Relatório/Apresentação do jogo Termo

Alunos: Gabriel Antunes Bernardi

Yuri Martins Teixeira

Carlos Eduardo de Almeida Vieira

Regras:

- Seguir as mesmas regras do Jogo Termo.
- Utilizar a linguagem Java
- Utilizar estruturas de dados em memória
- Não é necessário GUI

Apresentação

Decisões tomadas:

1. **Escolha das palavras:**
    - **Decisão**: Selecionar palavras comuns de 5 letras, abrangendo categorias como animais, transportes, cores, países e objetos.
    - **Justificativa**: Palavras de 5 letras equilibram complexidade e acessibilidade, tornando o jogo desafiador, mas não excessivamente difícil. A variedade de categorias mantém o jogo interessante e educativo.
2. **Mecânica do Jogo:**
    - **Decisão**: Implementar um jogo de adivinhação de palavras, inspirando no jogo “Termo”, onde o jogador tem um número limitado de tentativas para adivinhar a palavra secreta.

- **Justificativa**: Esta mecânica é simples, mas envolvente, e testa as habilidades de dedução e vocabulário do jogador.

3. **Estrutura de Dados - Array:**
    - **Decisão**: Usar um array para armazenar as palavras possíveis.
    - **Justificativa**: Arrays oferecem acesso rápido e eficiente (O(1)) para selecionar uma palavra aleatória, e são uma estrutura de dados simples e adequada para armazenar uma coleção fixa de elementos.
4. **Geração Aleatória de Palavras:**
    - **Decisão**: Utilizar a classe “Random” para escolher uma palavra aleatória do array a cada jogo.
    - **Justificativa**: A escolha aleatória aumenta a rejogabilidade e mantém cada jogo único.
5. **Interface do Usuário:**
    - **Decisão**: Utilizar a entrada padrão do console para a interação do usuário e “JOptionPane” para a mensagem de vitória.
    - **Justificativa**: A entrada do console é uma maneira simples e eficaz de interagir com o jogador em um programa Java. “JOpitionPane” fornece uma maneira simples de criar uma interface gráfica para a mensagem de vitória sem a necessidade de desenvolver uma GUI completa.
6. **Processamento de Entrada e Feedback:**
    - **Decisão**: Comparar a entrada do usuário com a palavra secreta, fornecendo feedback sobre letras corretas na posição correta e na posição errada.
    - **Justificativa**: Este feedback ajuda o jogador a refinar seus palpites em tentativas subsequentes, tornando o jogo interativo e educativo.
7. **Limpeza do Console**
    - **Decisão**: Implementar uma funcionalidade para “limpar” o console(imprimindo linhas em branco) antes de reiniciar o jogo
    - **Justificativa**: Limpar o console melhora a experiência do usuário ao reiniciar o jogo, mantendo a interface limpa e organizada.

8. **Limitação de Tentativas**
    - **Decisão**: Definir um número máximo de tentativas para adivinhar a palavra.
    - **Justificativa**: Um limite de tentativas aumenta a tensão e o desafio, incentivando o jogador a pensar cuidadosamente sobre cada palpite.

Justificar as escolhas de estruturas de dados/algoritmos de acordo com a notação BigO

1. **Array para Armazenar Palavras(Palavras):**
    - **Complexidade de Tempo**: O(1) para acessar uma palavra aleatória.
    - **Justificativa**: Um array fornece acesso direto a qualquer elemento em tempo constante. A escolha de uma palavra aleatória, portanto, não depende do tamanho do array.
2. **Geração Aleatória de Palavras:**
    - **Complexidade de Tempo**: O(1)
    - **Justificativa**: A geração de um índice aleatório e o acesso a uma palavra no array são operações de tempo constante. A complexidade não aumenta com o número de palavras no array.
3. **String para Construção de Dicas:**
    - **Complexidade de Tempo**: O(N) para construir a dica, onde N é o comprimento da palavra.
    - **Justificativa**: O algoritmo percorre cada letra da palavra uma vez para construir a dica, resultando em uma complexidade linear. Cada operação de append em “**StringBuilder**” é O(1) na maioria dos casos, exceto quando é necessário redimensionar o buffer interno.
4. **Processamento de Entrada do Usuário:**
    - **Complexidade de Tempo**: O(N) para cada palpite, onde N é o comprimento da palavra.
    - **Justificativa**: Para cada palpite, o algoritmo compara cada letra do palpite com a palavra secreta, resultando em um complexidade linear em relação ao

comprimento da palavra.

5. **Estrutura de Loop Principal(Tentativas):**
    - **Complexidade de Tempo**: O(M*N), onde M é o numero de tentativas e N é o comprimento da palavra
    - **Justificativa**: Em cada tentativa, o algoritmo processa a entrada do usuário (O(N)). O número total de operações é, portanto proporcional ao número de tentativas e ao comprimento da palavra.
6. **Limpeza do Console:**
    - **Complexidade de Tempo**: O(1).
    - **Justificativa**: A “limpeza” do console é feita imprimindo um número fixo de linhas em branco, independentemente do estado anterior do console, portanto é uma operação de tempo constante