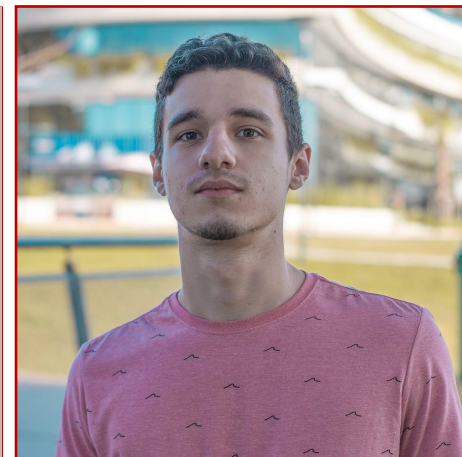
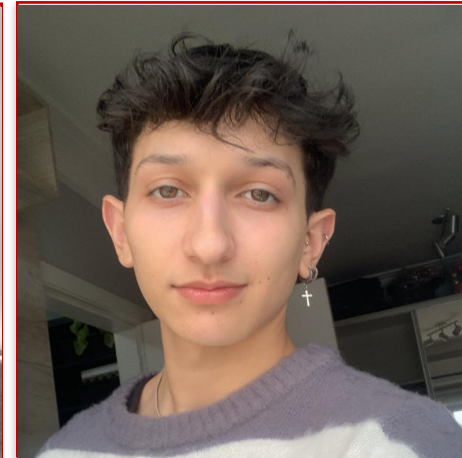
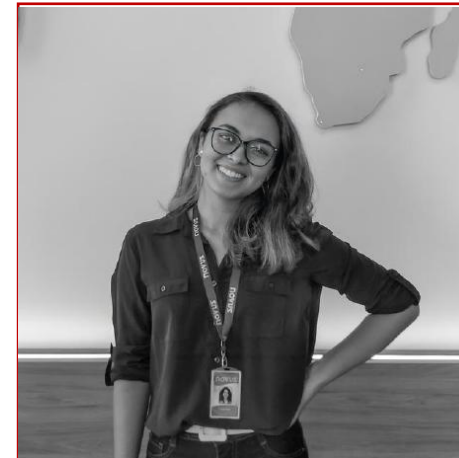


# JOGO DA FORÇA

Bruno Castanho, Leandra Machado, Lucas Gomes,  
Maria Eduarda Casali, Vitor Arguilar  
Facilitador: Lucas Gomes

Técnicas de Construção de Programas – Turma A  
Prof. Karina Kohl  
2025/1

GRUPO 07



# DESCRIÇÃO DO PROBLEMA

❑ **Contexto**

POO

❑ **Motivação**

Aprendizado Lúdico

Desafio

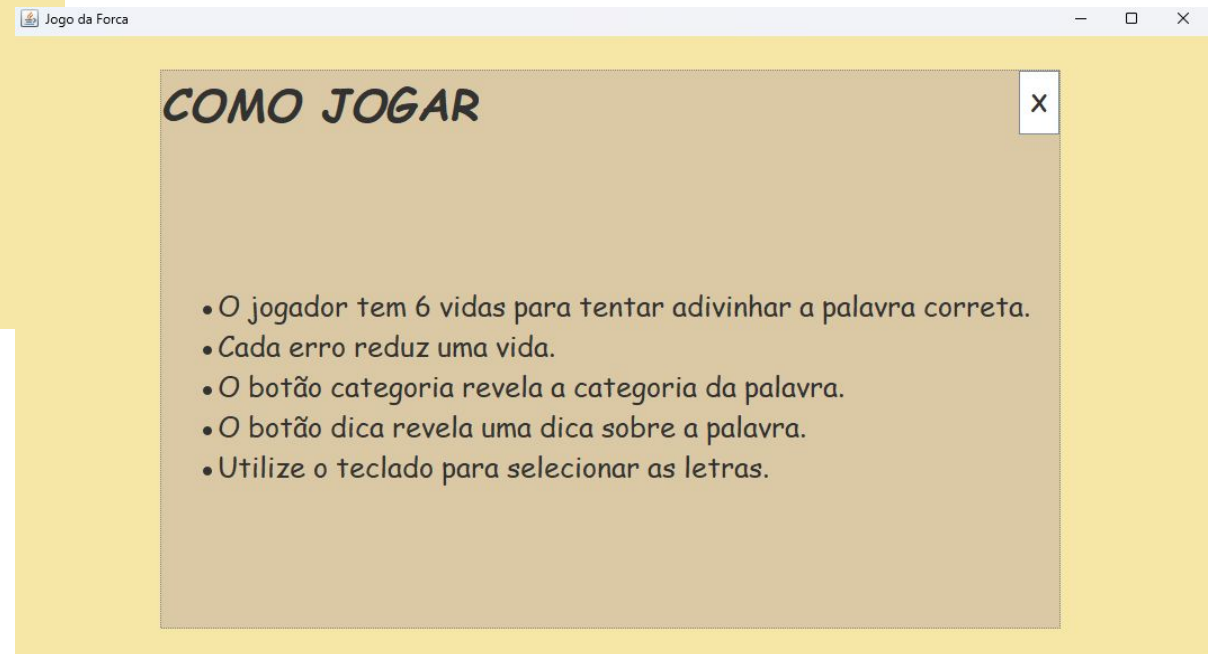
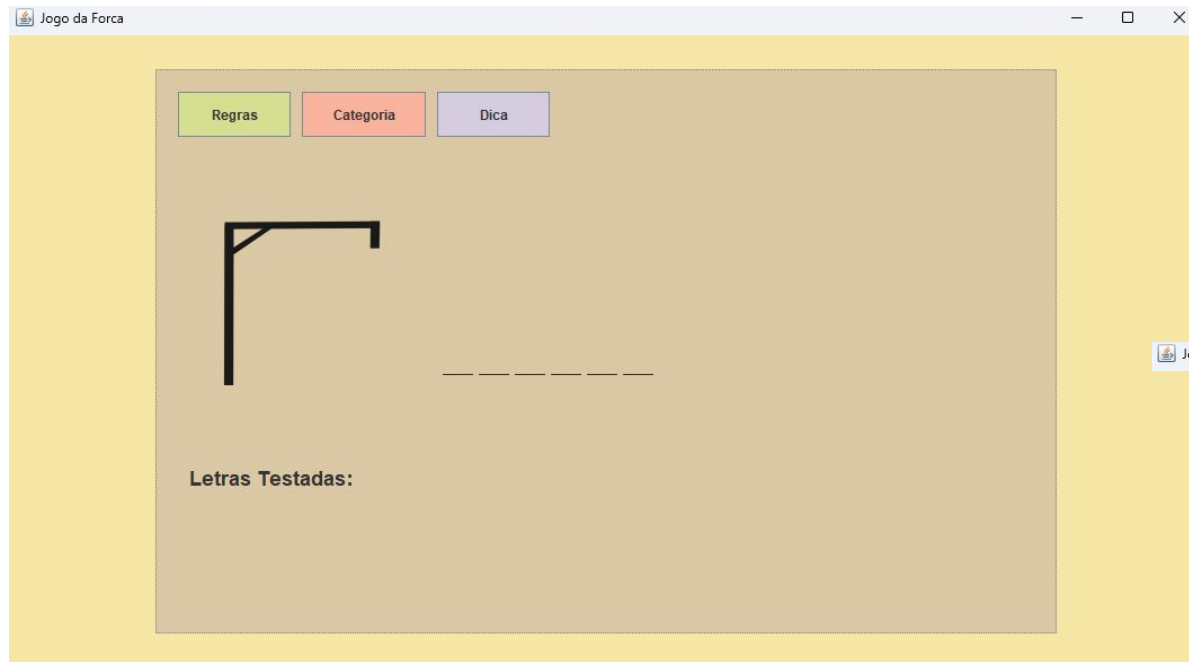
❑ **Objetivos**

Jogo

Explorar Conceitos

Praticidade

# DESCRIÇÃO DO PROBLEMA



# REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

## ❑ REQUISITOS FUNCIONAIS:

- ⇒ lógica de funcionamento
- ⇒ entendimento do usuário

## ❑ EXEMPLOS:

- ⇒ operação do botão regras
- ⇒ escolha da letra de cada rodada através do teclado
- ⇒ solicitação da categoria da palavra
- ⇒ compreensão visual das letras já selecionadas

# REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

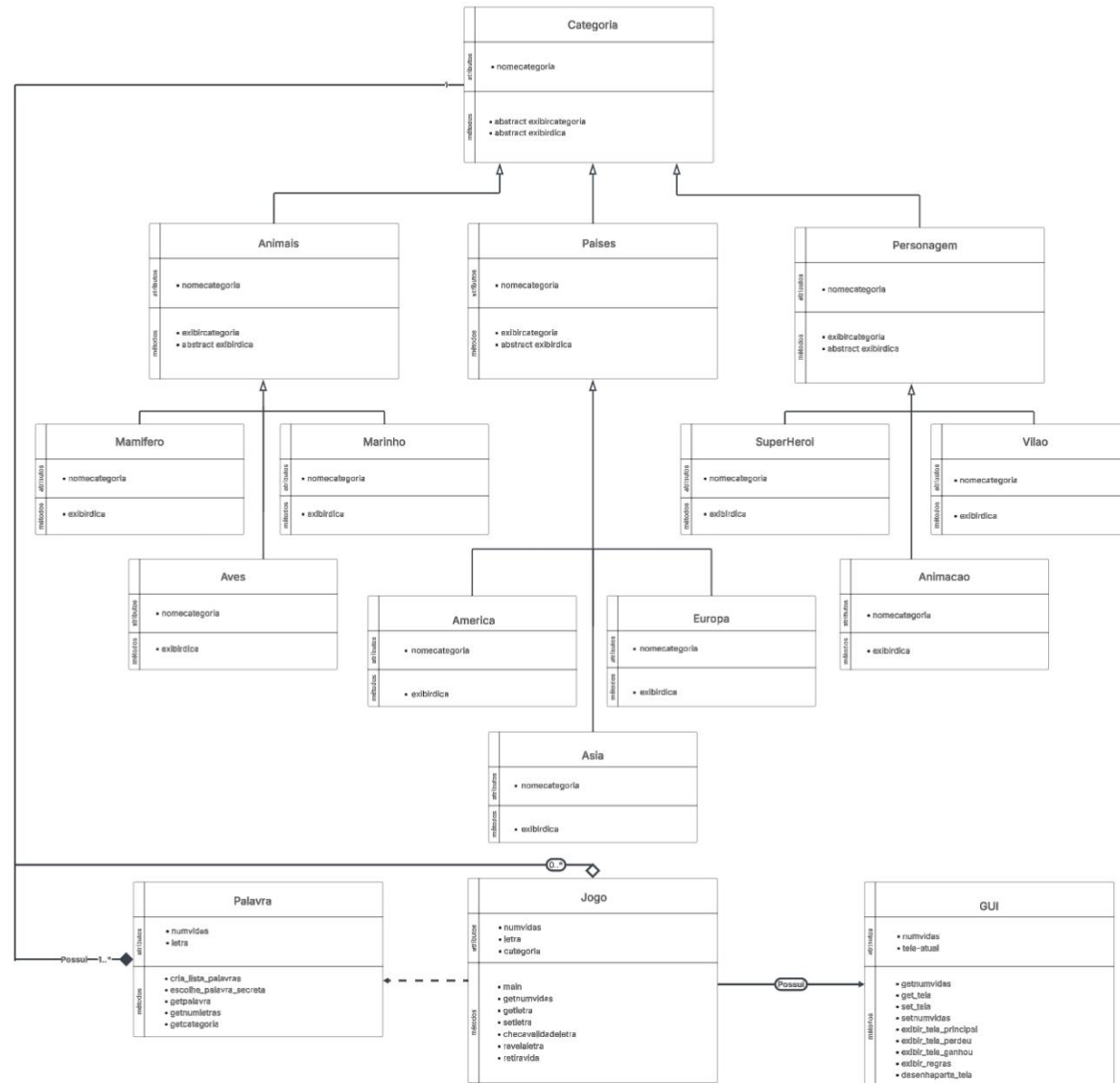
## ❑ REQUISITOS NÃO FUNCIONAIS:

- ⇒ experiência do usuário
- ⇒ segurança e estabilidade do programa

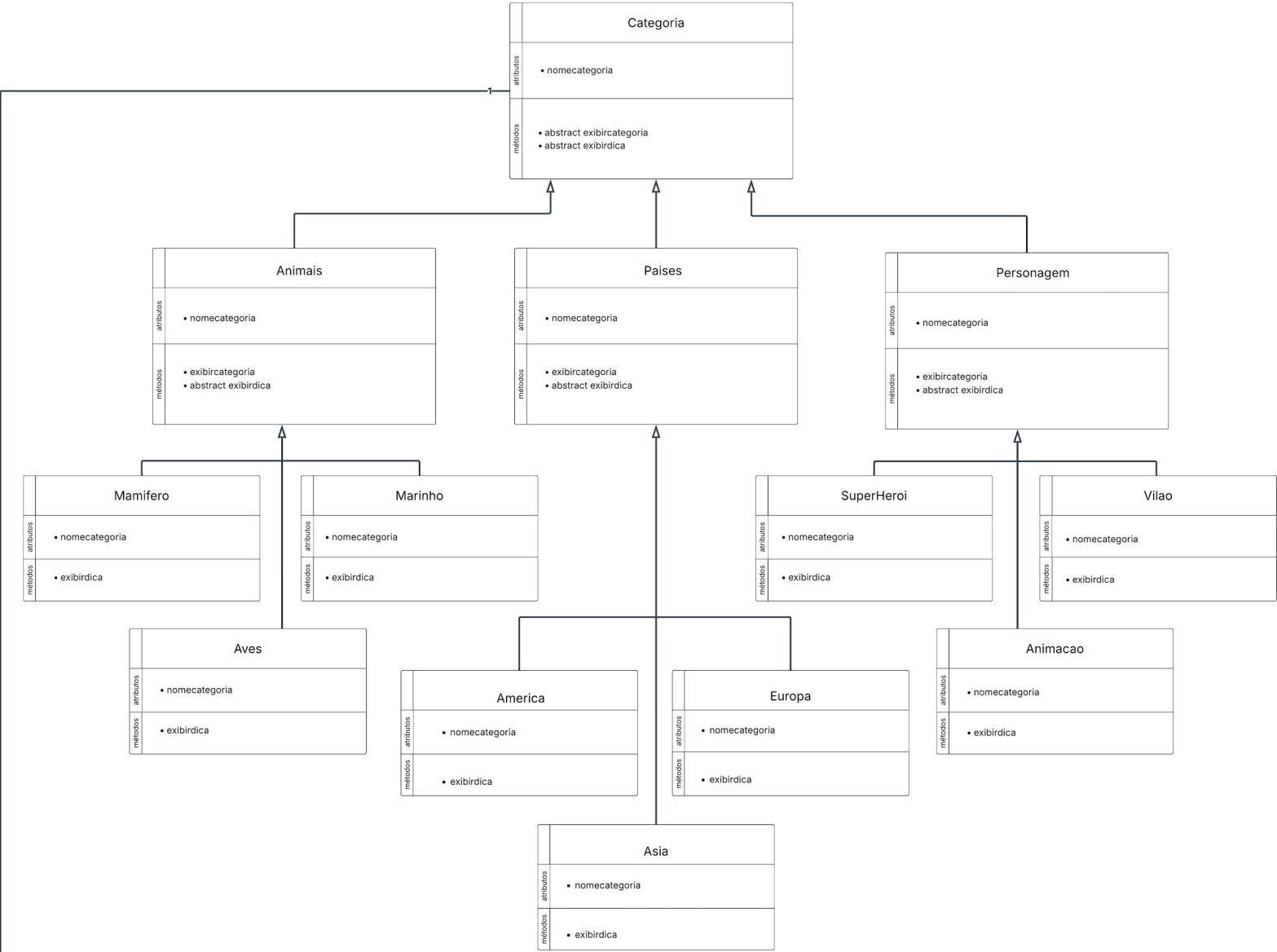
## ❑ EXEMPLOS:

- ⇒ carregamento da interface em no máximo 2 segundos
- ⇒ validação de novas funcionalidades por revisão de código
- ⇒ reinício da partida com 1 clique
- ⇒ contraste suficiente dos botões para boa visualização

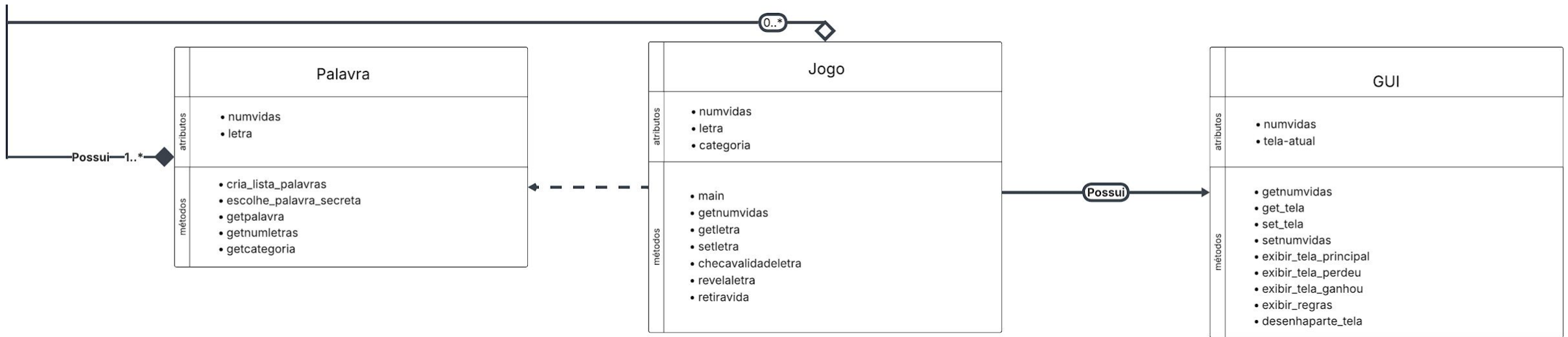
# DIAGRAMA DE CLASSES - ETAPA 1



# DIAGRAMA DE CLASSES - ETAPA 1



# DIAGRAMA DE CLASSES - ETAPA 1



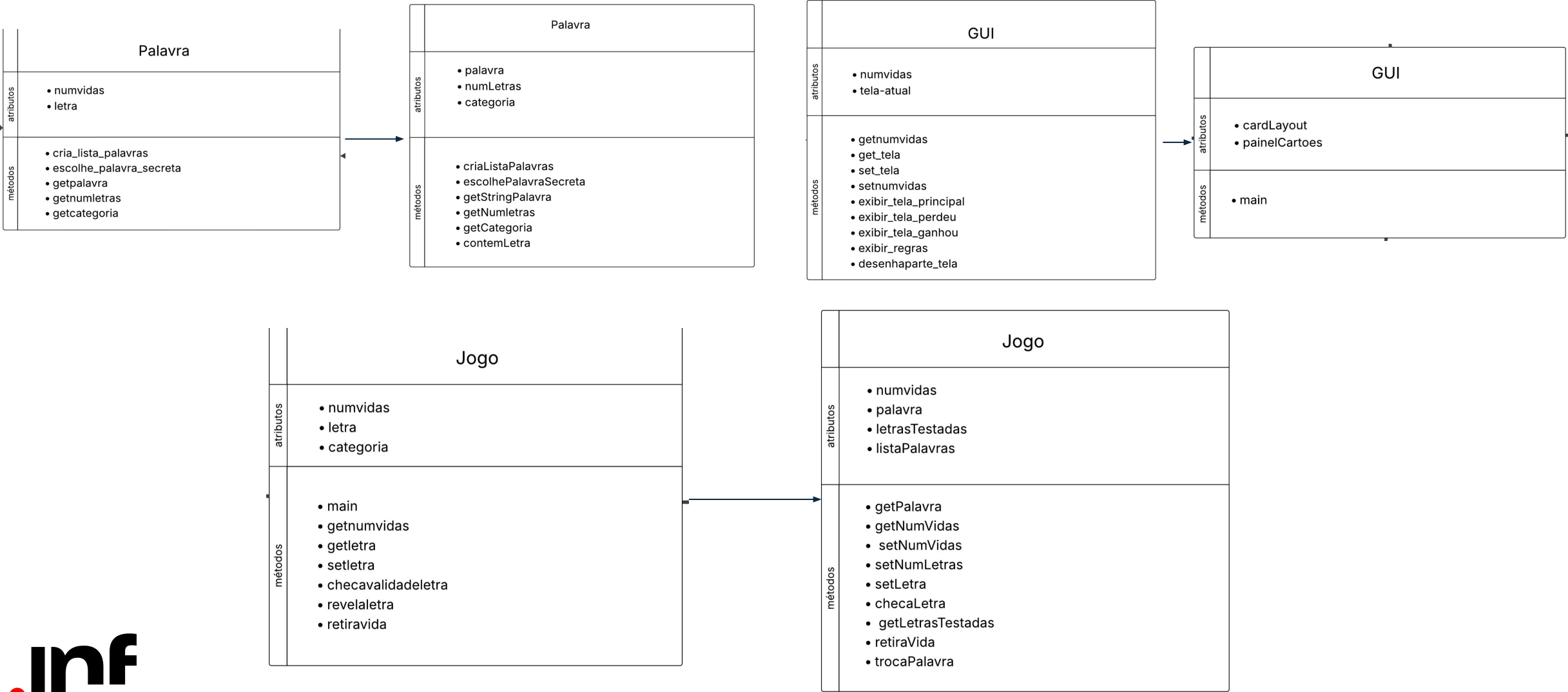


# DIAGRAMA DE CLASSES - ETAPA 1

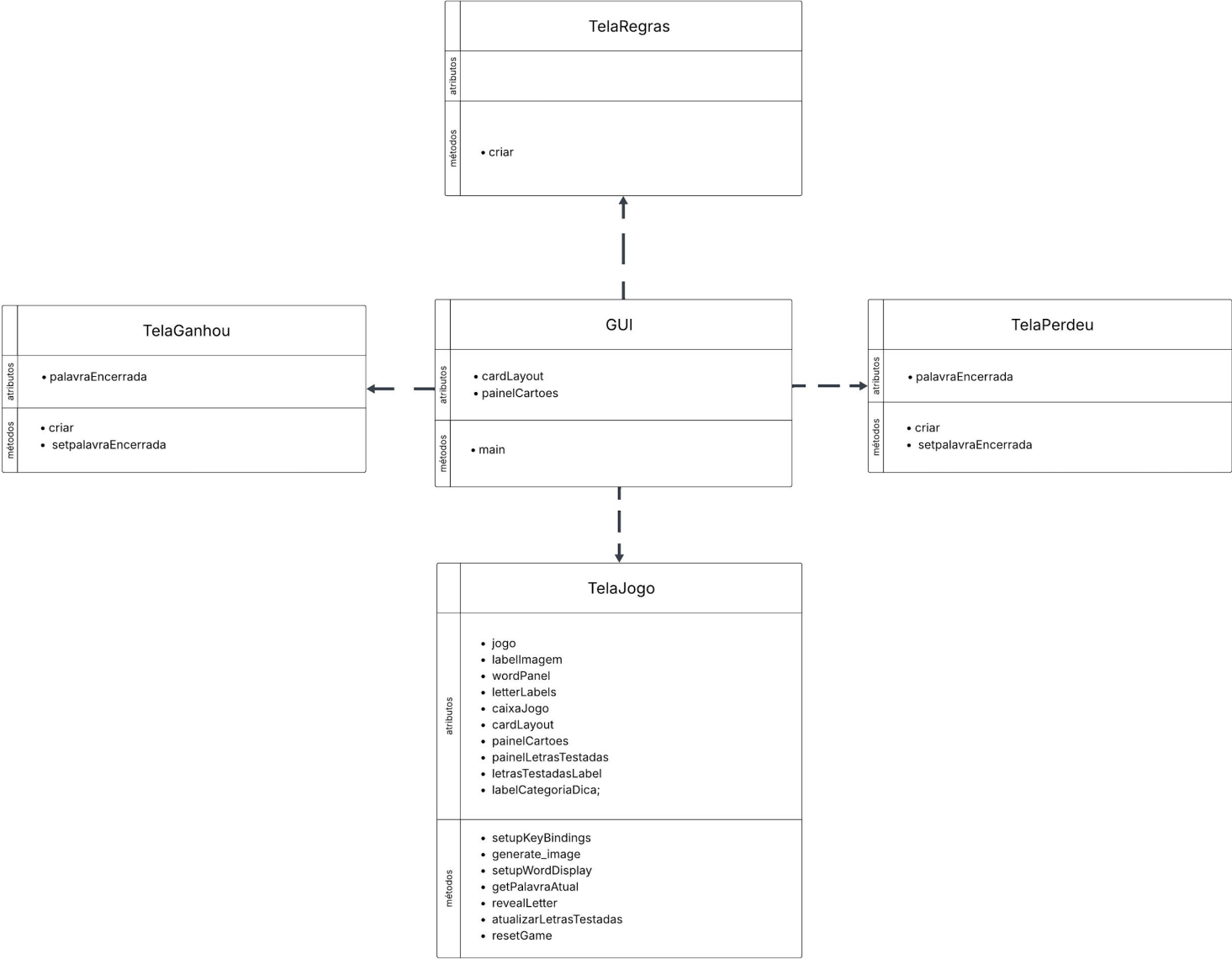
## Relacionamentos:

- ★ Dependência: funções da classe Jogo que utilizam como parâmetro a classe Palavra. Ex: checavalidadeletra.
- ★ Generalização/Especialização: subclasses de Categoria.
- ★ Associação/Agregação: Jogo possui uma Categoria, mas sem ela ainda existe.
- ★ Associação/Composição: Palavra possui uma Categoria, mas sem a existência de Palavra não faria sentido Categoria existir.


# DIAGRAMA DE CLASSES - ETAPA 2



# DIAGRAMA DE CLASSES - ETAPA 2



# VISÃO DO REPOSITÓRIO - README



The screenshot shows a GitHub repository README page. At the top, there's a 'README' tab and a dark header bar. Below the header, there are three colored boxes: 'JAVA' (grey), 'SWING' (orange), and 'JUNIT' (red). The main title is 'Trabalho Final TCP 2025/1 - Jogo da Forca' with a target icon. The text describes the project as part of the 'Técnicas de Construção de Programas (TCP) 2025/1' course, taught by Karina Kohl. It lists the project's objective: to apply knowledge in version control, data structures, and OOP. A button 'Review the assignment due date' is visible. The 'Integrantes' section lists five team members with their names as links.

README

JAVA SWING JUNIT

## 🎯 Trabalho Final TCP 2025/1 - Jogo da Forca

Este repositório contém o projeto elaborado pelo **Grupo 7** da disciplina de *Técnicas de Construção de Programas (TCP) 2025/1*, ministrada pela professora **Karina Kohl**.

O projeto tem como objetivo aplicar os conhecimentos adquiridos ao longo do semestre, como:

- Controle de versionamento (Git/GitHub)
- TAD (Tipos Abstratos de Dados)
- Programação Orientada a Objetos (POO)

📅 Review the assignment due date


### 👥 Integrantes

- [Bruno Castanho](#)
- [Leandra Machado](#)
- [Lucas Gomes](#)
- [Maria Eduarda Casali](#)
- [Vitor Arguilar](#)

# VISÃO DO REPOSITÓRIO - README

## Requisitos

- [Java JDK 24.0.1](#) ou superior instalado
- Ambiente compatível com execução de arquivos `.jar` (via terminal ou duplo clique)

 **Importante:** O projeto não funcionará corretamente sem a versão adequada da JRE ou JDK instalada. Você pode baixá-las aqui:  
[java.com](#)  
[oracle.com](#)

## Instalação

1. Clone este repositório ou faça o download manual:

```
git clone https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo07.git
```

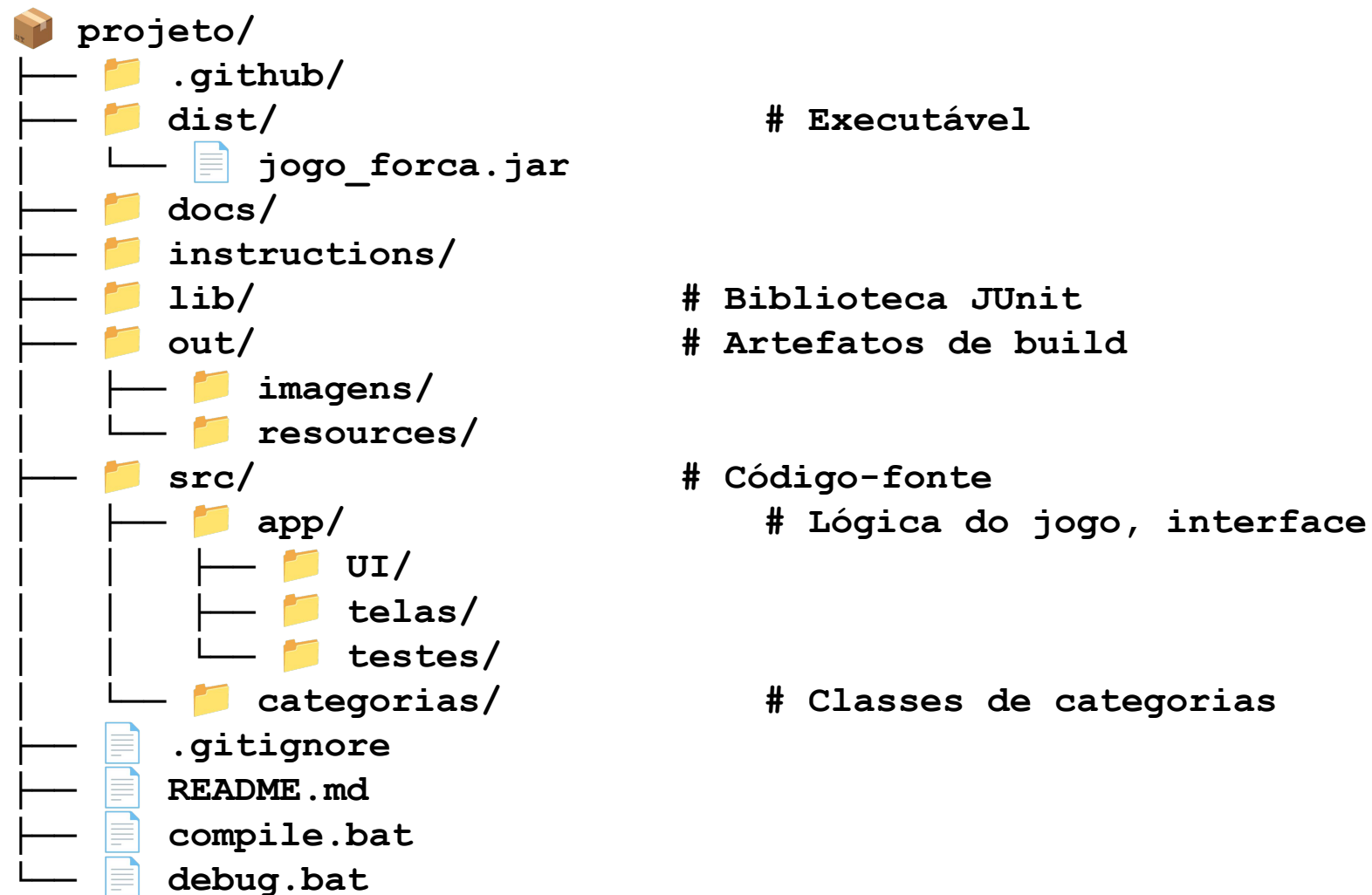
2. Execute o arquivo `compile.bat` presente na pasta raiz do projeto para compilar o código.
3. A compilação gerará um arquivo `.jar` na pasta `dist/`.

## Execução

Acesse a pasta `dist/` e execute o jogo via duplo clique (em sistemas que reconheçam `.jar` como executável) ou via terminal:

```
java -jar jogo_forca.jar
```

# VISÃO DO REPOSITÓRIO - DIRETÓRIOS



















# VISÃO DO REPOSITÓRIO - COMMITS

|  |         |          |    |
|--|---------|----------|----|
| <b>remove botoes de teste</b><br>lemachadoh committed 2 weeks ago  | a1a7f6e |          | <> |
| <b>Merge branch 'main' of <a href="https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo07">https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo07</a> into interface</b><br>lemachadoh committed 2 weeks ago | f058144 |          | <> |
| <b>Merge pull request #4 from SW-Engineering-Courses-Karina-Kohl/jogo</b><br>lemachadoh authored 2 weeks ago   | a914b5f | Verified | <> |
| <b>Merge branch 'jogo' of <a href="https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo07">https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo07</a> into jogo</b><br>VitorArguilar committed 2 weeks ago   |         |          |    |
| <b>logica jogo</b><br>VitorArguilar committed 2 weeks ago  |         |          |    |

|  |         |          |    |
|--|---------|----------|----|
| <b>Adc batch files para debug e geração do jar.</b><br>trshpnd committed 2 days ago      | 4a9c005 |          | <> |
| <b>Correção de bug de palavras com hífens e acentos.</b><br>trshpnd committed 2 days ago | 2404977 |          | <> |
| <b>Add files via upload</b><br>dudacasali authored 3 days ago                            | a8a36c3 | Verified | <> |
| <b>Add files via upload</b><br>dudacasali authored 3 days ago                            | dd4e537 | Verified | <> |
| <b>Add files via upload</b><br>dudacasali authored 3 days ago                            | f054ee1 | Verified | <> |

# VISÃO DO REPOSITÓRIO - BRANCHES

| Default   |   |              |        |       |  |   |
|---|---|--------------|--------|-------|--|---|
| Branch  | Updated   | Check status | Behind | Ahead | Pull request   |   |
| <code>main</code>        |  4 hours ago   |              |        |       |  #1   |  ...   |
| Default   |   |              |        |       |  |   |
| Active branches   |   |              |        |       |  |   |
| Branch  | Updated   | Check status | Behind | Ahead | Pull request   |   |
| <code>feedback</code>    |  4 days ago    |              | 58     | 1     |  #5   |  ...   |
| <code>interface</code>  |  2 weeks ago  |              | 21     | 0     |  #2  |  ...  |
| <code>jogo</code>      |  2 weeks ago |              | 29     | 0     |  #4 |  ... |



# VISÃO DO REPOSITÓRIO - LINK

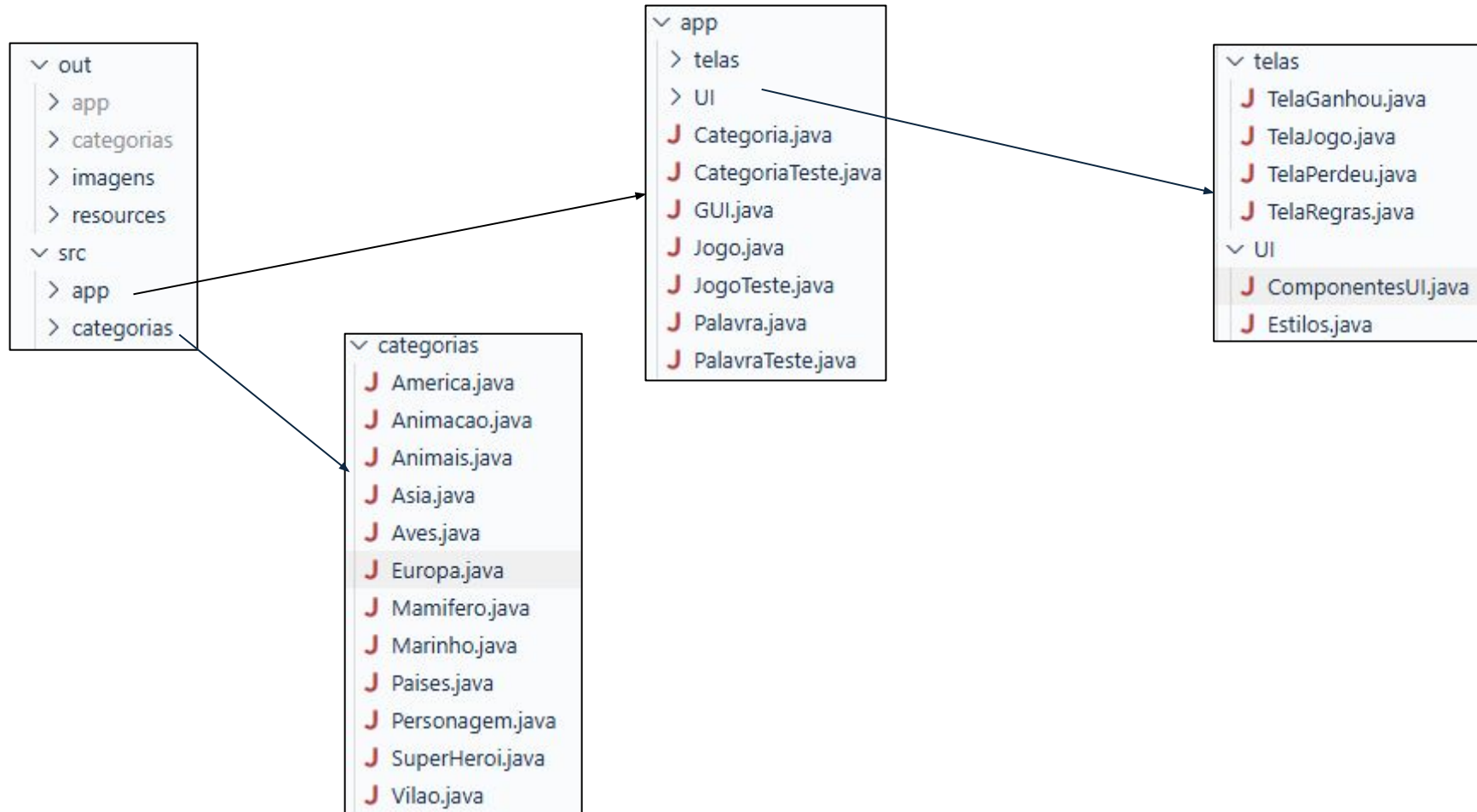
[https://github.com/SW-Engineering-Courses-Karina-Kohl/  
tcp-final-20251-grupo07](https://github.com/SW-Engineering-Courses-Karina-Kohl/tcp-final-20251-grupo07)

# VISÃO GERAL DO CÓDIGO

❖ **Sumário das Pastas**

❖ **Principais classes e funções**

# VISÃO GERAL DO CÓDIGO - Pastas



# VISÃO GERAL DO CÓDIGO - Jogo

```
public class Jogo {  
  
    private int numvidas;  
    private Palavra palavra;  
    private Set<Character> letrasTestadas;  
    private List<Palavra> listaPalavras;  
  
    public Jogo() {  
        //print test message  
        System.out.println(x:"Iniciando um novo jogo...");  
        this.numvidas = 6;  
        this.letrasTestadas = new HashSet<>();  
        this.listaPalavras = Palavra.criaListaPalavras();  
        this.trocaPalavra();  
        System.out.println("Palavra atual: " + palavra.getStringPalavra());  
    }  
}
```

```
public int checaLetra(char letra) {  
    // Verifica se a letra digitada pelo usuário está na palavra e ainda não foi testada  
    if (letrasTestadas.contains(letra)) {  
        return 0; // Letra já testada  
    }  
    if (this.palavra.contemLetra(letra)){  
        letrasTestadas.add(letra);  
        return 1; // Letra correta  
    }  
    else{  
        letrasTestadas.add(letra);  
        retiraVida(); // Retira uma vida se a letra não estiver na palavra  
        return -1; // Letra incorreta  
    }  
}
```

```
public void reset() {  
    // Reseta o jogo para um novo início  
    this.numvidas = 6;  
    this.letrasTestadas.clear();  
    this.trocaPalavra();  
    System.out.println("Palavra atual: " + palavra.getStringPalavra());  
}
```

# VISÃO GERAL DO CÓDIGO - GUI

```
public class GUI extends JFrame {  
    private CardLayout cardLayout;  
    private JPanel painelCartoes;
```

```
public static void main(String[] args) {  
    SwingUtilities.invokeLater(GUI::new);  
}
```

```
// Painel amarelo (fundo)  
JPanel painelFundo = new JPanel();  
painelFundo.setBackground(Estilos.AMARELO);  
painelFundo.setLayout(new GridBagLayout()); // centraliza o painel de conteúdo  
add(painelFundo);
```

```
// Painel cinza menor com CardLayout para alternar telas  
cardLayout = new CardLayout();  
painelCartoes = new JPanel(cardLayout);  
painelCartoes.setPreferredSize(Estilos.TAMANHO_TELA_JOGO);  
painelCartoes.setBackground(Estilos.CINZA);
```

```
painelCartoes.add(telaJogo, constraints:"JOGO");  
painelCartoes.add(telaRegras, constraints:"REGRAS");  
painelCartoes.add(telaPerdeu, constraints:"PERDEU");  
painelCartoes.add(telaGanhou, constraints:"GANHOU");  
  
painelFundo.add(painelCartoes);
```

# VISÃO GERAL DO CÓDIGO - Palavra

```
public class Palavra {  
    private String palavra;  
    private int numLetras;  
    private Categoria nomeCategoria;  
  
    public Palavra(String palavra, int numLetras, Categoria nomeCategoria) {  
        this.palavra = palavra;  
        this.numLetras = numLetras;  
        this.nomeCategoria = nomeCategoria;  
    }  
}
```

```
public static Palavra escolhePalavraSecreta(List<Palavra> lista) {  
    if (lista == null || lista.isEmpty()) {  
        return null; // trocar para lançar exceção?  
    }  
  
    int randIndex = ThreadLocalRandom.current().nextInt(lista.size());  
    return lista.get(randIndex);  
}
```

```
public static List<Palavra> criaListaPalavras() {  
    List<Palavra> listaPalavras = new ArrayList<>();  
  
    String[] arquivos = { "america.txt", "asia.txt", "europa.txt", "aves.txt", "mamiferos.txt", "peixes.txt",  
        "herois.txt", "viloes.txt", "animacao.txt" };  
  
    // Loop que itera sobre o array de diretórios.  
    // cada arquivo define a categoria a ser adicionada à palavra.  
  
    // exemplo para "viloes.txt":  
    // Categoria ctg = new Vilao("Personagem");  
    // Palavra p = new Palavra(linha, linha.length(), ctg);  
    // listaPalavras.add(p);  
}
```

# VISÃO GERAL DO CÓDIGO - Categorias

```
public abstract class Categoria {  
    public String nomecategoria;  
  
    public Categoria(String var1) {  
        this.nomecategoria = var1;  
    }  
  
    public abstract String exibirCategoria();  
  
    public abstract String exibirDica();  
}
```

```
public abstract class Animais extends Categoria {  
    public Animais (String nomecategoria) {  
        super(nomecategoria);  
    }  
  
    public abstract String exibirDica();  
  
    @Override  
    // Retorna o nome da categoria, impressão na tela é feita por outro método.  
    public String exibirCategoria () {  
        return "Categoria: " + nomecategoria;  
    }  
}
```

```
public class Aves extends Animais {  
    public Aves (String nomecategoria) {  
        super(nomecategoria);  
    }  
  
    @Override  
    // Retorna o nome da categoria, impressão na tela é feita por outro método.  
    public String exibirDica () {  
        return "Este animal é uma ave.";  
    }  
}
```



# VISÃO GERAL DO CÓDIGO - TelaJogo

```
public class TelaJogo extends JPanel {  
    private Jogo jogo;  
  
    private JLabel labelImagem;  
    private JPanel wordPanel;  
    private JLabel[] letterLabels;  
    private JPanel caixaJogo;  
    private CardLayout cardLayout;  
    private JPanel painelCartoes;  
    private JPanel painelLetrasTestadas;  
    private JLabel letrasTestadasLabel;  
    private JLabel labelCategoriaDica;
```

```
    private void resetGame() {  
        // remover o painel de Letras atual  
        caixaJogo.remove(wordPanel);  
        // reseta o jogo  
        jogo.reset();  
        // limpa letras testadas  
        letrasTestadasLabel.setText(text:"");  
        labelCategoriaDica.setText(text:"");  
        // reseta imagem  
        labelImagem.setIcon(new ImageIcon(generate_image()));  
        // reseta a exibição das letras  
        setupWordDisplay(jogo.getStringPalavra());  
        // recarrega layout  
        caixaJogo.revalidate();  
        caixaJogo.repaint();  
    }
```

```
public void actionPerformed(ActionEvent e) {  
    int letter_in = jogo.checaLetra(key.charAt(index:0));  
    Palavra palavraEncerrada = jogo.getPalavra(); // palavra antes do reset  
    switch (letter_in) {  
        case 1:  
            revealLetter(key.charAt(index:0), jogo.getStringPalavra());  
            atualizarLetrasTestadas();  
            if (todasLetrasAcertadas(jogo.getStringPalavra())) {  
                TelaGanhou.setPalavraEncerrada(palavraEncerrada);  
                cardLayout.show(painelCartoes, name:"GANHOU");  
                resetGame();  
            }  
            break;  
        case -1:  
            labelImagem.setIcon(new ImageIcon(generate_image()));  
            if (jogo.getNumvidas() <= 0) {  
                TelaPerdeu.setPalavraEncerrada(palavraEncerrada);  
                cardLayout.show(painelCartoes, name:"PERDEU");  
                resetGame();  
            }  
            atualizarLetrasTestadas();  
            break;  
    }  
}
```



- ❖ **Ferramenta utilizada: Junit**
  - **Mesma utilizada no laboratório.**
- ❖ **Aplicativo/Jogo com entradas “controladas”**
  - **Casos testados para validação dos métodos**
  - **Casos testados para validação dos atributos**

# TESTES

## → CategoriaTeste

```
@Test
public void testConstrutorECampos() {
    Categoria categoria = new Aves(nomecategoria:"Animais");

    assertEquals("Animais", categoria.nomecategoria);
    assertEquals("Categoria: Animais", categoria.exibirCategoria());
    assertEquals("Este animal é uma ave.", categoria.exibirDica());
}
```

```
@Test
public void testConstrutorECampos() {
    Categoria categoria = new Aves(nomecategoria:"Animais");

    assertEquals("Países", categoria.nomecategoria);
    assertEquals("Categoria: Animais", categoria.exibirCategoria());
    assertEquals("Este animal é uma ave.", categoria.exibirDica());
}
```

# TESTES

## → PalavraTeste

```
@Test
public void testCriacaoPalavra() {
    Categoria categoria = new America(nomecategoria:"Países");
    Palavra palavra = new Palavra(palavra:"Brasil", numLetras:6, categoria);

    assertEquals("Brasil", palavra.getStringPalavra());
    assertEquals(2, palavra.getNumLetras()); ❌ 6 != 2
}
```

❌ testCriacaoPalavra() \$(symbol-class) PalavraTeste < \$(symbol-namespace) app < \$(project) tcp-final-  
Expected [2] but was [6]

```
@Test
public void testEscolhePalavraSecretaListaVazia() {
    List<Palavra> lista = new ArrayList<>();
    Palavra secreta = Palavra.escolhePalavraSecreta(lista);
    assertNull(secreta);
}
```

```
@Test
public void testContemLetraPresente() {
    Categoria categoria = new America(nomecategoria:"Países");
    Palavra palavra = new Palavra(palavra:"Argentina", numLetras:9, categoria);

    assertTrue(palavra.contemLetra(letra:'A') || palavra.contemLetra(letra:'a'));
}
```

```
@Test
public void testEscolhePalavraSecreta() {
    Categoria categoria = new America(nomecategoria:"Países");
    List<Palavra> lista = new ArrayList<>();
    lista.add(new Palavra(palavra:"Chile", numLetras:5, categoria));
    lista.add(new Palavra(palavra:"Peru", numLetras:4, categoria));

    Palavra secreta = Palavra.escolhePalavraSecreta(lista);

    assertNotNull(secreta);
    assertTrue(lista.contains(secreta));
}
```

# TESTES

## → JogoTeste

```
@Test
public void testInicializacaoJogo() {
    assertEquals(6, jogo.getNumvidas(), "Jogo deve começar com 6 vidas");
    assertNotNull(jogo.getPalavra(), "Palavra não deve ser nula");
    assertTrue(jogo.getLetrasTestadas().isEmpty(), "Nenhuma letra deve ter sido testada");
}
```

```
@Test
public void testAcertarLetraCorreta() {
    String palavra = jogo.getStringPalavra();
    char letraCorreta = palavra.charAt(index:0); // pega a primeira letra da palavra secreta

    int resultado = jogo.checaLetra(letraCorreta);
    assertEquals(1, resultado, "Deveria retornar 1 para letra correta");
    assertTrue(jogo.getLetrasTestadas().contains(letraCorreta), "Letra correta deveria estar nas testadas");
    assertEquals(6, jogo.getNumvidas(), "Número de vidas não deve mudar em acerto");
}
```

✘ testAcertarLetraCorreta() \$(symbol-class) JogoTeste < \$(symbol-namespace) app < \$(project) tcp-final-20251-grupo07-main  
Expected [-1] but was [1]  
org.opentest4j.AssertionFailedError: Deveria retornar 1 para letra correta ==> expected: [-1] but was: [1] at app.JogoTeste.

```
@Test
public void testRetiraVida() {
    jogo.retiraVida();
    assertEquals(5, jogo.getNumvidas(), "Após retirar vida, deve ter 5 vidas restantes");
}
```

```
@Test
public void testResetJogo() {
    jogo.checaLetra(letra:'Z'); // erra de propósito
    jogo.checaLetra(letra:'X');
    assertEquals(4, jogo.getNumvidas(), "Após errar duas letras, vidas devem ser 4");

    jogo.reset();
    assertEquals(6, jogo.getNumvidas(), "Após reset, deve voltar a 6 vidas");
    assertTrue(jogo.getLetrasTestadas().isEmpty(), "Após reset, letras testadas devem estar vazias");
    assertNotNull(jogo.getPalavra(), "Após reset, palavra não deve ser nula");
}
```



## ★ Logs detalhados para debug durante o jogo

### Carregamento das listas de palavras

```
Iniciando um novo jogo...
Lendo: resources/america.txt
Arquivo carregado com sucesso!
Lendo: resources/asia.txt
Arquivo carregado com sucesso!
Lendo: resources/europa.txt
Arquivo carregado com sucesso!
Lendo: resources/aves.txt
Arquivo carregado com sucesso!
Lendo: resources/mamiferos.txt
Arquivo carregado com sucesso!
Lendo: resources/peixes.txt
Arquivo carregado com sucesso!
Lendo: resources/herois.txt
Arquivo carregado com sucesso!
Lendo: resources/viloes.txt
Arquivo carregado com sucesso!
Lendo: resources/animacao.txt
Arquivo carregado com sucesso!
```

### Funcionamento do jogo

```
Palavra atual: República Tcheca
I Letter pressed: 1
Numero de vidas: 6
Letra esperada: E, letra exibida:
R Letter pressed: 1
Numero de vidas: 6
Letra esperada: P, letra exibida:
E Letter pressed: 1
Numero de vidas: 6
Letra esperada: Ú, letra exibida:
P Letter pressed: 1
Numero de vidas: 6
Letra esperada: Ú, letra exibida:
L Letter pressed: 1
Numero de vidas: 6
Letra esperada: Ú, letra exibida:
C Letter pressed: 1
Numero de vidas: 6
Letra esperada: Ú, letra exibida:
A Letter pressed: 1
Numero de vidas: 6
```

# DEMONSTRAÇÃO

- ❑ **A execução do jogo pode ser feita:**
  - ★ **pelo terminal e IDE's**
  - ★ **pelo arquivo compile.bat, que gera o arquivo .jar do programa**

# CONCLUSÃO

## ❖ Dificuldades:

- 😞 escolha do jogo
- 😞 grafia de algumas palavras
- 😞 preparação da apresentação

## ❖ Aprendizados:

- 😄 manipulação dos pilares de POO
- 😄 noção de planejamento de software
- 😄 implementação em time

# CONCLUSÃO

## ★ Melhorias:



**implementação de dificuldade de jogo**



**parâmetros de qualidade de software**



**E... MUITO OBRIGADO!**



# DÚVIDAS

## ★ PERGUNTAS??

