JOGO DA VELHA

11052215-Gabriel dos Reis Camargo do Amaral

29 de Julho de 2018

Turma: A1 Matutino

1 INTRODUÇÃO

Neste trabalho foi desenvolvido um jogo da velha utilizando a linguagem Java. Para a realização deste projeto foram utilizadas as seguintes ferramentas: Notepad++ (utilizado para digitar o código fonte), GIMP 2.10.4(utilizado para realização das edições de imagens necessárias para o design do projeto), jdk1.8.0_77.(utilizado para compilar o código).

O projeto segue as regras clássica do jogo de velha apresentando dois modos diferentes de jogo: um modo para jogar com um amigo e outro para jogar contra o "computador". Foi necessário desenvolver uma inteligência artificial para o modo contra o "computador", a inteligência foi implementada utilizando o método minimax ." Em teoria da decisão, o minimax (ou minmax) é um método para minimizar a possível perda máxima" (WIKIPEDIA, 2018).

Para representação gráfica foi utilizada uma GUI desenvolvida utilizando o javax.swing e alguns elementos do java.awt. De forma sucinta, exista uma tela inicial no qual o usuário escolhe o modo de jogo (solo ou com um amigo), caso decida jogar com um amigo ele é direcionado diretamente para o tabuleiro após digitar os nomes dos jogadores. Caso decida jogar contra o "computador" será direcionado para uma tela na qual será necessário escolher o nível de dificuldade (dentre: fácil, médio ou impossível). O tabuleiro é um conjunto de nove botões que ao serem clicados dependendo de quem é a vez ele adiciona um X ou O ao botão.

O principal objetivo desse trabalho foi desenvolver os conceitos aprendidos na aula de Programação Orientada a Objeto para a realização de uma aplicação pratica, além de adquirir conhecimentos sobre GUI.

2 DESCRIÇÃO DAS CLASSES

Será apresentado uma breve descrição das classes que foram desenvolvidas neste trabalho:

- <Principal>: classe responsável pela execução do programa, faz as devidas chamadas para o funcionamento pleno do programa.

- <TelaInicial>: primeira tela a ser mostrada para o usuário, contém a escolha que usuário deve fazer referente ao modo de jogo que deseja jogar.

- <TelaDificuldade>: tela que contém a escolha que o usuário deve fazer referente ao nível de dificuldade que a inteligência artificial adotará.

- <TelaJogo>: tela que contém o tabuleiro, placar e outros elementos para o funcionamento visual do jogo propriamente dito.

- <Botao>: classe que define o comportamento dos botões do tabuleiro do jogo.

- <Tabuleiro>: classe que define o tabuleiro do jogo. Além de funções básicas do jogo como se a condição de vitória foi atingida.

- <Inteligencia>: classe que define o comportamento da inteligência artificial utilizada para realizar as jogadas do "computador".

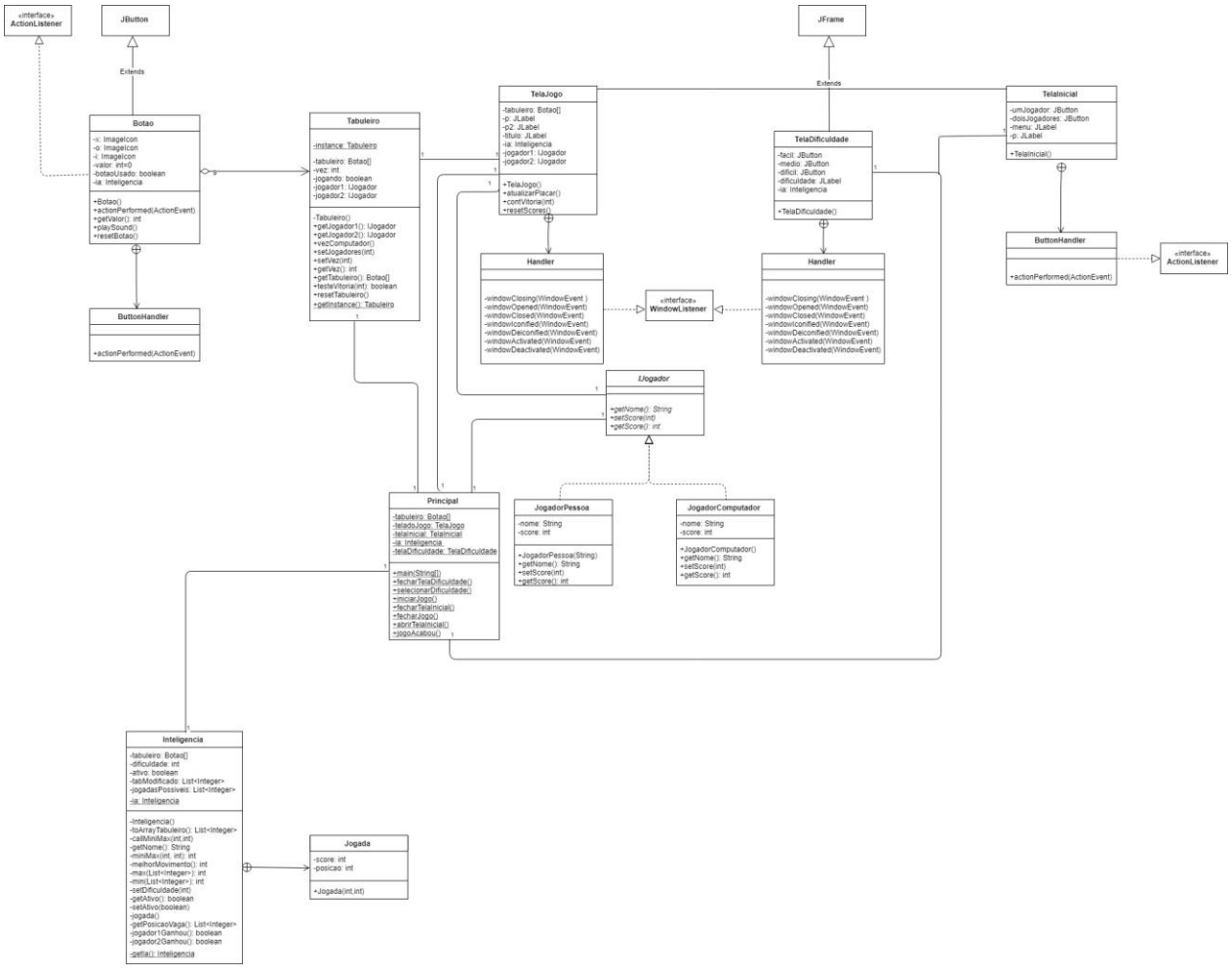
- <IJogador>: interfase utilizada para utilização do padrão *Strategy* para as classes de jogadores.

- <JogadorPessoa>: classe que define o comportamento de jogadores humanos.

- <JogadorComputador>: classe que define o comportamento do "computador".

Abaixo a figura 01 representa o digrama de classes UML deste trabalho:

Figura 1: diagrama UML



Fonte: (Autor, 2018).

Para melhor visualização do diagrama, a imagem da figura 1 foi anexada junto com o relatório.

3 Conceitos de orientação a objeto utilizados

Para a realização deste trabalho foram utilizados os seguintes conceitos de Orientação a Objeto:

- Herança: utilizado para estender JFrame e JButton.
- Interfase, Sobrescrita e Polimorfismo: para utilização do padrão *Strategy*.
- Classe aninhada: para realizar classes Handler para lidar com o evento de apertar o botão.
- Modificadores de acesso: para manter o encapsulamento do código.
- Encapsulamento: para melhor a forma estrutura do código.
- Abstração: classe que define os comportamentos das instancias.
- Métodos modificadores: get/set para melhorar o encapsulamento.
- Construtores: para garantir que atributos são inicializados.
- Tratamento de exceção: utilizado para o som que o botão faz quando clicado.
- Padrões de projeto utilizados: *Strategy* e *Singleton* (utilizado para garantir que exista uma única instância de Tabuleiro e Inteligência).
- Método e atributo de classe (static): utilizado com maior frequência na classe Principal do programa já que ela não é instanciada.

4 Participação do grupo

Responsável por todos os processos desde a modelagem até a implementação final: Gabriel Amaral. Para reforçar como os padrões *Singleton* e *Strategy* foram utilizados:

Singleton: utilizado para garantir que exista unicamente uma instância de Tabuleiro e uma de Inteligência. Para isso foi desenvolvido um método getInstance nas classes Tabuleiro e Inteligência e também o construtor deles foi definido com o modificador private para garantir que nenhuma outra classe os instancie (para maiores detalhes vide diagrama UML ou código fonte disponibilizado).

Strategy: utilizado para facilitar o tratamento das classes JogadorPessoa e JogadorComputador.