UNIVERSIDADE FEDERAL DE OURO PRETO
DEPARTAMENTO DE COMPUTAÇÃO - DECOM
Professor:

Guilhermo Camara Chavéz

RELATÓRIO PRÁTICO
Disciplina: BCC221 Turma: 11

Aluno:
Carlos Henrique Nunes Souza

Morro do Cruzeiro
2024

Sumário

Introdução

Este trabalho tem como objetivo a implementação de um jogo da forca, desenvolvido na linguagem de programação Java. O programa foi projetado para proporcionar uma experiência interativa e envolvente, desafiando os jogadores a adivinhar uma palavra secreta por meio de tentativas de letras.

Como descrito nas aulas, o projeto adota uma abordagem de desenvolvimento orientada a objetos, visando a criação de um código com alta coesão e baixo acoplamento. Para isso, foram explorados conceitos como encapsulamento e polimorfismo, assegurando que as diversas funcionalidades fossem organizadas em classes bem definidas, de maneira modular e reutilizável.

O programa foi dividido em cinco classes, sendo uma responsável pela interface gráfica, enquanto as demais gerenciam a lógica do jogo, a seleção de palavras, o histórico de tentativas e a representação do estado atual da forca. Através deste jogo, o programa busca oferecer uma maneira divertida de aprender e praticar o vocabulário, enquanto explora os princípios fundamentais da programação orientada a objetos.

Através deste sistema, o programa busca otimizar o fluxo de trabalho na oficina, garantindo maior controle sobre as operações diárias, desde a recepção do cliente até a execução e fechamento dos serviços prestados. Tudo isso usando conceitos de programação orientada a objetos.

A implementação do trabalho foi feita usando a **IDE Visual Studio Code da Microsoft.**
O Sistema operacional utilizado foi o **Windows 11 Home Single Language Versão 23H2.**

Como usar o programa:

Para compilar e executar o jogo da forca, é necessário ter o Java Development Kit (JDK) instalado no seu sistema. Após a instalação, siga os passos abaixo:

Compilação: Navegue até a pasta onde os arquivos do projeto estão localizados. Utilize o terminal ou prompt de comando e execute o seguinte comando para compilar os arquivos Java:

javac -d bin src/*.java

Este comando irá compilar todos os arquivos **.java** na pasta **src** e colocar os arquivos **.class** resultantes na pasta **bin**.

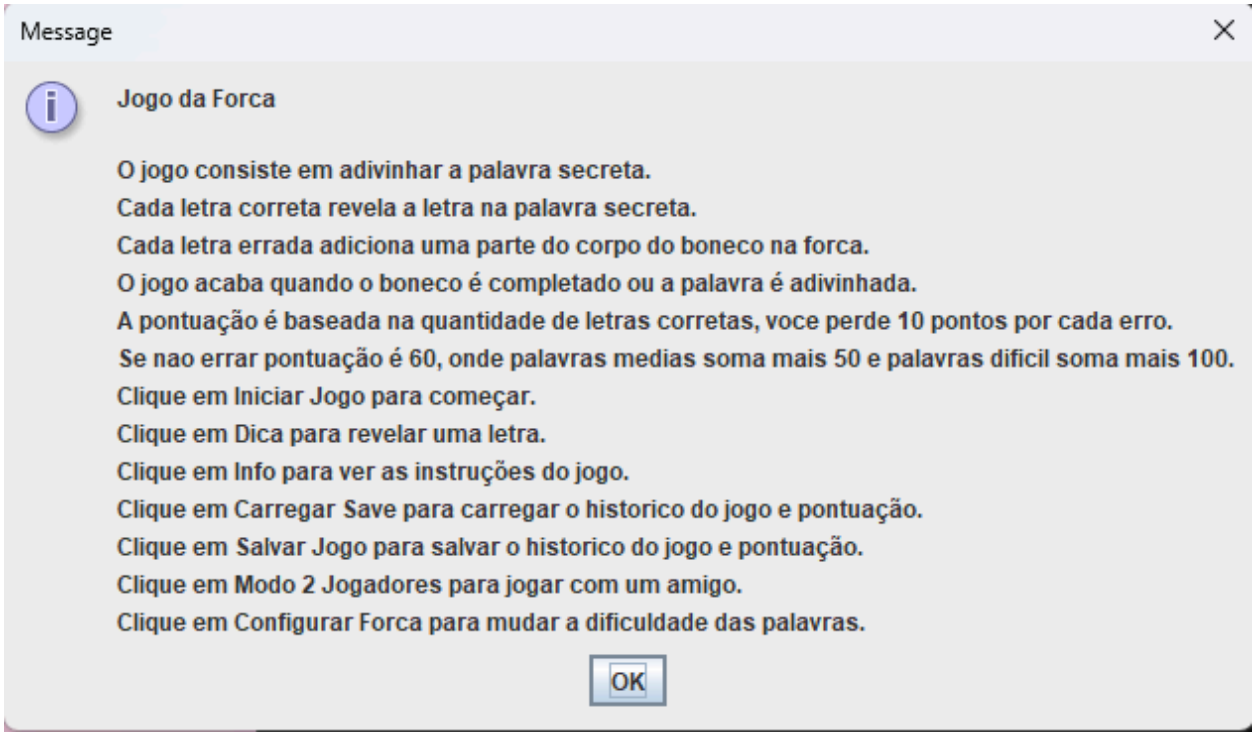
Execução: Para executar o programa, acesse a pasta **bin** e execute o seguinte comando:

java app.Main

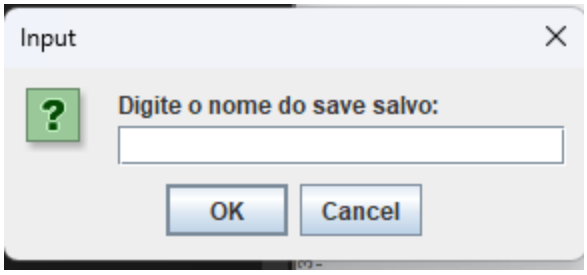
Interface Gráfica: Após a execução, o programa exibirá a seguinte interface gráfica do jogo da forca, onde você poderá começar a jogar.

Onde o usuário, visualiza uma inicial com diversos botoes.

No botão INFO , ao clicar, irá exibir uma tela popup onde contém as informações sobre o jogo, o que cada botão faz. segue o exemplo.

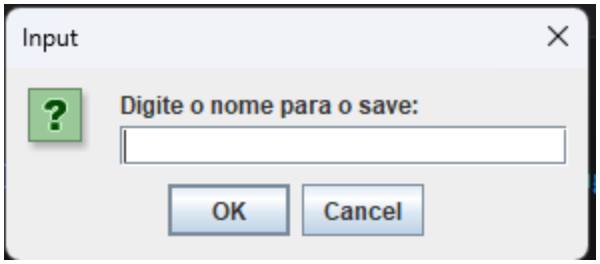


No botão CARREGAR SAVE, ao clicar abrirá um popup onde o usuário pode digitar o nome do SAVE, lembrando que o safe deve estar na pasta SRC do jogo, o usuário pode **carregar um save** de um jogo que ele já estava jogando, onde recupera o **histórico de derrotas e vitórias e a pontuação** que ele tinha conseguido até salvar.

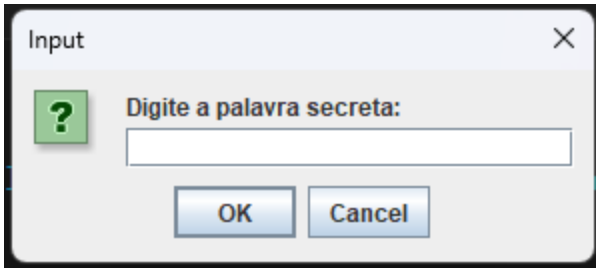


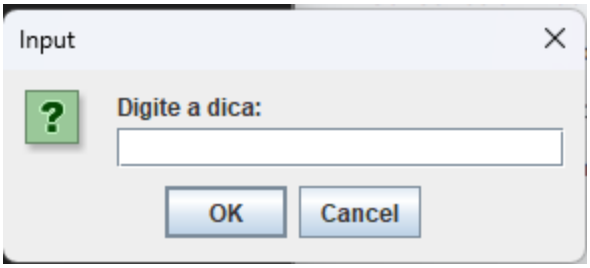
No botão SALVAR JOGO, ao clicar abrirá um popup onde o usuário pode digitar o nome do SAVE, lembrando que o safe irá estar na pasta SRC do jogo, o usuário pode **salvar** um jogo que ele já está jogando, onde salva o **histórico de derrotas e vitórias e a pontuação** que ele

tinha conseguido até o momento.

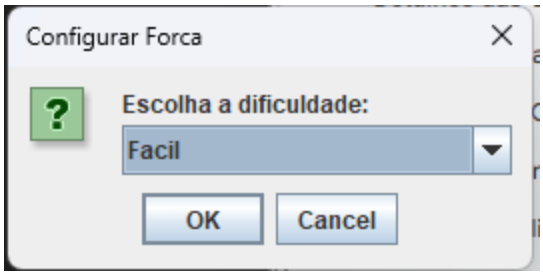


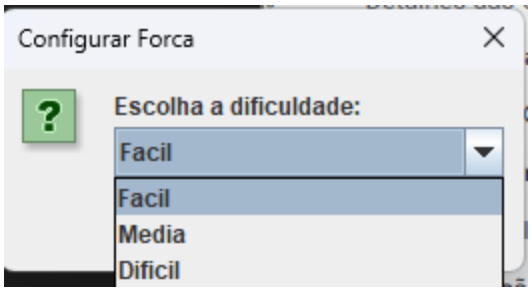
No botão MODO 2 JOGADORES, ao clicar abrirá um popup onde o usuário pode digitar a palavra secreta, seguido da dica para o amigo adivinhar, nesse modo ele digita a palavra e o amigo adivinha.





No botão CONFIGURAR FORCA , ao clicar abrirá um popup onde o usuário pode escolher em um menu interativo qual nível de dificuldade das palavras, dentre as opções (Facil, Medio, Dificil)





No botão INICIAR JOGO, ao clicar abrirá a tela atualiza dando inicio o jogo

onde o usuário visualiza a dica e clica em uma letra para verificar se existe, caso exista a letra

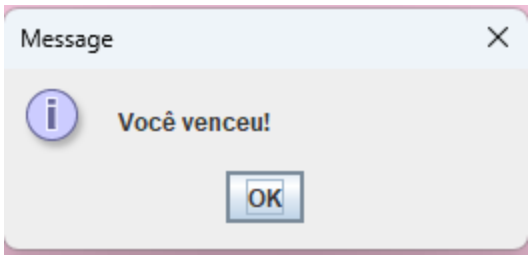
é revelada , e a letra é desabilitada, para saber quais foram as clicadas

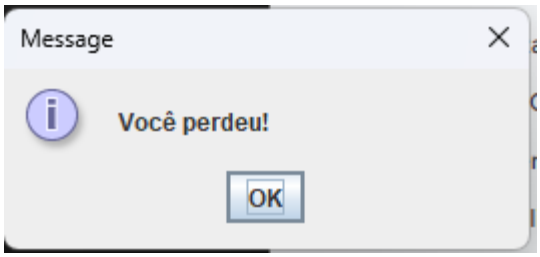
caso o usuário erre, e iniciado uma parte da forca que são dividas em 6 partes, CABEÇA, TRONCO, BRAÇO ESQUERDO, BRAÇO DIREITO, PERNA ESQUERDA e PERNA DIREITA, sendo 6 tentativas, onde a cada erro mostra uma parte nova, e diminui as TENTATIVAS RESTANTES

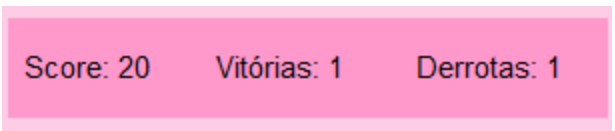
caso o usuário tenha muita dúvida ela pode clicar no botão DICA que revela uma letra aleatória

como forma de ajudar o usuário a adivinhar a palavra,

caso o usuário vença, o jogo vai mostrar um pop up mostrando uma mensagem de vitória ou caso perca uma de derrota, e ao mesmo tempo atualizando o SCORE , o número de VITÓRIAs, e o número de DERROTAS e se foi derrota completa o desenho.







após o fim do jogo, o usuário escolhe sair ou continuar clicando novamente em NOVO JOGO ou MODO 2 JOGADORES.

Desafios enfrentados:

Durante o desenvolvimento do jogo da forca, diversos desafios foram enfrentados::

1. **Integração de Funcionalidades:** Um dos principais desafios foi garantir que todas as funcionalidades do jogo operassem de forma coesa. Com elementos diversos, como a lógica do jogo, o gerenciamento de palavras e o histórico de partidas, era essencial assegurar que as interações entre essas partes fossem consistentes e que o fluxo de dados fosse gerido adequadamente.
2. **Manutenção de Dados e Coerência:** A integridade dos dados foi uma preocupação constante durante a implementação de operações como adicionar, remover e atualizar palavras e saves. O sistema precisava assegurar que todas as ações de criação, leitura, atualização e deleção fossem realizadas de maneira correta, evitando inconsistências nas informações do jogo e no histórico das partidas.
3. **Interface do Usuário:** O desenvolvimento de uma interface intuitiva e acessível para todos os tipos de jogadores apresentou desafios significativos. Era crucial que os menus, botões e telas fossem claros e funcionais, demandando uma série de testes e ajustes para aprimorar a experiência do usuário e garantir uma navegação fluida.
4. **Gerenciamento de Estados do Jogo:** Manter o controle do estado do jogo, como as tentativas restantes, as letras já adivinhadas e as palavras a serem adivinhadas, foi um desafio importante. A implementação de uma lógica robusta para gerenciar esses estados, assim como a recuperação de dados a partir de saves anteriores, foi essencial para proporcionar uma experiência de jogo continua e envolvente.

Detalhes da implementação:

A implementação do sistema foi baseada em uma arquitetura orientada a objetos, utilizando diversas classes para modelar a lógica do jogo da Forca, suas regras e interação com o usuário por meio de uma interface gráfica. Cada classe desempenha um papel específico na execução do jogo, representando diferentes entidades e funcionalidades necessárias para o funcionamento do sistema.

Estrutura de Classes:

1. **App**: A classe App é a responsável por iniciar o jogo. Ela contém o método principal main, que cria uma instância da classe Jogo e da **InterfaceGrafica** e inicia a aplicação. Ela atua como o ponto de entrada do sistema, conectando a lógica do jogo com a interface do usuário.
2. **Forca**: Esta classe modela a lógica central do jogo da forca. Ela gerencia o estado do jogo, como a palavra sendo adivinhada, as tentativas restantes, e os erros cometidos pelo jogador. Métodos como ***adicionarErro*** e ***verificarFimDeJogo*** são utilizados para atualizar o status do jogo e verificar se o jogador ganhou ou perdeu. A classe também cuida da exibição da palavra parcialmente adivinhada e das letras que o jogador já tentou.
3. **Historico**: A classe ***Historico*** é responsável por armazenar as partidas jogadas. Ela registra as palavras utilizadas, o status final da partida (vitória ou derrota), e a quantidade de tentativas usadas. Métodos para adicionar partidas e consultar o histórico permitem a coleta e análise de dados sobre o desempenho do jogador ao longo do tempo.
4. **InterfaceGrafica**: A classe ***InterfaceGrafica*** lida com a interação do jogador com o jogo por meio de uma interface gráfica. Ela cria e organiza os paineis e botões que o usuário usa para jogar, como o painel da forca, o painel de letras e o painel de tentativas. A interface gráfica também exibe a palavra a ser adivinhada e atualiza o número de tentativas restantes. Além disso, eventos de clique nos botões são configurados para capturar as jogadas do usuário e refletir as mudanças no estado do jogo.
5. **Jogo**: A classe ***Jogo*** centraliza a lógica do jogo da forca. Ela mantém o controle sobre a palavra atual, o número de tentativas, e os erros cometidos pelo jogador. Métodos como ***iniciarJogo*** e ***reiniciarJogo*** são responsáveis por preparar uma nova rodada, escolher uma nova palavra e reiniciar o estado da partida. A classe também colabora com a interface gráfica para atualizar o estado visual do jogo, como a exibição das letras já adivinhadas e o progresso no desenho da forca.
6. **Palavras**: Esta classe armazena o conjunto de palavras que podem ser utilizadas no jogo. Ela é responsável por fornecer uma palavra aleatória para cada nova rodada, garantindo que o jogador tenha uma experiência variada. Métodos como ***getPalavraAleatoria*** são implementados para buscar e retornar uma palavra da lista disponível.

Detalhes das Classes:

App:

A classe ***App*** é o ponto de entrada do jogo da forca, responsável por inicializar a interface gráfica do jogo. O método principal ***main*** cria uma instância da classe ***InterfaceGrafica*** e a torna visível, permitindo que o jogador interaja com o jogo. O uso de ***throws Exception*** garante que eventuais erros na execução sejam tratados adequadamente.

```java
public class App {

    public static void main(String[] args) throws Exception {
        InterfaceGrafica interfaceGrafica = new InterfaceGrafica();
        interfaceGrafica.setVisible(true);
    }
}
```

Forca

A classe ***Forca*** gerencia a construção da imagem da forca com base nos erros cometidos pelo jogador. Ela possui um ***array booleano*** forca com seis elementos, representando as partes da forca (cabeça, tronco, braços e pernas). No construtor, todos os elementos são inicializados como ***false***, indicando que nenhuma parte da forca foi desenhada.

Os métodos principais incluem:

- ***isEnforcado(int tentativasRestantes)***: verifica se o jogador foi enforcado, retornando true se não houver tentativas restantes.
- Métodos como ***getCabeca()***, ***getTronco()***, ***getBracoDireito()***, entre outros, retornam o estado de cada parte da forca.
- ***diminuirDesenho(int tentativasRestantes)***: atualiza o estado da forca, desenhando uma parte com base no número de tentativas restantes.
- ***resetarForca()***: reinicia o estado da forca para que todas as partes sejam desenhadas novamente.

```java

public class Forca {
    private Boolean[] forca = new Boolean[6];
    public Forca() {
    }
```

```
    public Boolean isEnforcado(int tentativasRestantes) {}

    public Boolean getPernaDireita() {}

    public Boolean getPernaEsquerda() {}

    public Boolean getBracoDireito() {}

    public Boolean getBracoEsquerdo() {}

    public Boolean getTronco() {}

    public Boolean getCabeca() {}

    public void diminuirDesenho(int tentativasRestantes) {}

    public void resetarForca() {}

}
```

Historico

A classe ***Historico*** gerencia o registro das partidas jogadas, armazenando informações sobre as ***palavras escolhidas, pontuações, vitórias e derrotas***. Ela possui quatro listas principais: ***indicesPalavrasEscolhidas***, ***pontuacoes***, ***contagemDeVitoriasLista***, e ***contagemDeDerrotasLista***, que armazenam os dados relacionados a cada partida.

Alguns dos métodos principais:

- **Adicionar** e **Setters**: métodos como ***adicionarIndicePalavraEscolhida(int indicePalavraEscolhida)*** e ***setPalavras(List<Palavras> palavras)*** permitem adicionar dados ao histórico ou modificar a lista de palavras.
- **Salvar** e **Carregar Jogo**: ***salvarGame(String NomeDoSave)*** grava o histórico em um arquivo texto, e ***carregaGame(String NomeDoSave)*** lê as informações do arquivo, restaurando o histórico da partida.

```java
public class Historico {
    private List<Palavras> palavras;
    private List<Integer> indicesPalavrasEscolhidas;
    private List<Integer> pontuacoes;
    private List<Integer> contagemDeVitoriasLista;
    private List<Integer> contagemDeDerrotasLista;

    public Historico(List<Palavras> palavras) {}

    public void setPalavras(List<Palavras> palavras) {}
    public List<Integer> getIndicesPalavrasEscolhidas() { return null; }
    public List<Integer> getPontuacoes() { return null; }
    public List<Integer> getContagemDeVitoriasLista() { return null; }
    public List<Integer> getContagemDeDerrotasLista() { return null; }
    public void setIndicesPalavrasEscolhidas(List<Integer>
indicesPalavrasEscolhidas) {}
    public void setPontuacoes(List<Integer> pontuacoes) {}
    public void setContagemDeVitoriasLista(List<Integer>
contagemDeVitoriasLista) {}
    public void setContagemDeDerrotasLista(List<Integer>
contagemDeDerrotasLista) {}
    public void adicionarIndicePalavraEscolhida(int indicePalavraEscolhida) {}
    public void adicionarPontuacao(int pontuacao) {}
    public void adicionarContagemDeVitorias(int contagemDeVitorias) {}
    public void adicionarContagemDeDerrotas(int contagemDeDerrotas) {}
    public void limparHistorico() {}
    public void exibirHistorico() {}
    public void salvarGame(String NomeDoSave) {}
    public void carregaGame(String NomeDoSave) {}
}
```

InterfaceGráfica:

A classe ***InterfaceGrafica*** cria a interface gráfica para um jogo da forca, utilizando ***Java Swing***. Ela configura a janela do jogo, exibe a forca, pontuação, letras e botões de controle. Alguns métodos principais incluem:

- **iniciarJogo()**: Inicia ou reinicia o jogo.
- **criarDesenhoForca()**: Atualiza o desenho da forca com base nas tentativas erradas.
- **exibirPalavraAtual()**: Mostra a palavra com as letras adivinhadas até o momento.
- **setTentativasRestantes()**: Exibe o número de tentativas restantes.
- **reiniciarBotoes()**: Habilita novamente todos os botões de letras após reiniciar o jogo.

```
Java
public class InterfaceGrafica extends JFrame {
    //Atributos principais
    private Jogo jogo;
    private JPanel painelPrincipal;
    private JPanel painelPalavra;
    private JPanel painelDica;
    private JLabel pernaDireita;
    private JLabel pernaEsquerda;
    private JLabel bracoDireito;
    private JLabel bracoEsquerdo;
    private JLabel tronco;
    private JLabel cabeca;
    private JPanel painelTentativas;
    private JPanel painelBotaoLetras;
    private JPanel painelScore;
    private JLabel imagemForca;

    //Construtor
    public InterfaceGrafica();

    //Métodos principais
    private void iniciarJogo();
    private void criarDesenhoForca();
    private void exibirPalavraAtual(String palavraAtual);
    private void setTentativasRestantes(int tentativas);
    private void reiniciarBotoes();

    //Outros métodos e eventos associados aos botões
    private void configurarEventos();
       ...
}
```

Jogo

A classe ***Jogo*** controla a lógica de um jogo, como a escolha de palavras, contagem de tentativas, pontuação e histórico. Alguns métodos principais:

- ***iniciarJogo()*:** Reseta o jogo e seleciona uma nova palavra aleatória.
- ***escolherPalavra()*:** Escolhe uma palavra aleatoriamente da lista.
- ***tentativa()*:** Verifica se a letra está na palavra e atualiza a palavra ou diminui as tentativas.
- ***revelarLetra()*:** Revela uma letra aleatória da palavra secreta.
- ***venceu()*:** Verifica se o jogador completou a palavra e atualiza a pontuação.
- ***perdeu():*** Verifica se o jogador perdeu.

```java
public class Jogo {

    public Jogo();

    public Historico getHistorico();

    public void setModo2Jogadores(Boolean modo2Jogadores);

    public void setDificuldade(String dificuldade);

    public void iniciarJogo();

    public void iniciarJogoModo2Jogadores(Palavras palavra);

    public void salvaHistoricoNasListas();

    public Boolean jogoIniciado();

    public int getPontuacao();

    public void setPontuacao(int pontuacaoNova);

    public int getContagemDeVitorias();

    public void setContagemDeVitorias(int contagemDeVitorias);

    public int getContagemDeDerrotas();
       ...
    }
```

Palavras

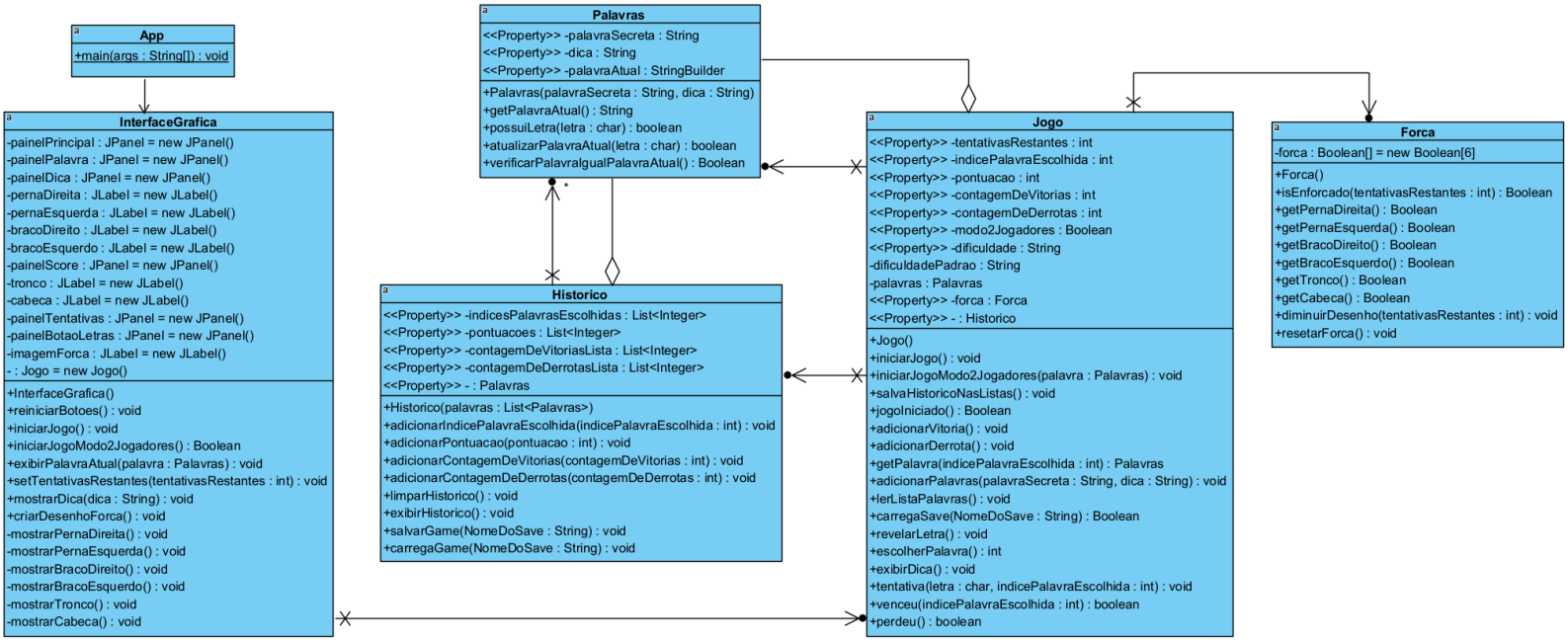
A classe ***Palavras*** representa uma palavra secreta no jogo e sua lógica de manipulação. Ela contém a palavra secreta, uma dica para o jogador e a versão atual da palavra que está sendo revelada ao longo do jogo.

Principais métodos:

- ***Construtor Palavras(String palavraSecreta, String dica)***: Inicializa a palavra secreta e a dica, além de criar uma string ***(palavraAtual)*** que começa com ***"_"*** para cada letra da palavra secreta.
- ***getPalavraSecreta():*** Retorna a palavra secreta.
- ***getDica():*** Retorna a dica da palavra.
- ***getPalavraAtual():*** Retorna a versão atual da palavra, com letras reveladas e ***"_"*** nas posições ainda não descobertas.
- ***possuiLetra(char letra):*** Verifica se a letra passada está na palavra secreta, revelando-a em ***palavraAtual*** se estiver.
- ***atualizarPalavraAtual(char letra):*** Atualiza a ***palavraAtual*** com uma letra, caso ela esteja na palavra secreta.
- ***verificarPalavraIgualPalavraAtual():*** Verifica se a palavra secreta foi completamente revelada comparando-a com a ***palavraAtual***.

```java
public class Palavras {

    public Palavras(String palavraSecreta, String dica);

    public String getPalavraSecreta();

    public String getDica();

    public String getPalavraAtual();

    public void setPalavraAtual(StringBuilder palavraAtual);

    public boolean possuiLetra(char letra);

    public boolean atualizarPalavraAtual(char letra);

    public Boolean verificarPalavraIgualPalavraAtual();
}
```

Criei um modelo UML utilizando o programa Astah UML, mas o arquivo resultante acabou ficando muito extenso. Para melhor ilustrar as relações entre as classes e os métodos de cada uma, incluí o arquivo UMLForca.png na pasta “docs/uml” do meu projeto.



Conceitos utilizados

No desenvolvimento do jogo da forca, foram aplicados diversos conceitos fundamentais de programação orientada a objetos e design de software:

1. Orientação a Objetos (POO)

A implementação do jogo da forca segue os princípios da POO, que permite a organização do código em estruturas modulares e reutilizáveis chamadas classes. Isso facilita a manutenção, extensão e compreensão do código.

2. Encapsulamento

Cada classe no projeto tem seus próprios atributos e métodos, encapsulando o comportamento e dados relacionados a uma funcionalidade específica. Por exemplo, a classe ***Jogo*** gerencia o estado e a lógica do jogo da forca, enquanto a classe ***Palavras*** é responsável pela seleção e manipulação das palavras usadas no jogo.

3. Herança e Polimorfismo

O uso de herança está presente na forma de classe que compartilha o uso comum de metodos por exemplo, a classe interface que estende de JFrame. O polimorfismo, por sua vez, permite que métodos sejam chamados de forma genérica, garantindo flexibilidade no comportamento da aplicação.

4. Modularização

O código foi dividido em várias classes que representam diferentes componentes do jogo, como ***App, Forca, Historico, InterfaceGrafica, Jogo,*** e ***Palavras.*** Cada classe tem uma responsabilidade específica, promovendo um design modular e bem estruturado. A classe ***App*** atua como o ponto de entrada da aplicação, enquanto ***InterfaceGrafica*** lida com a interação do jogador.

6. Persistência de Dados e Leitura de Dados Externos

A classe ***Historico*** armazena e recupera informações sobre partidas anteriores, proporcionando persistência de dados e permitindo ao jogador revisar seu desempenho em partidas anteriores. Também é possível carregar diferentes palavras salvas em um arquivo ***.txt.***

Conclusão

Durante o desenvolvimento do meu projeto de um jogo da forca em JAVA, criei uma aplicação interativa que facilita a experiência de jogar e gerenciar as palavras escolhidas para o jogo.

Utilizei princípios de orientação a objetos, como herança e polimorfismo, para garantir que o código fosse organizado e de fácil manutenção. As funcionalidades foram distribuídas em classes bem definidas, promovendo uma comunicação eficiente entre os diferentes elementos do jogo.

O sistema permite que os jogadores interajam de maneira dinâmica com o jogo, enquanto o administrador pode adicionar ou remover palavras de forma simples. Em resumo, esse projeto foi uma excelente oportunidade para aplicar conceitos de programação e design de software, resultando em uma solução funcional que atende aos requisitos estabelecidos.

Referências

Microsoft Documentation. (2024). *Visual Studio Code Documentation*.

- Disponível em: https://code.visualstudio.com/docs
- Documentação oficial da IDE utilizada para o desenvolvimento do sistema.

GNU Make Manual. (2024). *Make - A GNU Manual*.

- Disponível em: https://www.gnu.org/software/make/manual/make.html
- Guia sobre o uso do `make`, uma ferramenta essencial para a construção e compilação do projeto.

Astah UML. (2024). *Astah UML Documentation*.

- Disponível em: https://astah.net/astah-uml
- Documentação sobre o uso do Astah UML para modelagem e visualização das relações entre as classes.