão da
biblioteca tkinter
from tkinter import messagebox
# from tkinter import Tk, Text
from tkinter import Text


def OkBotao():  # Cria a função que define a resposta ao evento quando
o botão Ok é acionado
    MsgBox = messagebox.showwarning(title='Alerta!', message='A
resposta está aqui')
    if MsgBox == "Ok":
        root.destroy()  # O método destroy remove a janela ou widget
da tela e libera todos os recursos associados a ele


lista = ["valor 1", "valor 2", "valor 3"]  # Deefine a lista de
valores da combobox
root = Tk()  # Cria uma janela principal
root.title("Módulo Principal")  # Cria o título da janela principal
frm1 = ttk.Frame(root, padding=10)  # Cria um container para outros
widgets e instancia frm1
frm1.grid()  # Cria uma grade de células
ttk.Label(frm1, text="Oi, Mundo!!!!").grid(column=0, row=0)
Text(frm1, height=1, width=15).grid(column=0, row=1)
variavel_controle = StringVar()  # StringVar() é uma classe usada para
criar uma variável de controle
                                 # para widgets de texto
ttk.Combobox(frm1, textvariable=variavel_controle, values=lista,
height=1, width=15).grid(column=1, row=1)
frm2 = ttk.Frame(root, padding=10)  # Cria um container para Ok e Exit
e instancia frm2
frm2.grid()  # Cria uma grade de células
ttk.Button(frm2, text="Ok", command=OkBotao).grid(column=0, row=0)
ttk.Button(frm2, text="Exit", command=root.destroy).grid(column=1,
row=0)
root.geometry("400x100")  # Ajusta a largura e altura do container
root.mainloop()  # Atualiza a interface gráfica após ocorrer um evento
                 # Trata-se de um loop infinito que espera por eventos
```

4. Utilizar o método Text.get()

Observe que o método *get()* retorna string.

```
from tkinter import *  # Importa todos os módulos do pacote Tkinter
from tkinter import ttk  # A biclioteca ttk é uma extensão da
biblioteca tkinter
from tkinter import messagebox
from tkinter import Tk, Text


def OkBotao():  # Cria a função que define a resposta ao evento quando
o botão Ok é acionado
    if variavel_controle.get() == "fatorial":
        mensagem = fatorial(int(texto.get("1.0", "end-1c"))) # O
método get() retorna uma string
    elif variavel_controle.get() == "soma":
        mensagem = soma(int(texto.get("1.0", "end-1c")))
    else:
        mensagem = nada(int(texto.get("1.0", "end-1c")))
    MsgBox = messagebox.showwarning(title='Alerta!', message=mensagem)
    if MsgBox == "Ok":
        root.destroy()  # O método destroy remove a janela ou widget
da tela e libera todos os recursos associados a ele


def fatorial(numero):
    if numero == 0 or numero == 1:
        return 1
    else:
        return numero * fatorial(numero-1)


def soma(numero):
    if numero == 0 or numero == 1:
        return 1
    else:
        return numero + soma(numero-1)


def nada(numero):
    return numero


lista = ["fatorial", "soma", "mesmo numero"]  # Deefine a lista de
valores da combobox
root = Tk()  # Cria uma janela principal
root.title("Módulo Principal")  # Cria o título da janela principal
frm1 = ttk.Frame(root, padding=10)  # Cria um container para outros
widgets e instancia frm1
frm1.grid()  # Cria uma grade de células
ttk.Label(frm1, text="Oi, Mundo!!!!").grid(column=0, row=0)
texto = Text(frm1, height=1, width=15)
texto.grid(column=0, row=1)
variavel_controle = StringVar()  # StringVar() é uma classe usada para
criar uma variável de controle
# para widgets de texto
ttk.Combobox(frm1, textvariable=variavel_controle, values=lista,
height=1, width=15).grid(column=1, row=1)
frm2 = ttk.Frame(root, padding=10)  # Cria um container para Ok e Exit
```

```
e instancia frm2
frm2.grid()  # Cria uma grade de células
ttk.Button(frm2, text="Ok", command=OkBotao).grid(column=0, row=0)
ttk.Button(frm2, text="Exit", command=root.destroy).grid(column=1,
row=0)
root.geometry("400x100")  # Ajusta a largura e altura do container
root.mainloop()  # Atualiza a interface gráfica após ocorrer um evento
# Trata-se de um loop infinito que espera por eventos
```