

CENTRO UNIVERSITÁRIO INTERNACIONAL UNINTER

ESCOLA SUPERIOR POLITÉCNICA

TECNOLOGIA EM CIÊNCIA DE DADOS

ESTRUTURA DE DADOS

ATIVIDADE PRÁTICA

nome: ivan felipe de oliveira santos dourado– RU: 3357756

Profª: Luciane Kanashiro

ibicaraí – bahia

2023

# EXERCÍCIO 1

Com a finalidade de melhorar o atendimento e priorizar os casos mais urgentes, a direção de um hospital criou um sistema de triagem em que um profissional da saúde classifica a ordem de atendimento com base numa avaliação prévia do paciente, entregando-lhe um cartão numerado verde (V) ou amarelo (A), que define o menor ou maior grau de urgência da ocorrência, respectivamente. Para informatizar esse processo, o software desenvolvido tem como base o seguinte trecho de código-fonte:

(Código fonte)

Texto

Descrição gerada automaticamente

Uma imagem contendo Texto

Descrição gerada automaticamente

Na linha 30, a função inserir recebe o número e a cor do cartão entregue ao paciente na triagem. Pacientes com cartão verde são inseridos no final da fila pela função inserir No Final (linhas 18-28). Pacientes com cartão amarelo têm prioridade no atendimento e são inseridos no início da fila, em ordem de chegada, pela função inserir prioridade. Portanto, se são entregues os cartões 10-V, 11-V, 5- A, 12-V e 6-A, nessa ordem, a fila deve ficar assim organizada:

Diagrama

Descrição gerada automaticamente com confiança média

Solução do aluno:

# Aluno: Ivan Felipe De Oliveira Santos Dourado

# RU: 3357756

# Feito no VSCode utilizando Python

# Definindo a classe para o elemento da lista encadeada

class ElementoDaListaSimples:

    def \_\_init\_\_(self, dado, cor):

        self.dado = dado      # Valor ou informação que o nodo irá armazenar, no caso, o código do cartão do paciente

        self.cor = cor        # Cor do cartão do paciente, que define sua prioridade

        self.proximo = None   # Referência para o próximo nodo na lista encadeada

# Definindo a classe principal para a lista encadeada simples

class ListaEncadeadaSimples:

    def \_\_init\_\_(self, nodos=None):

        self.head = None      # Referência para o primeiro nodo da lista

        # Se nodos forem fornecidos durante a inicialização

        if nodos is not None:

            nodo = ElementoDaListaSimples(dado=nodos.pop(0))  # Criação do primeiro nodo

            self.head = nodo   # Configurando o primeiro nodo como cabeça da lista

            # Iterando sobre os nodos restantes e os adicionando à lista

            for elem in nodos:

                nodo.proximo = ElementoDaListaSimples(dado=elem)

                nodo = nodo.proximo

    # Função para inserir um nodo no final da lista

    def inserirNoFinal(self, nodo):

        # Se a lista estiver vazia

        if self.head is None:

            self.head = nodo

            return

        # Se não estiver vazia, iteramos até o último nodo

        nodo\_atual = self.head

        while nodo\_atual.proximo:

            nodo\_atual = nodo\_atual.proximo

        # Adicionando o novo nodo ao final da lista

        nodo\_atual.proximo = nodo

        return

    # Função para inserir um nodo com prioridade (cartão amarelo)

    def inserirPrioridade(self, nodo):

        # Se a lista estiver vazia

        if self.head is None:

            self.head = nodo

            return

        # Se o primeiro paciente da lista tiver um cartão verde, inserimos o paciente com cartão amarelo na primeira posição

        if self.head.cor == "V":

            nodo.proximo = self.head

            self.head = nodo

            return

        # Se o primeiro paciente já tiver um cartão amarelo, encontramos a posição logo após o último paciente com cartão amarelo

        nodo\_atual = self.head

        while nodo\_atual.proximo and nodo\_atual.proximo.cor == "A":

            nodo\_atual = nodo\_atual.proximo

        # Inserindo o novo paciente com cartão amarelo após o último paciente com cartão amarelo

        nodo.proximo = nodo\_atual.proximo

        nodo\_atual.proximo = nodo

        return

    # Função para inserir um nodo, considerando prioridade

    def inserir(self, dado, cor):

        nodo = ElementoDaListaSimples(dado, cor)   # Criando um novo nodo

        # Se a lista estiver vazia

        if self.head is None:

            self.head = nodo

            return

        # Se o nodo a ser inserido tem um cartão verde, ele vai para o final da fila

        else:

            if nodo.cor == "V":

                self.inserirNoFinal(nodo)

            # Se o nodo a ser inserido tem um cartão amarelo, ele é inserido de acordo com a prioridade

            else:

                self.inserirPrioridade(nodo)

        return

# Função para a interface de usuário

def interface\_triagem():

    lista\_pacientes = ListaEncadeadaSimples()   # Inicializando uma nova lista encadeada

    print("Sistema de Triagem de Pacientes")

    while True:

        print("\nOpções:")

        print("1 - Inserir novo paciente na fila")

        print("2 - Visualizar fila atual")

        print("3 - Encerrar")

        escolha = input("Escolha uma opção: ")

        # Adicionando um novo paciente à fila

        if escolha == "1":

            dado = input("Digite o código do cartão do paciente: ")

            while True:

                cor = input("Digite a cor do cartão (V para Verde ou A para Amarelo): ").upper()

                # Validando a entrada da cor

                if cor in ['V', 'A']:

                    break

                else:

                    print("Opção inválida. Por favor, digite V ou A.")

            # Inserindo o paciente na lista encadeada

            lista\_pacientes.inserir(dado, cor)

        # Visualizando a fila atual de pacientes

        elif escolha == "2":

            paciente\_atual = lista\_pacientes.head

            if not paciente\_atual:

                print("Fila vazia.")

                continue

            print("\nFila de Pacientes:")

            # Iterando sobre os nodos e imprimindo os detalhes

            while paciente\_atual:

                print(f"Código: {paciente\_atual.dado} - Cor: {paciente\_atual.cor}")

                paciente\_atual = paciente\_atual.proximo

        # Encerrando o sistema

        elif escolha == "3":

            print("Encerrando sistema de triagem.")

            break

        else:

            print("Opção inválida. Tente novamente.")

# Executando a interface de usuário

interface\_triagem()

Imagem do código funcionando no seu computador:

Inserindo pacientes na fila :

Texto

Descrição gerada automaticamente

Visualizando fila e encerrando aplicação:

Texto

Descrição gerada automaticamente

# EXERCÍCIO 2

Enunciado:

Uma empresa trabalha na produção de concreto e terceiriza o serviço de transporte do produto. Os caminhoneiros telefonam para a empresa e registram seu interesse pelo trabalho. Todas as manhãs os caminhoneiros estacionam o caminhão no pátio da empresa e aguarda a sua vez. O atendimento segue o critério de ordem de chegada. Esse processo é, atualmente , controlado pela secretária, que utiliza sua agenda para gerenciar os motoristas diariamente. A empresa, que carrega, no máximo 10 caminhões por dia, pretende informatizar esse processo. Implemente a solução desse problema, utilizando o conceito de fila.

Solução do aluno:

# Aluno: Ivan Felipe De Oliveira Santos Dourado

# RU: 3357756

# Feito no VSCode utilizando Python

# Definição da classe chamada FilaDeCaminhoes.

class FilaDeCaminhoes:

    # Método construtor da classe, que é chamado automaticamente ao criar uma instância da classe.

    # Ele tem um parâmetro padrão 'capacidade', que tem valor padrão de 10.

    def \_\_init\_\_(self, capacidade=10):

        self.fila = [] # Inicializa a lista 'fila' que armazena os caminhoneiros.

        self.capacidade = capacidade  # Define a capacidade máxima da fila.

    # Método para registrar o interesse de um caminhoneiro.

    def registrar\_interesse(self, caminhoneiro):

        # Verifica se a fila está cheia ou não.

        if len(self.fila) < self.capacidade:

            # Adiciona o caminhoneiro ao final da fila.

            self.fila.append(caminhoneiro)

            return f"{caminhoneiro} registrado com sucesso!"

        else:

            # Se a fila estiver cheia, informa que a capacidade foi atingida.

            return "Capacidade máxima alcançada! Por favor, tente novamente amanhã."

  # Método para atender o próximo caminhoneiro na fila.

    def atender\_caminhoneiro(self):

        # Verifica se há algum caminhoneiro na fila.

        if len(self.fila) > 0:

            # Remove o caminhoneiro do começo da fila e o armazena na variável

            caminhoneiro\_atendido = self.fila.pop(0)

            return f"Atendendo {caminhoneiro\_atendido}."

        else:

            # Se a fila estiver vazia, informa que não há caminhoneiros na fila.

            return "Nenhum caminhoneiro na fila!"

# Método para listar os caminhoneiros na fila

    def listar\_fila(self):

        # Retorna os caminhoneiros da fila como uma string separada por vírgulas.

        return ", ".join(self.fila)

    # Método que fornece uma interface de usuário simples para interagir com a fila.

    def interface(self):

# Loop infinito para continuar executando até que o usuário decida encerrar.

        while True:

             # Imprime as opções disponíveis.

            print("\nOpções:")

            print("1 - Registrar interesse")

            print("2 - Atender caminhoneiro")

            print("3 - Listar fila")

            print("4 - Encerrar")

            op = int(input("Escolha uma opção: "))# Solicita ao usuário que escolha uma opção.

             # Estrutura condicional para determinar a ação a ser tomada com base na escolha do usuário.

            if op == 1:

                caminhoneiro = input("Digite o nome do caminhoneiro: ")

                print(self.registrar\_interesse(caminhoneiro))

            elif op == 2:

                print(self.atender\_caminhoneiro())

            elif op == 3:

                print(f"Caminhoneiros na fila: {self.listar\_fila()}")

            elif op == 4:

                print("Encerrando sistema...")

                break   # Encerra o loop e, consequentemente, a interface.

            else:

                print("Opção inválida!")

# Criando uma instância da classe FilaDeCaminhoes.

fila = FilaDeCaminhoes()

# Chamando o método interface() para que o usuário possa interagir com a fila.

fila.interface()

Imagem do código funcionando no seu computador:  
  
Adicionando caminhoneiros a fila:  
Texto

Descrição gerada automaticamente

Listando fila e atendendo caminhoneiros:

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Retornando fila vazia e encerrando aplicação:

Texto

Descrição gerada automaticamente