FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA - UNIFOR
CENTRO DE TECNOLOGIA
CURSO ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

T164-87 PROGRAMAÇÃO ORIENTADA A OBJETOS

Relatório de Desenvolvimento de um jogo Quiz

Equipe:
Felipe Martins
Jardiana Galvão
Samuel Chagas
Saul Santos

Fortaleza-CE
2023

PROGRAMAÇÃO ORIENTADA A OBJETOS - POO

PROF. SAMUEL BARROCAS

TRABALHO FINAL AV3

EQUIPE:

NOME	FUNÇÃO
Felipe Martins	Responsável pelos Testes
Jardiana Galvão	Responsável pelos diagramas e relatório
Samuel Chagas	Responsável pela interface e codificação
Saul Santos	Responsável pela interface e codificação

DETALHES:

Projeto: Quiz Game em Java

Tema: O Quiz “CodeJava TheQuiz” abordará questões sobre conceitos relativos à Programação Orientada a Objetos abordados durante o semestre 2 do Curso Análise e Desenvolvimento de Sistemas, Unifor, ministrada pelo Prof. Samuel Barrocas.

APRESENTAÇÃO DO JOGO E REGRAS:

O jogo consiste em uma lista de 10 perguntas sobre linguagem de programação de programação com itens de 'a' até 'd' para o jogador escolher o item correto. À medida que o jogador vai selecionando as respostas o jogo verifica se o item escolhido condiz com a resposta correta e salva a pontuação caso seja a resposta certa. No final das 10 perguntas o jogo apresenta o total da pontuação que o jogador atingiu.

ASPECTOS SOBRE O DESENVOLVIMENTO DO PROJETO

Introdução

Este relatório documenta o desenvolvimento bem-sucedido de um jogo do tipo Quiz em Java, denominado "JavaQuiz". O projeto envolveu diversas etapas, desde o planejamento inicial até a implementação e testes finais. O JavaQuiz visa proporcionar uma experiência interativa e educativa, desafiando os jogadores com perguntas de conhecimento sobre programação Orientada a Objetos.

1. Objetivo do Projeto

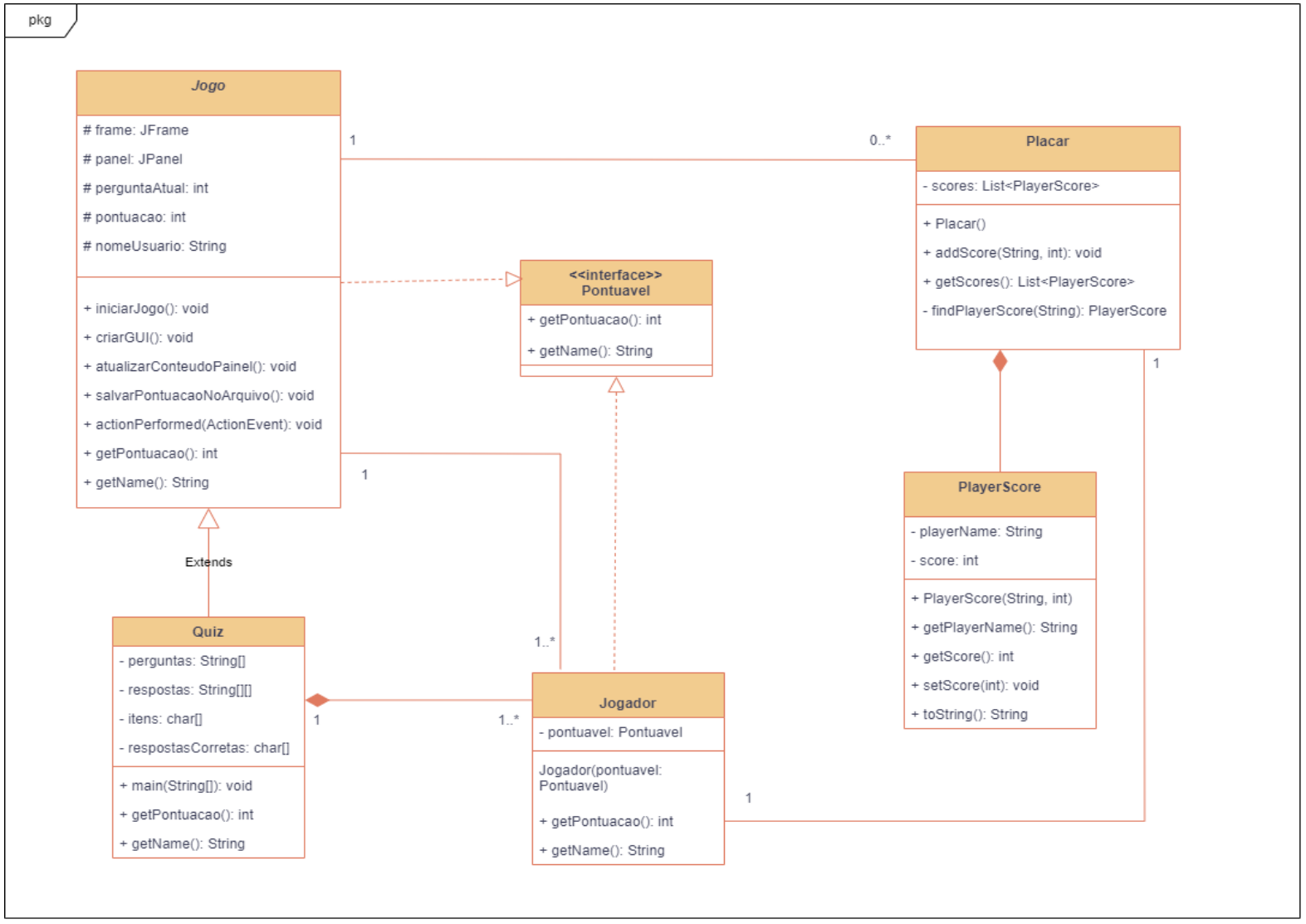
O objetivo principal do projeto foi criar um jogo de Quiz interativo, utilizando a linguagem de programação Java. O JavaQuiz foi concebido para entreter e educar de forma lúdica, abordando perguntas relacionadas à Programação Orientada a Objetos.

2. Metodologia de Desenvolvimento

As etapas de desenvolvimento foram realizadas por meio de sprints semanais com o acompanhamento do professor, e de forma resumida se dividiram em:

- **Planejamento:** Definição das funcionalidades, elaboração das categorias de perguntas e criação de um esboço inicial do design e diagrama de classes.
- **Codificação:** Desenvolvimento do código-fonte em Java, utilizando boas práticas de programação e orientação a objetos.
- **Testes**: Realização de testes unitários e integração para garantir a estabilidade e correção de bugs.
- **Aprimoramento**: Incorporação de feedbacks e otimizações para melhorar a experiência do usuário, bem como os requisitos solicitados para o projeto e sugestões obtidas durante as sprints.

3. Diagrama de Classes



4. Requisitos do Projeto

A construção do jogo incorporou os conceitos estudados na disciplina de programação Orientada a Objetos, conforme detalhado a seguir:

4.1. Classe Abstrata e Herança:

- Classe Abstrata: A classe abstrata é representada por *abstract class Jogo*. Ela é abstrata e contém métodos abstratos.

```
31 abstract class Jogo implements ActionListener,Pontuavel  {
32     protected JFrame frame;
33     protected JPanel panel;
34     protected int perguntaAtual = 0;
35     protected int pontuacao = 0;
36     protected String nomeUsuario;
37 
38     public void iniciarJogo() {
39         SwingUtilities.invokeLater(() -> {
40             criarGUI();
41         });
42     }
43 
44     protected abstract void criarGUI();
45     protected abstract void atualizarConteudoPainel();
46     protected abstract void exibirPlacarLideres();
47     protected abstract void salvarPontuacaoNoArquivo();
48 
49     @Override
50     public abstract void actionPerformed(ActionEvent e);
51 }
52 
```

- Herança: A classe *Quiz* herda da classe abstrata *Jogo* com *public class Quiz extends Jogo.*

```
57 public class Quiz extends Jogo {
58
59     private JFrame frame;
60     private JPanel panel;
61     private int perguntaAtual = 0;
62     private int pontuacao = 0;
63     private String nomeUsuario;
64
65     String[] perguntas = {
77
78     String[][] respostas = {
90
91     char[] itens = {'A', 'B', 'C', 'D'};
92
93     char[] respostasCorretas = {'D', 'C', 'B', 'A', 'D', 'B', 'A', 'C', 'B', 'B'};
94
95     public static void main(String[] args) {
96         Quiz quiz = new Quiz();
97         quiz.iniciarJogo();
98         Jogador jogador = new Jogador(quiz);
99         System.out.println(jogador.getName());
100        System.out.println("O quiz vale de 0 a " + jogador.getPontuacao());
101    }
102
103    @Override
104    public int getPontuacao() {
105        return pontuacao;
106    }
107
108    @Override
109    public String getName() {
110        return nomeUsuario;
111    }
```

4.2. Interface e Implementação:

- Interface: A interface é representada por interface *Pontuave*l. Ela define métodos *getPontuacao* e *getName*.

```
9  interface Pontuavel {
10     int getPontuacao();
11     String getName();
12 }
13
14 class Jogador implements Pontuavel {
15     public Jogador(Pontuavel pontuavel) {
16     }
17
18     @Override
19     public int getPontuacao() {
20         return 10;
21     }
22
23     @Override
24     public String getName() {
25         return "Bem vindo Aluno de POO";
26     }
27 }
28
```

- Implementação: A classe *Jogador* implementa a interface *Pontuavel* com *class Jogador implements Pontuavel.*

```
14 class Jogador implements Pontuavel {
15     public Jogador(Pontuavel pontuavel) {
16     }
17
18     @Override
19     public int getPontuacao() {
20         return 10;
21     }
22
23     @Override
24     public String getName() {
25         return "Bem vindo Aluno de POO";
26     }
27 }
28
29 abstract class Jogo implements ActionListener,Pontuavel  {
30     protected JFrame frame;
31     protected JPanel panel;
32     protected int perguntaAtual = 0;
33     protected int pontuacao = 0;
34     protected String nomeUsuario;
35
36     public void iniciarJogo() {
37         SwingUtilities.invokeLater(() -> {
38             criarGUI();
39         });
40     }
```

4.3. Sobrescrita de Método:

- Há sobrescrita de métodos nas classes *Jogador* (métodos *getPontuacao* e *getName*) e na classe Quiz (métodos *getPontuacao*, *getName, criarGUI, atualizarConteudoPainel, exibirPlacarLideres, salvarPontuacaoNoArquivo* e *actionPerformed*).

```
14 class Jogador implements Pontuavel {
15     public Jogador(Pontuavel pontuavel) {
16     }
17
18     @Override
19     public int getPontuacao() {
20         return 10;
21     }
22
23     @Override
24     public String getName() {
25         return "Bem vindo Aluno de POO";
26     }
27 }
```

```
55 public class Quiz extends Jogo {
56
```

```
.00
.01     @Override
.02     public int getPontuacao() {
.03         return pontuacao;
.04     }
.05
.06     @Override
.07     public String getName() {
.08         return nomeUsuario;
.09     }
.10
.11     @Override
.12     protected void criarGUI() {
.13         frame = new JFrame("");
```

4.4. Atributos Privados com Métodos Get e Set:

- Todos os atributos nas classes *Jogo, Quiz, Jogador,* e *PlayerScore* são privados. Métodos *get* e *set* estão implementados para esses atributos.

```
41     public static class PlayerScore {
42         private String playerName;
43         private int score;
44
45         public PlayerScore(String playerName, int score) {
46             this.playerName = playerName;
47             this.score = score;
48         }
49
50         public String getPlayerName() {
51             return playerName;
52         }
53
54         public int getScore() {
55             return score;
56         }
57
58         public void setScore(int score) {
59             this.score = score;
60         }
61
62         @Override
63         public String toString() {
64             return playerName + ": " + score;
65         }
```

4.5. Tratamento de Exceção:

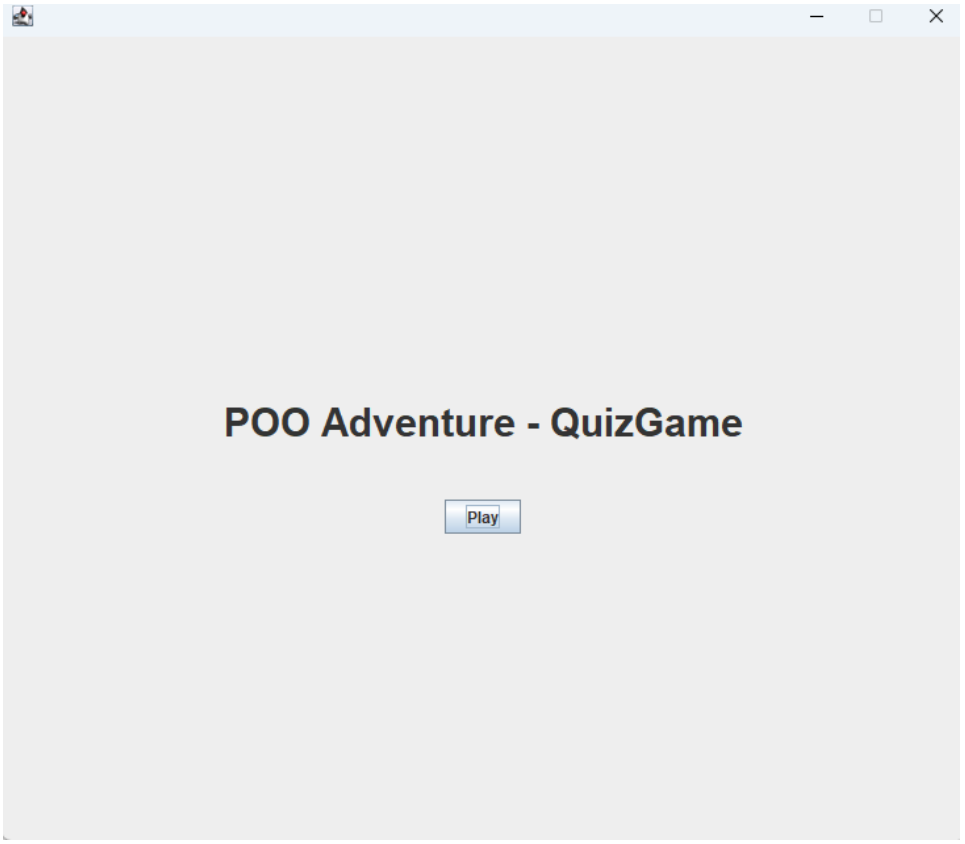
- Há um tratamento de exceção na classe *Quiz* ao salvar a pontuação no arquivo (*salvarPontuacaoNoArquivo*). O *IOException* é tratado usando *try* e *catch*.

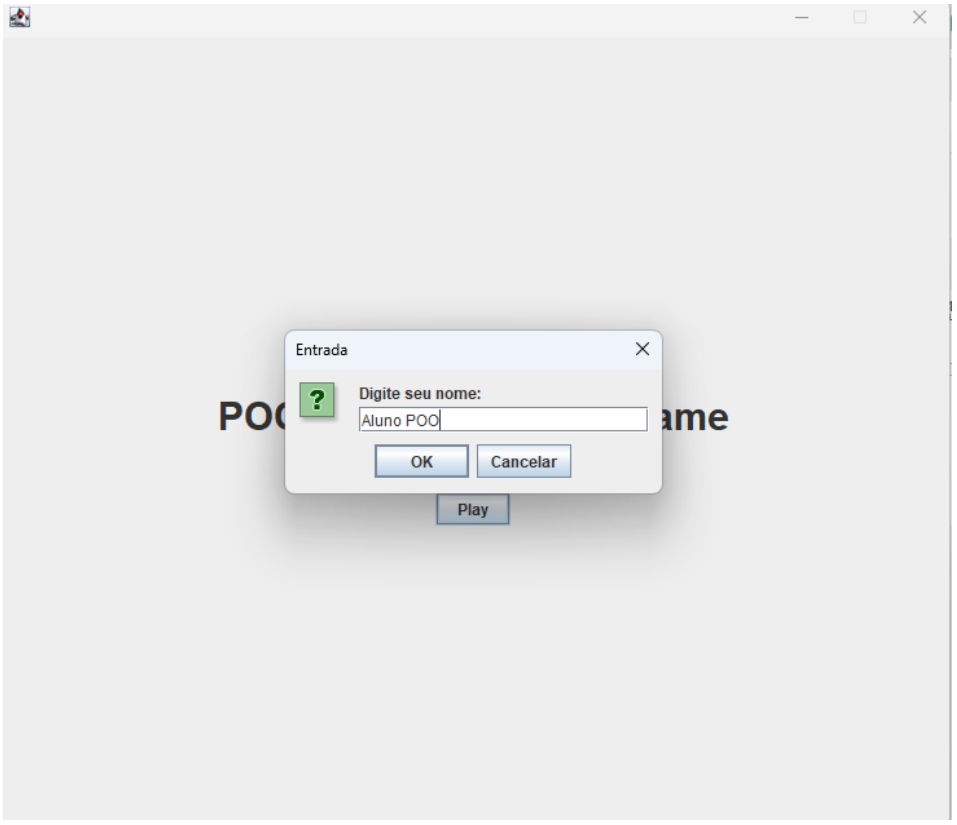
```
239
240    protected void salvarPontuacaoNoArquivo() {
241        try (PrintWriter writer = new PrintWriter(new FileWriter("D:\\Temp\\ws-eclipse\\Quiz\\src\\pontuacao.txt", true))) {
242            writer.println("Nome: " + nomeUsuario + ", Pontuação: " + pontuacao);
243        } catch (IOException e) {
244            e.printStackTrace();
245        }
246    }
247
```

4.6. Interface Gráfica com o Usuário (GUI):

- A interface gráfica está implementada na classe *Quiz*, especialmente no método *criarGUI*. Ela utiliza a biblioteca *Swing* para criar uma interface interativa.

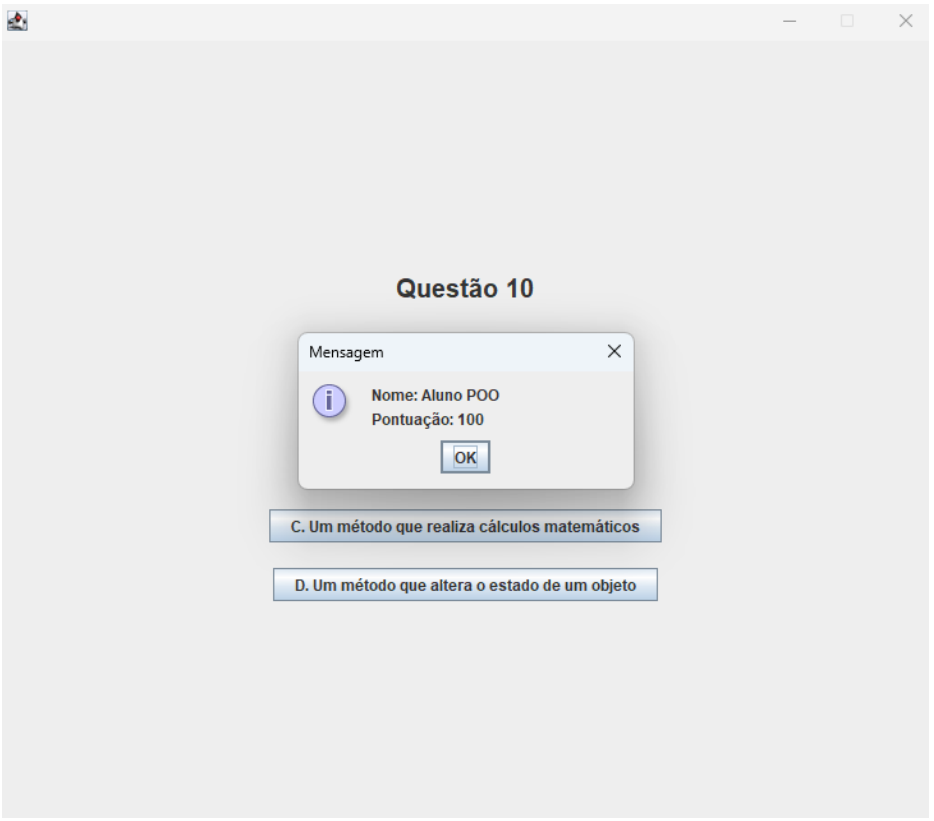
```
112    protected void criarGUI() {
113        frame = new JFrame("");
114        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
115        frame.setSize(750, 650);
116        frame.setResizable(false);
117
118        panel = new JPanel(new GridBagLayout());
119
120        JLabel titleLabel = new JLabel("POO Adventure - QuizGame");
121        titleLabel.setFont(new Font("Roboto", Font.BOLD, 30));
122
123        GridBagConstraints titleConstraints = new GridBagConstraints();
124        titleConstraints.gridx = 0;
125        titleConstraints.gridy = 0;
126        titleConstraints.insets = new Insets(50, 0, 30, 0);
127        panel.add(titleLabel, titleConstraints);
128
129        JButton playButton = new JButton("Play");
130
131        GridBagConstraints buttonConstraints = new GridBagConstraints();
132        buttonConstraints.gridx = 0;
133        buttonConstraints.gridy = 1;
134        buttonConstraints.insets = new Insets(10, 0, 10, 0);
135        panel.add(playButton, buttonConstraints);
136
137        buttonConstraints.gridy = 2;
138
139        frame.add(panel, BorderLayout.CENTER);
140
141        frame.setVisible(true);
```


POO Adventure - QuizGame
Play


Entrada
Digite seu nome:
Aluno POO
OK
Cancelar
Play


Questão 1
Quais desses não está entre os pilares de POO?
A. Herança
B. Polimorfismo
C. Abstração
D. Interface


Questão 10
Mensagem
Quiz concluído! Pontuação: 100
OK
C. Um método que realiza cálculos matemáticos
D. Um método que altera o estado de um objeto



Esses critérios estão distribuídos ao longo do código em diferentes partes, conforme mencionado acima. O código demonstra uma estrutura orientada a objetos, encapsulamento, tratamento de exceções e interação com o usuário por meio de uma interface gráfica.

5. Arquitetura do Jogo

A arquitetura do jogo segue o padrão Model-View-Controller (MVC). Abaixo destacamos uma breve descrição de cada parte:

Model (Jogo, Jogador, PlayerScore):

- `Jogo` é a classe abstrata que define a estrutura básica do jogo, contendo métodos abstratos que são implementados em suas subclasses.
- `Quiz` é a implementação concreta do jogo, estendendo a classe abstrata `Jogo` e fornecendo implementações específicas para os métodos abstratos.
- `Jogador` representa um jogador, implementando a interface `Pontuavel` para obter pontuação e nome.
- `PlayerScore` é uma classe interna usada para representar a pontuação do jogador, mantida pelo sistema de placar.

View (Interface Gráfica):

- A interface gráfica do usuário (GUI) é construída usando a biblioteca Swing do Java.
- A janela principal (`JFrame`), painéis (`JPanel`), rótulos (`JLabel`), botões (`JButton`), etc., são utilizados para criar uma interface interativa.

Controller (Event Listeners):

- A lógica de controle é gerenciada pelos event listeners, especialmente no método `actionPerformed`.
- Botões, como "Play" e "Próxima Pergunta", têm listeners que respondem aos eventos do usuário, desencadeando ações correspondentes.

6. Principais Funcionalidades

I. Iniciar Jogo:

- O jogo começa quando o usuário clica no botão "Play".
- O jogador é solicitado a inserir seu nome.

II. Apresentação de Perguntas:

- O jogo apresenta uma série de perguntas sobre programação orientada a objetos (POO).
- Cada pergunta é exibida na interface gráfica.

III. Escolher Respostas:

- O jogador escolhe uma das opções de resposta para cada pergunta.

IV. Pontuação:

- A pontuação é calculada com base nas respostas corretas.
- Cada resposta correta contribui para a pontuação total.

V. Exibição de Resultados:

- Ao final do jogo, a pontuação do jogador é exibida na interface gráfica.
- Uma mensagem informa ao jogador sobre o término do quiz.

VI. Placar de Líderes:

- As pontuações dos jogadores são salvas em um arquivo.
- Uma função de placar de líderes exibe os melhores resultados.

VII. Tratamento de Exceções:

- Exceções relacionadas à entrada do usuário e manipulação de arquivos (IOException) são tratadas.

VIII. Interface Gráfica Interativa:

- A GUI permite uma interação amigável com o usuário, incluindo botões, caixas de diálogo e atualizações dinâmicas.

Essas funcionalidades fornecem uma experiência completa de jogo de Quiz educativo, incentivando os jogadores a testarem seus conhecimentos em programação orientada a objetos.

Conclusão

O desenvolvimento do "POO Adventure - QuizGame" foi concluído com êxito, proporcionando uma experiência educativa para os usuários e desafiadora para a equipe que desenvolveu o jogo.. O projeto demonstra a capacidade de criar jogos interativos em Java, explorando conceitos de POO.