PROGRAMAÇÃO ESTRUTURADA
AP2 – parte 2 – TRABALHO EM GRUPO
PROF. CLAYTON JONES ALVES DA SILVA

Condições gerais:

1. O trabalho (parte 2 da AP2) perfaz 50% da nota da segunda 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. **Todos os arquivos .py deverão ser compactados e anexados no formato .zip**. Além disso, **o código do sistema no *Git Hub* deverá ser atualizado**.
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: **19 de junho de 2023**.

Especificações do sistema:

1. Fase 1

Conforme definidas nos requisitos iniciais.
Todos os pontos indicados na avaliação da fase 1 deverão ser ajustados.
Ajustar o que for necessário em função dos requisitos da fase 2.

2. Fase 2

Módulo *GerEstoque*

a. O sistema deve manter um conjunto **atualizado** de todos os itens cadastrados no módulo *GerEstoque*, contendo *código*, *descrição*, *valor* e *saldo* em estoque.

b. Os códigos dos itens devem ser cadastrados como *strings*. São as chaves da lista contendo *descrição*, *valor* e *saldo* em estoque do respectivo item. Por exemplo, código do item ‘345694’, refere-se à lista [‘motor 150 cv’,250.76,10].

c. Após um item ser cadastrado, ainda que o saldo em estoque seja 0, deve ter mantida sua descrição e valor na lista.

d. As transações de inclusão e exclusão operam sobre o mesmo conjunto de dados dos itens e devem ser sincronizadas.

e. Além das transações de incluir e excluir itens, criar a transação **alterar item**. A transação deve permitir atualizar a descrição, o valor e o saldo em estoque.

Módulo *GerCliente*

f. O sistema deve manter um conjunto **atualizado** de todos os itens cadastrados no módulo *GerCliente*, contendo *cpf*, *renda*.

g. Os códigos de cpf devem ser cadastrados como *strings*. São as chaves da *renda*. Por exemplo, código do item 'xxx.xxx.xxx-xx', refere-se à renda R$ 10.000,00.

h. As transações de inclusão e exclusão operam sobre o mesmo conjunto de dados dos itens e devem ser sincronizadas.

i. Além da transação de incluir clientes, criar a transação **alterar cliente**. A transação deve permitir atualizar a renda do cliente.

Módulo *GerCaixa*

j. O sistema deve manter um cadastro das vendas da organização. A lista de vendas deve conter *data da venda, código do cliente, código do item vendido, quantidade vendida do item.*

k. Além da **transação de movimentação financeira**, criar a transação **cadastrar venda**.

l. O sistema deve permitir determinar a quantidade de itens vendidos de acordo com o dia pesquisado.

3. 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
```

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

```
--------------------------------------
RELATÓRIO DE ITEM ALTERADO
ITEM          DESCRICAO       VALOR      SALDO EM ESTOQUE
345694        motor 150 cv    250.76      10
--------------------------------------
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
```

Cada transação **de alterar clientes** deverá gerar o mesmo relatório.

2.3 Interface do módulo *GerCaixa*

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

```
--------------------------
RELATÓRIO DE MOVIMENTAÇÃO FINANCEIRA
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
```

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

```
--------------------------
CADASTRO DA VENDA
Data da movimentação: 12/07/2023
Código do cliente: xxx.xxx.xxx-xx
Código do item: 345694
Total das vendas: 2 unidades
--------------------------
Teclar 0 para retornar à tela principal
```

4. 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
```