Atividade Prática: A Trilha do Herói na Masmorra

Atividade Prática: A Trilha do Herói na Masmorra

1. Introdução

Bem-vindo, aventureiro! No mundo dos jogos, mapas e caminhos são constantemente criados e modificados. Nesta atividade, vamos usar listas encadeadas para modelar algo dinâmico: o mapa de uma masmorra. Cada sala será um nó na nossa lista, e o caminho do herói será a travessia do início ao fim.

Seu objetivo é implementar funcionalidades que permitam **modificar e combinar** esses caminhos, simulando a descoberta de novas áreas e a alteração do ambiente do jogo.

2. Objetivos de Aprendizagem

Ao final desta atividade, você será capaz de:

- Modelar um caminho linear (uma trilha) usando uma lista simplesmente encadeada.
- Implementar um algoritmo de **fusão de listas** (concatenação).
- Praticar a **inserção de nós no meio** de uma lista existente.
- Implementar um algoritmo para **remover múltiplos nós** que atendem a uma condição específica.

3. O Cenário

Você receberá o arquivo `Masmorra.java`. Ele define a estrutura de uma `Masmorra`.

- Cada **nó** da lista é um `SalaNode` que possui um `tipo` ("monstro", "tesouro", "vazia") e um `valor` (dano do monstro, quantidade de ouro, etc.).
- A classe `Masmorra` já possui métodos para `adicionarSala` no final e para `explorarMasmorra`, que narra a jornada do herói.

Sua missão é dar vida a essa masmorra, implementando as funções que a tornam um lugar dinâmico e mutável.

4. Instruções e Desafios

No código-fonte fornecido, encontre a seção **"DESAFIO: IMPLEMENTAR ESTES MÉTODOS"** e implemente a lógica para as seguintes funções:

Desafio 1: Conectando Caminhos - `fundirMasmorras(Masmorra outraMasmorra)`

Um terremoto abriu uma nova passagem! Esta função deve conectar uma `outraMasmorra` ao final da masmorra atual, criando um caminho único e mais longo.

- Você deve percorrer a masmorra principal até encontrar sua última sala.
- A referência `proximo` dessa última sala deve então apontar para a `entrada` da `outraMasmorra`.

- **Ponto Crítico:** Após a fusão, a `outraMasmorra` original deve ser "esvaziada" (sua `entrada` deve ser definida como `null`). Isso é crucial para garantir que a lista encadeada não seja gerenciada por dois objetos diferentes, o que causaria comportamentos inesperados.

Desafio 2: Criando Passagens Secretas - `inserirCaminhoSecreto(...)`

O herói encontrou um mapa! A função deve criar e inserir uma nova sala imediatamente **após a primeira ocorrência** de uma sala com um tipo específico.

- **Exemplo:** `inserirCaminhoSecreto("vazia", "tesouro", 100)` deve encontrar a primeira sala vazia e inserir uma sala de tesouro valendo 100 moedas logo depois dela.
- Se não houver nenhuma sala do tipo especificado, nenhuma inserção deve ocorrer.

Desafio 3: Limpando a Área - `removerSalasPerigosas(int danoMaximo)`

O herói ficou mais forte e decidiu limpar a masmorra de ameaças muito grandes. Esta função deve percorrer toda a lista e remover todas as salas do tipo "monstro" cujo `valor` (dano) seja **maior** que o `danoMaximo` permitido.

- **Atenção aos casos de borda:**
  - A primeira sala (`entrada`) é um monstro perigoso e precisa ser removida.
  - Duas ou mais salas de monstros perigosos estão em sequência. Seu código deve ser capaz de remover todas elas corretamente.