

SIMPAR – Simulação de Passageiros em Partida Aérea

TRABALHO DE ALGORITMOS E ESTRUTURAS DE DADOS
PROJETO FASE 1

PAULO ENES DA SILVEIRA

António Sabino 30000481
Bruno Saraiva 20160782
Eros Eloy 30000154
Hafsa Ben Messaoud 30001577

| Turma A2 Diurno |

▪ Índice

| | |
|--|-------|
| 1. Introdução..... | 3 |
| 2. Código: | |
| 2.1 Descrição e explicação das estruturas de dados utilizadas..... | 3-4 |
| 2.2 Descrição do código implementado..... | 5-13 |
| 3. Manual do Utilizador..... | 14-15 |
| 4. Conclusão..... | 16 |
| 4.1 Objetivos alcançados | |
| 4.2 Dificuldades | |
| 4.3 Organização do grupo | |
| 5. Listagem do Programa Fonte..... | 17-24 |

1. Introdução

No desenvolvimento deste trabalho iremos abordar vários subtemas e conceitos, como por exemplo; as classes, os objectos, a estrutura dos dados, entre outros.

Um programa é uma lista de instruções escritas para resolver um problema, ou para executar uma ação. Estas instruções são elaboradas sob forma de código, e este último indica ao programa o que tem de fazer e de que forma.

Uma classe é uma estrutura que abstrai um conjunto de objetos com características similares. Ela define o comportamento de seus objetos (através de métodos), e os seus respetivos estados possíveis (através de atributos).

Neste projeto iremos apenas utilizar o *python*. Esta ferramenta é uma linguagem de programação orientada a objetos, multi-paradigma e multiplataforma que promove a programação imperativa estruturada, funcional e orientada para objetos. O objetivo é consolidar a matéria dada nas aulas, principalmente o estudo das filas de espera, através do tema do “Aeroporto” proposto pelo professor, no sentido de avaliar a possibilidade de optimização deste serviço de atendimento.

2. Código:

2.1 Descrição e explicação das estruturas de dados utilizadas

Na ciência da computação, uma estrutura de dados é um modo particular de armazenamento e organização de dados em um computador de modo que possam ser usados eficientemente, facilitando assim a sua busca e modificação.

As estruturas de dados dividem-se em homogêneas (matrizes) e heterogêneas (registros):

- Estruturas homogêneas (matrizes, classes):

Utilizadas para a construção das tabelas, ou seja, são conjuntos de dados formados pelo mesmo tipo de dado primitivo. Por exemplo:

```
class Passageiro:
```

```
    def __init__(self,bag_pass,ciclo_in):
```

```
        self.bag_pass = bag_pass
```

```
        self.ciclo_in = ciclo_in
```

```
    def __str__(self):
```

```
        return '[b:' + str(self.bag_pass) + ' t:' + str(self.ciclo_in) + ']'
```

```
    def obtem_bag_pass(self):
```

```
        return self.bag_pass
```

```
    def obtem_ciclo_in(self):
```

```
        return self.ciclo_in
```

- Estruturas heterogêneas (registros):

As listas e as funções definidas são registros pois são conjuntos de dados formados por tipos de dados diferentes. Por exemplo:

```
def existem_balcoes_com_fila(list):
    for g in range(len(list)):
        if not list[g].obtem_fila().isEmpty():
            return True

def simpar_simula(num_pass, num_bag, num_balcoes, ciclos):
    balcoes = []
    for i in range(num_balcoes):
        balcoes.append(Balcao(i,randint(1,num_bag)))
        i+=1
    for i in range(ciclos):
        print('\n««« CICLO nº ', i, ' »»»')
        atende_passageiros(i,balcoes)
        balcaomenor = fila_menor(balcoes)
        if passl(num_pass,balcoes)!=0:
            if (i == (ciclos)/3) or (i < (2*(ciclos))/3):
                balcoes[balcaomenor].fila.enqueue(Passageiro((randint(1,num_bag)),i))
            elif (i == (2*ciclos)/3) or (i < ciclos):
                if randint(0,100) < 80:
                    balcoes[balcaomenor].fila.enqueue(Passageiro((randint(1,num_bag)),i))
            elif (i == (ciclos)):
                if randint(0,100) < 60:
                    balcoes[balcaomenor].fila.enqueue(Passageiro((randint(1,num_bag)),i))
```

2.2 Descrição do código implementado

Importação das bibliotecas vitais e criação das classes para a execução do programa

```
from random import randint
```

```
from pythonds.basic import Queue
```

```
class Passageiro:
```

```
    def __init__(self,bag_pass,ciclo_in):
```

```
        self.bag_pass = bag_pass
```

```
        self.ciclo_in = ciclo_in
```

```
    def __str__(self):
```

```
        return '[b:' + str(self.bag_pass) + ' t:' + str(self.ciclo_in) + ']'
```

```
#Metodos obtem
```

```
    def obtem_bag_pass(self):
```

```
        return self.bag_pass
```

```
    def obtem_ciclo_in(self):
```

```
        return self.ciclo_in
```

```
class Balcao:
```

```
    def __init__(self,n_balcao,bag_utemp):
```

```
        self.n_balcao = n_balcao
```

```
        self.fila = Queue()
```

```
        self.inic_atend = 0
```

```
        self.passt_atend = 0
```

```
        self.numt_bag = 0
```

```
        self.tempt_esp = 0
```

```
        self.bag_utemp = bag_utemp
```

```
    def __str__(self):
```

```
        return '[b:' + str(self.n_balcao) + ' t:' + str(self.inic_atend)
```

```
        + ' fila:' + str(self.fila.size()) + ']'
```

Métodos de mudança de variáveis

```
def muda_inic_atend(self, tempoatendimento):  
    self.inic_atend += tempoatendimento
```

```
def incr_passt_atend(self):  
    self.passt_atend += 1
```

```
def muda_numt_bag(self, bagpass):  
    self.numt_bag += bagpass
```

```
def muda_tempt_esp(self,t):  
    self.tempt_esp += t
```

Métodos para obter os números

```
def obtem_n_balcao(self):  
    return self.n_balcao
```

```
def obtem_fila(self):  
    return self.fila
```

```
def obtem_inic_atend(self):  
    return self.inic_atend
```

```
def obtem_passt_atend(self):  
    return self.passt_atend
```

```
def obtem_numt_bag(self):  
    return self.numt_bag
```

```
def obtem_tempt_esp(self):  
    return self.tempt_esp
```

```
def obtem_bag_utemp(self):  
    return self.bag_utemp
```

Função que demonstra todos os balcões

```
def mostra_balcoes(list):
```

```
    #print inicial
```

```
    print("\n| Lista de balcões |")
```

```
    # Para todos os balcões
```

```
    for b in range(len(list)):
```

```
        print('    '+str(list[b]))
```

#Verificar se existem ainda balcoes com fila

```
def existem_balcoes_com_fila(list):
```

```
    #Para todos os balcões
```

```
    for g in range(len(list)):
```

```
        #Se o balcao b não estiver vazio
```

```
            if not list[g].obtem_fila().isEmpty():
```

```
                #Return True quer dizer que ainda existem
```

```
                #balcões com fila
```

```
                return True
```

#Encontrar a fila menor

```
def fila_menor(balcoes):
```

```
    #Número do balcão menor
```

```
    balcaomenor = 0
```

```
    #Tamanho da fila do balcão menor
```

```
    menorB = balcoes[0].obtem_fila().size()
```

```
    #Para todos os balcões, a partir do balcão 1 que compara com o 0
```

```
    for indice in range(1,len(balcoes)):
```

```
        #Se o balcão j tiver uma fila de menor tamanho
```

```
        if balcoes[indice].obtem_fila().size() < menorB:
```

```
            #O tamanho da fila menor é atualizado
```

```
            menorB = balcoes[indice].obtem_fila().size()
```

```
            #O numero do balcão com a fila menor é atualizado
```

```
            balcaomenor = indice
```

```
    #Return do numero do balcão com menor fila
```



```
return balcaomenor
```

```
#Numero de lugares livres para passageiros:
```

```
#maxp é o máximo de passageiros
```

```
#list é o a lista balcoes
```

```
def passl(maxp,list):
```

```
    #Passageiros totais
```

```
    passtotal=0
```

```
    #Para todos os balcões
```

```
    for h in range(len(list)):
```

```
        #Incrementa ao total de passageiros, o total de cada balcão
```

```
        passtotal+=list[h].obtem_passt_atend()
```

```
        #Return do número de lugares livres
```

```
    return (maxp-passtotal)
```

```
def atende_passageiros(tempo,balcoes):
```

```
    for c in range(len(balcoes)):
```

```
#A função 'def_atende_passageiros' vai ler o número de balcões existentes na lista de balcões.
```

```
        balcao=balcoes[c]
```

```
        if not balcao.obtem_fila().isEmpty():
```

```
            for c in range(balcao.obtem_fila().size()):
```

```
#Se a lista não for vazia então entramos no ciclo for
```

```
                p = balcao.obtem_fila().items[0]
```

```
                bagp = p.obtem_bag_pass()
```

```
                tempo_de_atendimento = tempo - balcao.obtem_inic_atend()
```

```
                ut_bag = bagp / balcao.obtem_bag_utemp()
```

```
                if ut_bag < tempo_de_atendimento:
```

```
#Se o tempo para despachar as bagagens de p for menor que o tempo de atendimento então executa-se o seguinte:
```

```

tempo_de_espera = tempo - p.ciclo_in
balcao.muda_inic_atend((tempo+1))
balcao.incr_passt_atend()
balcao.muda_numt_bag(bagp)
balcao.muda_tempt_esp(tempo_de_espera)
balcao.obtem_fila().dequeue()

```

else:

#Caso contrário se o tempo para despachar as bagagens for maior então:

```

if balcao.obtem_fila().isEmpty():
    balcao.muda_inic_atend(tempo)

```

#Criação das variáveis para obter os resultados finais

def apresenta_resultados(balcoes):

```

print("\n--- RESULTADOS FINAIS ---")

```

```

for i in range(len(balcoes)):

```

```

    #Para cada balcão temos:

```

```

    balcao=balcoes[i]

```

```

    #Se o balcão atual atendeu passageiros, afixa

```

```

    if balcao.obtem_passt_atend() > 0:

```

```

        bagpciclo = balcao.obtem_bag_utemp()

```

```

        atend = balcao.obtem_passt_atend()

```

```

        bagppass = (balcao.obtem_numt_bag())/balcao.obtem_passt_atend()

```

```

        tme = (balcao.obtem_tempt_esp())/balcao.obtem_passt_atend()

```

```

    #Print do relatório final do balcão

```

```

    print("\n-- Balcão "+str(i)+" despachou "+ str(bagpciclo) +\

```

```

        " bagagens por ciclo: \n"

        + str(atend) + " passageiros atendidos com média de
bagagens/passageiro = "\

        + str(round(bagppass, 2)) +\

        ".\nTempo médio de espera = "+ str(round(tme, 2)))

#Se não foram atendidos passageiros, afixa:

else:

    print("\n-- Balcão "+str(i)+" não atendeu passageiros")

```

#Função Simulação, corpo fundamental da simulação

```

def simpar_simula(num_pass, num_bag, num_balcoes, ciclos):

    balcoes = []

    for i in range(num_balcoes):

        balcoes.append(Balcao(i,randint(1,num_bag)))

        i+=1

    for i in range(ciclos):

        print('\n««« CICLO nº ', i, ' »»»')

        atende_passageiros(i,balcoes)

        balcaomenor = fila_menor(balcoes)

        if passl(num_pass,balcoes)!=0:

            if (i == (ciclos)/3) or (i < (2*(ciclos))/3):

                balcoes[balcaomenor].fila.enqueue(Passageiro((randint(1,num_bag)),i))

            elif (i == (2*ciclos)/3) or (i < ciclos):

                if randint(0,100) < 80:

                    balcoes[balcaomenor].fila.enqueue(Passageiro((randint(1,num_bag)),i))

            elif (i == (ciclos)):

                if randint(0,100) < 60:

```

```

        balcoes[balcaomenor].fila.enqueue(Passageiro((randint(1,num_bag)),i))

    mostra_balcoes(balcoes)

    ciclo_atual=ciclos

    while existem_balcoes_com_fila(balcoes):

        print('\n« « « Ciclo nº ' + str(ciclo_atual) + ' » » »')

        atende_passageiros (ciclo_atual, balcoes)

        mostra_balcoes(balcoes)

        ciclo_atual+=1

    apresenta_resultados(balcoes)

```

#Função principal que executa a simulação e permite a entrada via utilizador de dados

```

def simular():

    print("\nSimulação de Passageiros em Partida Aérea\n ")

    ciclos = int(input('Quantos ciclos para esta simulação ? '))

    num_balcoes = int(input('Número de balcões abertos? '))

    num_pass = int(input('Quantidade de passageiro com bagagens para o voo? '))

    num_bag = int(input('Quantidade máxima de bagagens estipulada ? '))

    simpar_simula(num_pass,num_bag,num_balcoes,ciclos)

    Pergunta = input("Deseja voltar a simular com outros valores? [S/N] ")

    if (Pergunta.upper() == 'S'):

```

```
simular()
```

```
if (Pergunta.upper() == 'N'):  
    print('Obrigado por utilizar o SIMPAR.')
```

```
#Função principal
```

```
#Execução
```

```
if __name__ == '__main__':
```

```
    simular()
```

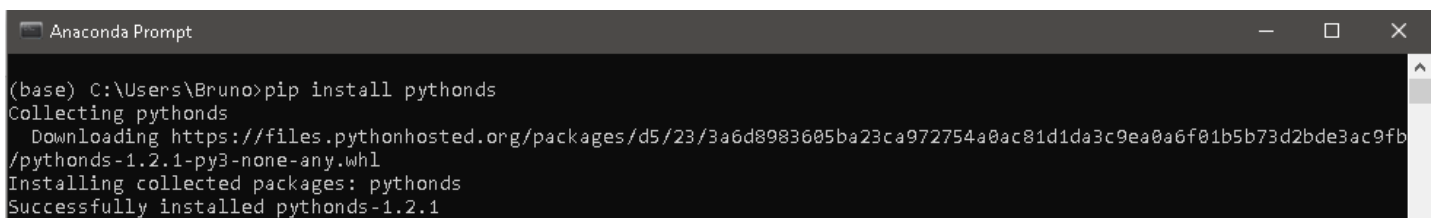
3. Manual do Utilizador

- 1- O primeiro passo é instalar a biblioteca pythonds para a utilização do comando 'from pythonds.basic import Queue', que nos permite a utilização do método



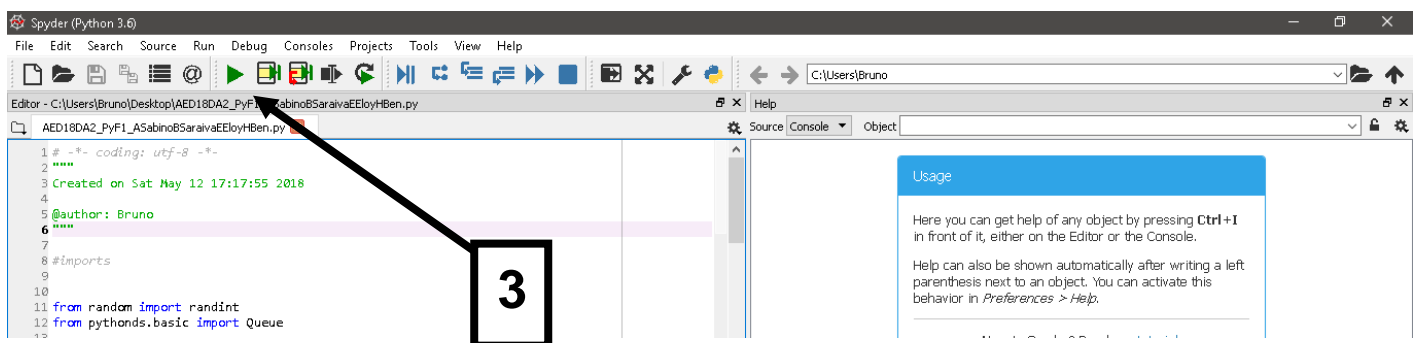
```
Anaconda Prompt
(base) C:\Users\Bruno>pip install pythonds
```

- 2- Receberá a mensagem no Anaconda prompt, caso tenha sucesso a enunciar o seguinte:



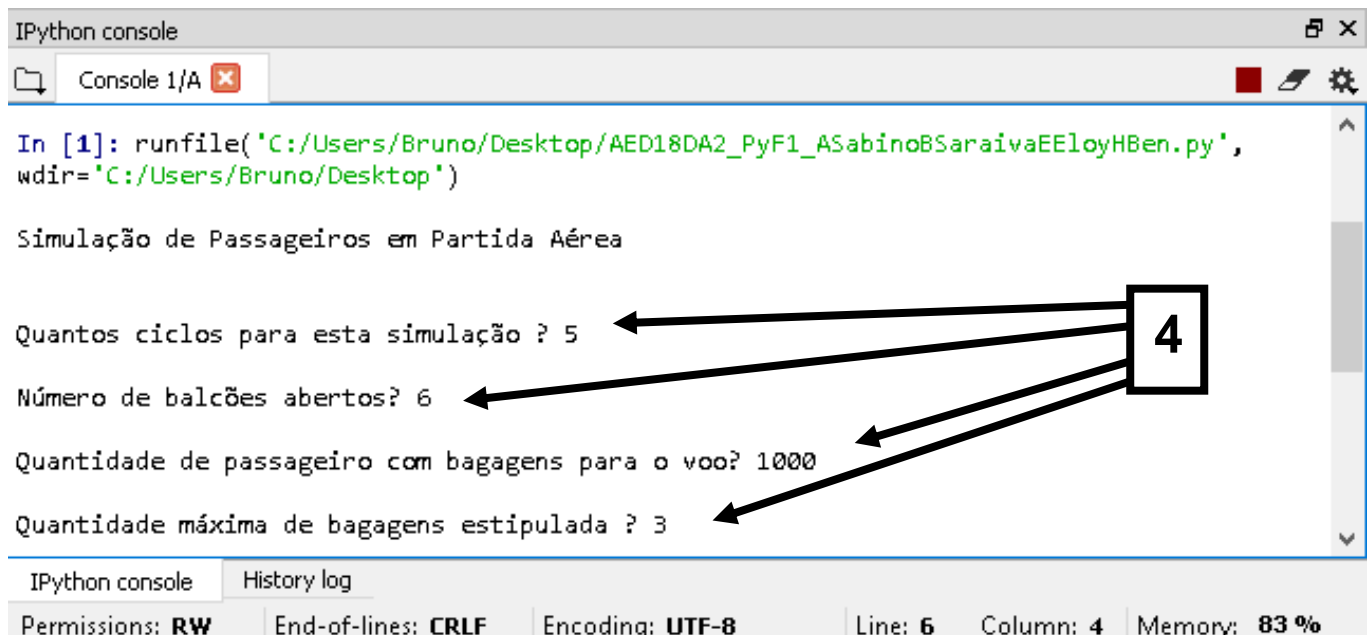
```
Anaconda Prompt
(base) C:\Users\Bruno>pip install pythonds
Collecting pythonds
  Downloading https://files.pythonhosted.org/packages/d5/23/3a6d8983605ba23ca972754a0ac81d1da3c9ea0a6f01b5b73d2bde3ac9fb/pythononds-1.2.1-py3-none-any.whl
Installing collected packages: pythonds
Successfully installed pythonds-1.2.1
```

Agora é possível utilizar a biblioteca pythonds.



- 3- Inicializar o programa clicando na seta verde, como demonstrado na figura, ou se preferir carregue na tecla F5;

- 4- Introduzir apenas números, digitar um carácter sem ser um número irá causar um erro no programa e terá de voltar à instrução 3 do manual de utilizador.



```

IPython console
Console 1/A

In [1]: runfile('C:/Users/Bruno/Desktop/AED18DA2_PyF1_ASabinoBSaraivaEEloyHBen.py',
wdir='C:/Users/Bruno/Desktop')

Simulação de Passageiros em Partida Aérea

Quantos ciclos para esta simulação ? 5
Número de balcões abertos? 6
Quantidade de passageiro com bagagens para o voo? 1000
Quantidade máxima de bagagens estipulada ? 3

IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 6 Column: 4 Memory: 83 %

```

- 5- O programa é executado e temos a informação sobre o atendimento dos passageiros com mala nos respetivos balcões bem como os ciclos.

```

««« Ciclo nº 8 »»»
| Lista de balcões |
| b:0 t:16         |
| b:1 t:10         |
| b:2 t:0          |
| b:3 t:0          |
| b:4 t:0          |

Fim Da Simulação:


O Balcão 0 teve uma média de bagagens por ciclo de 4, e 3 passageiros atendidos com estimativa de bagagens: 2.67.
Tempo: 2.67

O Balcão 1 teve uma média de bagagens por ciclo de 3, e 2 passageiros atendidos com estimativa de bagagens: 1.5.
Tempo: 1.5

O Balcão 2 não despachou bagagens neste simulação
O Balcão 3 não despachou bagagens neste simulação
O Balcão 4 não despachou bagagens neste simulação

Deseja voltar a simular com outros valores? [S/N] n
Obrigado por utilizar o SIMPAR.

```



- 6- Após a visualização dos resultados será feita a questão: 'Deseja voltar a simular com outros valores? [S/N]', então deverá responder somente com 'S' e 'N'. Se a resposta for 'S' então programa vai reiniciar, se a resposta for 'N' programa simulação acaba.

4. Conclusão

Neste trabalho, cujo objetivo era criar um programa com a finalidade de se ver a organização do sistema de um aeroporto, chegamos a conclusão que a realização deste projeto foi muito importante para o nosso desempenho e desenvolvimento, pois ajudou a consolidar e a compreender melhor a matéria abordada nas aulas. Permitiu também aprofundar os nossos conhecimentos a cerca de vários comandos utilizados Python como também tirar algumas dúvidas.

Tivemos algumas dificuldades no código, nomeadamente, no número a considerar nos balcões e também na distribuição dos passageiros pelos respetivos balcões. Apesar dessa dificuldade o código funciona corretamente.

Devido à má organização de todos os membros de grupo não foi possível entregar um código com todos os erros corrigidos bem como explicar vários aspetos importantes do código, deste modo o documento em papel será entregue no dia seguinte à data limite de entrega de trabalho, apenas o código e o relatório foram entregues dentro da data limite.

5. Listagem do Programa Fonte

```
#imports

from random import randint

from pythonds.basic import Queue

# Principais classes a considerar - início

#PASSAGEIRO

class Passageiro:

    def __init__(self,bag_pass,ciclo_in):

        self.bag_pass = bag_pass

        self.ciclo_in = ciclo_in

    def __str__(self):

        return '[b:' + str(self.bag_pass) + ' t:' + str(self.ciclo_in) + ']'

    def obtem_bag_pass(self):

        return self.bag_pass

    def obtem_ciclo_in(self):

        return self.ciclo_in

#BALCÃO

class Balcao:

    def __init__(self,n_balcao,bag_utemp):

        self.n_balcao = n_balcao

        self.fila = Queue()

        self.inic_atend = 0
```

```
self.passt_atend = 0

self.numt_bag = 0

self.tempt_esp = 0

self.bag_utemp = bag_utemp

def __str__(self):

    return '[b:' + str(self.n_balcao) + ' t:' + str(self.inic_atend)

    + ' fila:' + str(self.fila.size()) + ']'

def muda_inic_atend(self, tempoatendimento):

    self.inic_atend += tempoatendimento

def incr_passt_atend(self):

    self.passt_atend += 1

def muda_numt_bag(self, bagpass):

    self.numt_bag += bagpass

def muda_tempt_esp(self,t):

    self.tempt_esp += t

def obtem_n_balcao(self):

    return self.n_balcao

def obtem_fila(self):

    return self.fila

def obtem_inic_atend(self):

    return self.inic_atend

def obtem_passt_atend(self):

    return self.passt_atend
```

```
def obter_numt_bag(self):  
    return self.numt_bag  
  
def obter_tempt_esp(self):  
    return self.tempt_esp  
  
def obter_bag_utemp(self):  
    return self.bag_utemp  
  
# Principais classes a considerar - Fim  
  
# Principais funções a considerar - Início  
def mostra_balcoes(list):  
    print("\n| Lista de balcões |")  
  
    for b in range(len(list)):  
  
        print('    '+str(list[b]))  
  
def existem_balcoes_com_fila(list):  
  
    for g in range(len(list)):  
  
        if not list[g].obter_fila().isEmpty():  
  
            return True  
  
def fila_menor(balcoes):  
  
    balcaomenor = 0
```

```
menorB = balcoes[0].obtem_fila().size()

for indice in range(1,len(balcoes)):

    if balcoes[indice].obtem_fila().size() < menorB:

        menorB = balcoes[indice].obtem_fila().size()

        balcaomenor = indice

return balcaomenor

def passl(maxp,list):

    passtotal=0

    for h in range(len(list)):

        passtotal+=list[h].obtem_passt_atend()

    return (maxp-passtotal)

def atende_passageiros(tempo,balcoes):

    for c in range(len(balcoes)):

        balcao=balcoes[c]

        if not balcao.obtem_fila().isEmpty():
```

```

for c in range(balcao.obtem_fila().size()):

    p = balcao.obtem_fila().items[0]

    bagp = p.obtem_bag_pass()

    tempo_de_atendimento = tempo - balcao.obtem_inic_atend()

    ut_bag = bagp / balcao.obtem_bag_utemp()

    if ut_bag < tempo_de_atendimento:

        tempo_de_espera = tempo - p.ciclo_in

        balcao.muda_inic_atend((tempo+1))
        balcao.incr_passt_atend()
        balcao.muda_numt_bag(bagp)
        balcao.muda_tempt_esp(tempo_de_espera)

        balcao.obtem_fila().dequeue()

    else:

        if balcao.obtem_fila().isEmpty():
            balcao.muda_inic_atend(tempo)

#Apresenta os resultados finais
def apresenta_resultados(balcoes):

    print("\nFim Da Simulação: ")

    for k in range(len(balcoes)):

```

```
balcao=balcoes[k]
```

```
if balcao.obtem_passt_atend() > 0:
```

```
    bagpciclo = balcao.obtem_bag_utemp()
```

```
    atend = balcao.obtem_passt_atend()
```

```
    bagppass = (balcao.obtem_numt_bag()/balcao.obtem_passt_atend())
```

```
    tme = (balcao.obtem_tempt_esp()/balcao.obtem_passt_atend())
```

```
    print('\nO Balcão '+str(k)+' teve uma média de bagagens por ciclo de '+ str(bagpciclo) +', e '\
```

```
    + str(atend) + ' passageiros atendidos com estimativa de bagagens: '\
```

```
    + str(round(bagppass, 2)) +\
```

```
    '\nTempo: '+ str(round(tme, 2)))
```

```
else:
```

```
    print('\n O Balcão '+ str(k) +' não despachou bagagens neste simulação')
```

```
#Simulação
```

```
def simpar_simula(num_pass, num_bag, num_balcoes, ciclos):
```

```
    balcoes = []
```

```
    for i in range(num_balcoes):
```

```
        balcoes.append(Balcao(i,randint(1,num_bag)))
```

```
    i+=1
```

```
    for i in range(ciclos):
```

```
        print("\n««« CICLO nº ', i, ' »»»")
```

```
        atende_passageiros(i,balcoes)
```

```

    balcaomenor = fila_menor(balcoes)

    if passl(num_pass,balcoes)!=0:

        if (i == (ciclos)/3) or (i < (2*(ciclos))/3):

            balcoes[balcaomenor].fila.enqueue(Passageiro((randint(1,num_bag)),i))

        elif (i == (2*ciclos)/3) or (i < ciclos):

            if randint(0,100) < 80:

                balcoes[balcaomenor].fila.enqueue(Passageiro((randint(1,num_bag)),i))

            elif (i == (ciclos)):

                if randint(0,100) < 60:

                    balcoes[balcaomenor].fila.enqueue(Passageiro((randint(1,num_bag)),i))

    mostra_balcoes(balcoes)

    ciclo_atual=ciclos

    while existem_balcoes_com_fila(balcoes):

        print("\n««« Ciclo nº ' + str(ciclo_atual) + ' »»»")

        atende_passageiros (ciclo_atual, balcoes)

        mostra_balcoes(balcoes)

        ciclo_atual+=1

    apresenta_resultados(balcoes)

#Principais funções a considerar - Fim

#Função principal

def simular():

    print("\nSimulação de Passageiros em Partida Aérea\n ")

```

```
ciclos = int(input('Quantos ciclos para esta simulação ? '))  
num_balcoes = int(input('Número de balcões abertos? '))  
num_pass = int(input('Quantidade de passageiro com bagagens para o voo? '))  
num_bag = int(input('Quantidade máxima de bagagens estipulada ? '))  
simpar_simula(num_pass,num_bag,num_balcoes,ciclos)  
Pergunta = input("Deseja voltar a simular com outros valores? [S/N] ")  
  
if (Pergunta.upper() == 'S'):  
  
    simular()  
  
if (Pergunta.upper() == 'N'):  
    print('Obrigado por utilizar o SIMPAR.')  
#Função principal  
  
#Execução  
if __name__ == '__main__':  
  
    simular()
```