FATEC JAHU

AVALIAÇÃO P1 – ESTRUTURA DE DADOS – 17/04/2025

Prof. Me. Tiago Antonio da Silva

CORREÇÃO DAS QUESTÕES

1) Você foi contratado para automatizar o atendimento de uma clínica com duas filas:
   - Fila Normal: pacientes comuns.
   - Fila Emergencial: pacientes com prioridade.

   Cada atendimento é registrado em uma pilha de histórico que permite desfazer o último atendimento (em caso de erro).
   Tarefas:
   a) Implemente as duas filas.
   b) A cada atendimento, retire da fila de emergência se houver, senão da normal.
   c) Registre o nome do paciente atendido na pilha de histórico.
   d) Implemente uma função de desfazer último atendimento, retornando o paciente à fila correta.

```
// sua implementação
const Fila = require('../../Aula 5 - Fila/Fila.js');

let filaNormal = new Fila();
let filaEmergencial = new Fila();
let historico = []; // pilha

function adicionarPaciente(nome, emergencia = false) {
    if (emergencia) filaEmergencial.enqueue(nome);
    else filaNormal.enqueue(nome);
}

function atenderPaciente() {
    let paciente;
    if (!filaEmergencial.isEmpty()) {
        paciente = filaEmergencial.dequeue();
    } else if (!filaNormal.isEmpty()) {
        paciente = filaNormal.dequeue();
    } else {
        console.log("Nenhum paciente para atender.");
        return;
    }
```

```
    historico.push(paciente);
    console.log("Atendido:", paciente);
}

function desfazerUltimoAtendimento() {
    if (historico.length === 0) return console.log("Nada a desfazer");
    let paciente = historico.pop();
    filaNormal.enqueue(paciente); // volta pra fila normal
    console.log(`Desfeito! ${paciente} voltou para a fila normal.`);
}

// Simulação
adicionarPaciente("Carlos");
adicionarPaciente("Dona Maria", true);
adicionarPaciente("Ana");
adicionarPaciente("João", true);

atenderPaciente(); // Maria
atenderPaciente(); // João
atenderPaciente(); // Carlos

desfazerUltimoAtendimento(); // Carlos volta
```

2) Um jogo armazena as fases em uma LinkedList que representa a sequência de progressão. O jogador pode avançar e também voltar fases (como um botão "voltar" no jogo). Sempre que volta, a fase atual é armazenada em uma pilha de fases anteriores.
   Tarefas:
   a) Crie uma LinkedList com 5 fases do jogo.
   b) Simule a progressão pelas fases.
   c) Ao voltar de fase, mova a fase atual para a pilha.
   d) Permita restaurar uma fase anterior da pilha de volta à lista.

```
const LinkedList = require('../../Aula 6 - LinkedList/LinkedList.js');
let fases = new LinkedList();
let fasesAnteriores = []; // pilha para guardar nós

// Criando fases do jogo
["Fase 1", "Fase 2", "Fase 3", "Fase 4", "Fase 5"].forEach(f =>
fases.insertAtEnd(f));
```

```
// Simula o jogador avançando nas fases
let atual = fases.head;
console.log("Fase atual:", atual.value);

// Avançar até a Fase 3
for (let i = 0; i < 2; i++) {
    fasesAnteriores.push(atual); // guarda a fase atual antes de
avançar
    atual = atual.next;
    console.log("Avançou para:", atual.value);
}

// Jogador decide voltar uma fase
let faseAnterior = fasesAnteriores.pop();
atual = faseAnterior;
console.log("Voltou para:", atual.value);

// Jogador quer restaurar a fase seguinte
if (atual.next !== null) {
    // salva a atual antes de avançar de novo
    fasesAnteriores.push(atual);
    atual = atual.next;
    console.log("Restaurou fase:", atual.value);
}
```

3) Você deve rastrear encomendas em trânsito por diversas cidades. Cada cidade é um nó de uma DoublyLinkedList, indicando o caminho (ida e volta). Novas encomendas entram em uma fila de despacho, aguardando para serem enviadas pela rota.
   Tarefas:
   a) Monte uma DoublyLinkedList com cidades: "SP", "RJ", "BH", "BA".
   b) Adicione 3 encomendas em uma fila.
   c) Simule o envio das encomendas pelas cidades (percorrendo a lista).
   d) Ao chegar ao fim, percorra de volta até SP.

```
const DoublyLinkedList = require('../../Aula 7 -
DoubleLinkedList/DoublyLinkedList - Completa.js');
const Fila = require('../../Aula 5 - Fila/Fila.js');

let rota = new DoublyLinkedList();
```

```
let encomendas = new Fila();

// Criando rota
["São Paulo", "Rio de Janeiro", "Belo Horizonte",
"Salvador"].forEach(cidade => rota.append(cidade));

// Enfileirando encomendas
encomendas.enqueue("Encomenda A");
encomendas.enqueue("Encomenda B");
encomendas.enqueue("Encomenda C");

function rastrearEncomenda(encomenda) {
    console.log(`Rastreando ${encomenda}:`);

    // Ida
    let cidade = rota.head;
    while (cidade) {
        console.log(`-> ${cidade.value}`);
        cidade = cidade.next;
    }

    // Volta
    cidade = rota.tail.prev; // já passou pela última
    while (cidade) {
        console.log(`<- ${cidade.value}`);
        cidade = cidade.prev;
    }

    console.log("Entrega finalizada!\n");
}

// Simulação do envio
while (!encomendas.isEmpty()) {
    let encomenda = encomendas.dequeue();
    rastrearEncomenda(encomenda);
}
```