UNIVERSIDADE FEDERAL DE VIÇOSA - CAMPUS FLORESTAL

Trabalho Prático Programação Orientada a Objetos
Jogo da Forca

Nome: Aline Cristina Santos Silva
Luana Tavares Anselmo
Gustavo Luca Ribeiro da Silva
Guilherme Gustavo Da cunha
Gabriel Henrique Bramante
Gabriel Lázaro G. Marques

Matrícula: 5791,5364,5797,5378,5794,5800

Florestal - MG
2025

1. Introdução

O presente documento tem como objetivo descrever o desenvolvimento, funcionamento e características principais do **Jogo da Forca**, um projeto criado para proporcionar entretenimento, ao mesmo tempo que desafia as habilidades de raciocínio lógico e vocabulário dos jogadores.

A implementação do jogo foi realizada utilizando as seguintes ferramentas e tecnologias:

- **NetBeans 8.2** como ambiente de desenvolvimento integrado (IDE);
- **JavaFX** para a criação da interface gráfica interativa;
- **Scene Builder**, utilizado como componente visual para facilitar a construção e a personalização das telas do jogo.

Este projeto busca oferecer uma experiência educativa e divertida, com uma interface intuitiva e um design atraente, adaptado ao ambiente digital. O jogador deverá adivinhar uma palavra com base em letras fornecidas, dentro de um limite de tentativas.

2. Organização

O projeto foi estruturado de forma modular e organizada, facilitando a manutenção, a extensão futura e a compreensão por parte dos desenvolvedores. Ele está dividido em cinco pacotes principais, cada um com uma função específica dentro da aplicação. A seguir, detalhamos cada um deles:

- **controller:** Este pacote contém os controladores responsáveis pela lógica das interações entre os usuários e as telas da aplicação.
    - Cada controlador está vinculado a uma tela específica da interface, proporcionando a ponte entre os elementos visuais (definidos nos arquivos XML) e a lógica do jogo.
    - Os controladores também gerenciam eventos como cliques em botões, digitação de letras e alterações nos estados do jogo.
    - A separação clara entre visual (View) e controle (Controller) segue o padrão de arquitetura MVC (Model-View-Controller), promovendo maior organização e flexibilidade no desenvolvimento.

- **Css**: Este pacote é responsável por todo o tratamento visual do jogo, garantindo que a interface do usuário tenha um design agradável e consistente.
    - Os arquivos neste pacote são escritos em CSS (Cascading Style Sheets) e definidos em conformidade com o padrão JavaFX.
    - Estilos como fontes, cores, espaçamentos, tamanhos de componentes e interações visuais estão organizados de maneira separada, permitindo alterações rápidas no layout sem impacto no código-fonte principal.
    - Exemplo: configuração de cores para os temas do jogo ou estilização de botões.

- **Imgs:** Este pacote armazena as imagens utilizadas nas telas do jogo, adicionando elementos gráficos que enriquecem a experiência do usuário.
    - Inclui ícones, figuras temáticas e quaisquer outros recursos visuais não textuais do jogo.
    - A centralização dos arquivos de imagens facilita a substituição e o gerenciamento dos recursos visuais, especialmente em atualizações do design.
    - Todas as imagens estão em formatos compatíveis (como PNG ou JPG), com resoluções otimizadas para desempenho e qualidade visual.

- **Temas:** Este pacote reúne os arquivos de texto (.txt) que contêm os temas utilizados no jogo.
    - Cada arquivo de texto possui uma lista de palavras relacionadas a um tema específico, garantindo que o jogo tenha um grande repertório para proporcionar maior diversão e desafio ao usuário.
    - Exemplo de temas: frutas, profissões, animais, entre outros.
    - A estrutura modular desse pacote permite a fácil adição de novos temas e palavras ao jogo, sem necessidade de modificar o código existente.

- **View:** Este pacote contém as telas do jogo, definidas em arquivos no formato XML, criados utilizando o Scene Builder.
    - As telas são responsáveis pela apresentação visual e interação com os usuários.
    - A separação entre a View e a lógica do jogo facilita o trabalho em equipe, permitindo que designers e desenvolvedores trabalhem de forma independente.

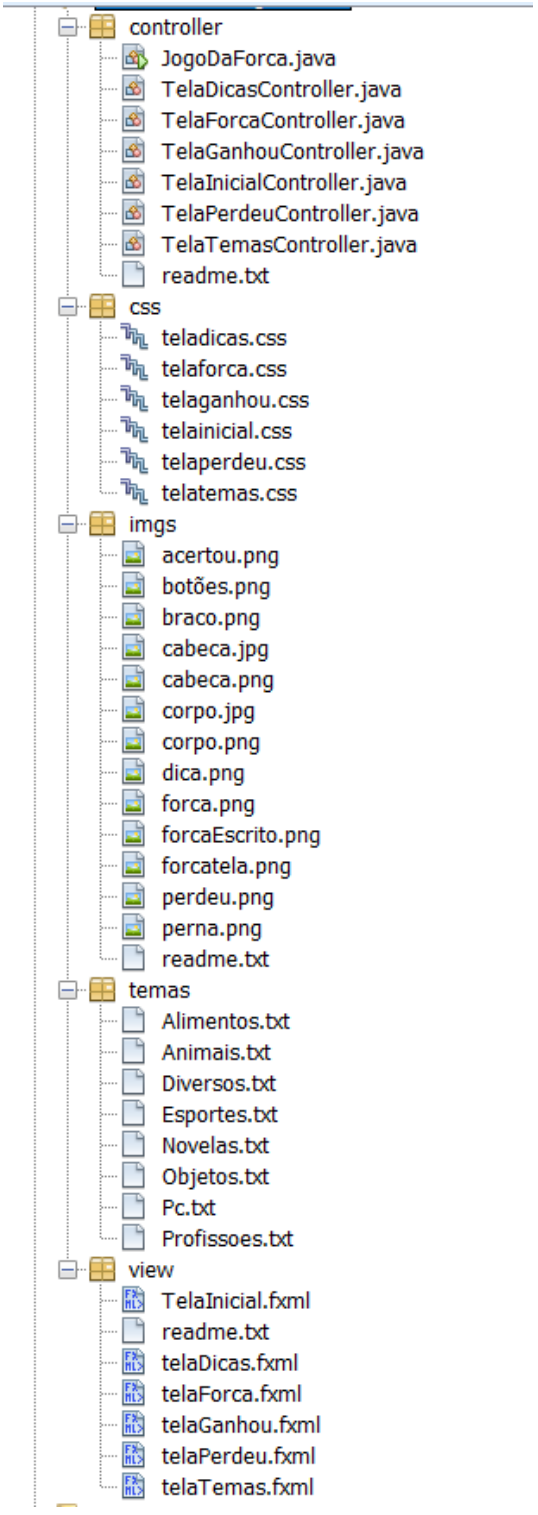

Figura 1: Organização Dos pacotes

3. Desenvolvimento

O projeto conta com diversas partes interconectadas para o funcionamento completo, mas algumas são essenciais por desempenharem papéis-chave na estrutura e execução. Abaixo, destacamos as seções e funcionalidades centrais do sistema:

1. Controle da Lógica de Temas e Dados (TelaTemasController)

- **Seleção de Temas**: O jogador pode escolher um ou mais temas por meio de caixas de seleção (CheckBox) na interface gráfica. Cada tema está associado a um arquivo .txt armazenado em um diretório específico (/temas), o que facilita a modificação e expansão do jogo no futuro.
- **Carregamento de Dados**: O controlador lê os arquivos de temas selecionados, carregando as palavras e suas respectivas dicas em uma estrutura Map. Essa associação (Map<String[], String>) garante que cada par de palavra e dica esteja vinculado ao tema correto.
- **Sorteio de Dados**: Após o carregamento, um par palavra/dica é sorteado aleatoriamente para garantir uma experiência única a cada jogo. O tema, a palavra e a dica são então repassados para a tela principal.

```
private List<String> obterTemasSelecionados() {
    List<String> temas = new ArrayList<>();
    if (cbAnimais.isSelected()) temas.add("Animais");
    if (cbAlimentos.isSelected()) temas.add("Alimentos");
    if (cbDiversos.isSelected()) temas.add("Diversos");
    if (cbEsportes.isSelected()) temas.add("Esportes");
    if (cbNovelas.isSelected()) temas.add("Novelas");
    if (cbObjetos.isSelected()) temas.add("Objetos");
    if (cbPC.isSelected()) temas.add("Pc");
    if (cbProfissoes.isSelected()) temas.add("Profissoes");
    return temas;
}

private Map<String[], String> carregarPalavrasDosTemas(List<String> temasSelecionados) {
    Map<String[], String> palavrasComTemas = new HashMap<>();

    for (String tema : temasSelecionados) {
        String caminhoArquivo = CAMINHO_TEMAS + tema + ".txt";
        try (BufferedReader br = new BufferedReader(new FileReader(caminhoArquivo))) {
            String nomeTema = br.readLine(); // Lê a primeira linha do arquivo (nome do tema)
            if (nomeTema == null) continue;

            palavrasComTemas.putAll(
                br.lines()
                  .map(linha -> linha.split(";"))
                  .filter(partes -> partes.length == 2)
                  .collect(Collectors.toMap(partes -> partes, partes -> nomeTema))
            );
        } catch (IOException e) {
            System.err.println("Erro ao carregar o arquivo do tema: " + tema);
            e.printStackTrace();
        }
    }
    return palavrasComTemas;
}
```

Figura 2: Funções para Selecionar Temas e Carregar Dados

```java
private void abrirTelaForca(String palavra, String dica, String tema) {
    try {
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/view/telaForca.fxml"));
        Parent root = loader.load();

        TelaForcaController forcaController = loader.getController();
        forcaController.inicializarPalavra(palavra, dica, tema);

        Stage stage = new Stage();
        stage.setTitle("Jogo da Forca");
        stage.setScene(new Scene(root));
        stage.show();

        Stage stageAtual = (Stage) btnIniciar.getScene().getWindow();
        stageAtual.close();
    } catch (IOException e) {
        System.err.println("Erro ao abrir a tela da Forca!");
        e.printStackTrace();
    }
}
```

Figura 3: Função para Sortear Dados

2. Tela Principal do Jogo (TelaForcaController)

- **Exibição Inicial**: Na tela principal, são configurados elementos gráficos essenciais para o jogo, como:
  - **Traços Representando a Palavra**: Inicialmente, a palavra é mostrada como uma sequência de traços, que vão sendo substituídos pelas letras acertadas.
  - **Erros**: Um contador de erros acompanha o progresso do jogador, e cada erro resulta na exibição de uma nova parte do boneco na forca.
  - **Botões do Alfabeto**: Esses botões permitem que o jogador escolha uma letra por vez. Após a escolha, o botão é desabilitado.
  - **Dica**: O botão de dica é inicialmente desativado e só se torna acessível quando o contador de erros atinge o valor de 5/6, oferecendo um suporte ao jogador em situações críticas.
- **Processamento do Jogo**:
  - A validação de letras é feita ao clicar em qualquer botão do alfabeto, atualizando os traços ou o contador de erros, conforme necessário.
  - Quando o jogador acerta todas as letras ou atinge o limite de erros, uma nova tela (de vitória ou derrota) é exibida.

Como as funções da tela principal são extensas, optamos por explicar apenas a lógica de funcionamento dessa tela, evitando sobrecarregar a documentação com excesso de imagens.

3. Integração e Modularidade

As informações são cuidadosamente repassadas entre as telas usando métodos que permitem modularidade e clareza na separação de responsabilidades. Por exemplo:

- O método **inicializarPalavra**, no controlador da tela principal, recebe os dados da palavra, dica e tema da tela anterior e configura todos os elementos visuais de acordo.
- O uso de estruturas como o Map assegura uma organização clara e eficiente dos dados, simplificando a associação entre palavras, dicas e temas.

4. Resultados

Para facilitar a visualização e compreensão dos resultados, optamos por produzir um vídeo demonstrativo onde o jogo é testado em funcionamento. Acreditamos que, dessa forma, será possível observar todas as funcionalidades implementadas de maneira clara e objetiva. O vídeo inclui a navegação pelas telas, a interação com os botões, a dinâmica do jogo da forca, e as ações correspondentes aos acertos e erros. Segue abaixo o link para acessar o vídeo:

https://youtu.be/362sBdnwerE (Sugerimos assistir na velocidade 0.8x para melhor visualização)

5. Conclusão

O desenvolvimento deste projeto de jogo da forca proporcionou um aprendizado valioso no contexto de programação orientada a objetos, design de interfaces gráficas e integração de componentes diversos. Ao utilizar **JavaFX** para construir a interface gráfica e a interação visual do usuário, aliado ao **Scene Builder** para o design intuitivo das telas, conseguimos desenvolver uma aplicação funcional, visualmente agradável e bem estruturada.

A organização do projeto em pacotes como `controller`, `css`, `imgs`, `temas` e `view` favoreceu a modularidade e a manutenção do código, possibilitando maior clareza e reusabilidade dos componentes implementados. Cada funcionalidade foi cuidadosamente projetada, desde a escolha dos temas, o sorteio aleatório das palavras e dicas, até o acompanhamento dinâmico dos erros e acertos na tela principal.

Um dos aspectos mais destacados foi o trabalho com a passagem de dados entre as telas, incluindo o uso de mapas e coleções para gerenciar palavras e temas, o que proporcionou flexibilidade na lógica de carregamento e interação entre os elementos do jogo. Além disso, a criação de recursos como a exibição da dica condicionada ao número de erros e o desenho progressivo do boneco foram implementações criativas que enriqueceram a experiência do jogador.

O projeto demonstra uma solução robusta para um jogo clássico, trazendo modernidade e adequações às exigências de um aplicativo desktop. Apesar dos desafios enfrentados, como a configuração de caminhos para arquivos externos e o controle lógico entre as telas, todos foram superados com foco e aprimoramento contínuo, resultando em um produto funcional e atrativo.

Por fim, é importante destacar que este jogo foi projetado com fins educacionais, mostrando a integração de conceitos de desenvolvimento de software, lógica de programação e design de interfaces. Esperamos que essa aplicação inspire e agregue valor tanto ao aprendizado quanto ao entretenimento de quem interagir com ela.

6. Referências

I. Playlist no Youtube sobre JavaFX, Scene Builder e Netbeans (basicamente todos os integrantes do grupo assistiram suas aulas) : https://www.youtube.com/watch?v=DpyVnys1nvs

II. https://docs.oracle.com/javafx/scenebuilder/1/use_java_ides/sb-with-nb.htm