ADDETC – Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores

LEIM -Licenciatura Engenharia informática e multimédia

Modelação e Programação

Trabalho final

Turma:

LEIM-23D

Trabalho realizado por:

Miguel Távora Nº45102

Docente:

João Ventura

Data: 10/07/2021

Índice

Índice ilustrações

1. Introdução

O projeto final de Modelação e Programação tem como objetivo a criação de uma aplicação escolhida livremente pelo aluno. No caso deste projeto foi escolhido o jogo de tabuleiro Damas, onde as regras seguidas são as regras seguidas em Portugal. O desenvolvimento da aplicação deve ser feita de tal maneira que seja separada a parte lógica da parte gráfica da aplicação. O jogo possui as seguintes regras:

- Só é possível mover uma peça de cada vez
- O movimento das peças tem de ser sempre na diagonal e para a frente no caso das peças
- Quando uma peça chega á ultima posição do campo adversário é promovida a dama
- A movimento da dama é feito de tal maneira que pode movimentar-se livremente nas diagonais do jogo desde que as posições estejam vazias
- Para uma peça comer a outra é necessário que a peça adversária esteja na próxima posição para a qual a peça se pode movimentar e a próxima posição na mesma diagonal esteja vazia e deve ser sempre para a frente
- Uma dama pode comer nas diagonais livremente, contudo após comer uma peça só pode comer outras peças se estiver na vizinhança de 4 das diagonais e se a próxima posição estiver vazia, sempre que a dama come outra peça deve ficar na posição imediamente após a posição da peça adversária que comeu
- Para uma peça comer em cadeia esta deve comer a primeira peça e após comer a primeira deve haver uma nova peça inimiga nas diagonais da frente e no seguimento da diagonal a posição estar vazia, a peça deve movimentar-se sempre para a frente
- As peças brancas devem ser sempre as primeiras a começar o jogo
- O jogador vence quando o adversário não tiver mais peças para jogar ou não conseguir realizar nenhuma jogada

2. Desenvolvimento

2.1 Motor

2.1.1 Interface Piece

O motor encontra-se no package board, no seu interior existem duas classes que são a classe Board e a classe Engine. A classe Board representa o tabuleiro do jogo, o tabuleiro do jogo é um array bidimensional sobre uma interface designada Piece. A interface Piece obriga a implementação de quatro métodos um método de identificação do tipo da peça, um método para saber quais as posições de uma peça/dama pode comer, as posições que uma peça/dama pode movimentar-se e um método para afetar as coordenadas corrente da peça.

Existem cinco classes que implementam a interface sendo elas: BlackPiece, Empty, Queen, Unplayable, e WhitePiece. Para as classes BlackPiece e WhitePiece a principal diferença entre elas é o facto de as peças brancas se movimentarem sempre para cima e as peças pretas para baixo no array bidimensional. A classe Empty representa uma posição vazia no array, ou seja, que é possível movimentar-se para lá. A classe Unplayable é a classe no qual não é possível jogar. Tanto a classe Unplayable como Empty definem os métodos da interface mas só implementam o método *getPositionType*. Por fim a classe Queen representa a dama.

O método *getPositionType* é o método utilizado para saber qual é o tipo do objeto numa determinada posição do array. Os identificador seguem a seguinte tabela:

Classe	Identificafor
Unplayable	-1
Empty	0
WhitePiece	1
BlackPiece	2
WhiteQueen	3
BlackQueen	4

Tabela 1 - Identificação das instâncias

O método *getEatablePositions* retorna numa lista as posições que a determinada instância pode comer no formato de String. A String possui o seguinte formato: “posX;posY” onde posX é a posição nas colunas e posY é a posição nas linhas. Através do método split no caracter “;” e da conversão de String para inteiro é possível as duas coordenadas. Cada índice da lista é a posição final da peça após comer a peça inimiga. Caso a lista tenha tamanho superior a 1 quer dizer que a peça pode comer de duas formas diferentes, caso não possa comer o tamanho da lista será 0.

O método *getMovePositions* é o método para obter as posições para a qual as peças podem movimentar-se, mas sem comer outra peça. Da mesma maneira que o método *getEatablePositions* retorna uma lista de Strings e a obtenção das coordenadas é feita da mesma maneira. Este método só deve ser chamado caso não exista nenhuma peça para comer. Caso esta lista dê tamanho 0 para todas as peças quer dizer que o jogador da peça perdeu.

Por fim o método *setPosition* é utilizado para afetar a posição nas colunas e nas linhas da instância da peça sempre que ela se move. É necessário este método para quando está a obter as posições para comer ou movimentar a peça esta saber qual a coordenada á qual está associada.

Para fazer a pesquisa de comer ou movimentar uma peça todas as peças recebem como argumento a instância da classe Board para ter acesso ao atributo board que é o array bidimensional de Piece.

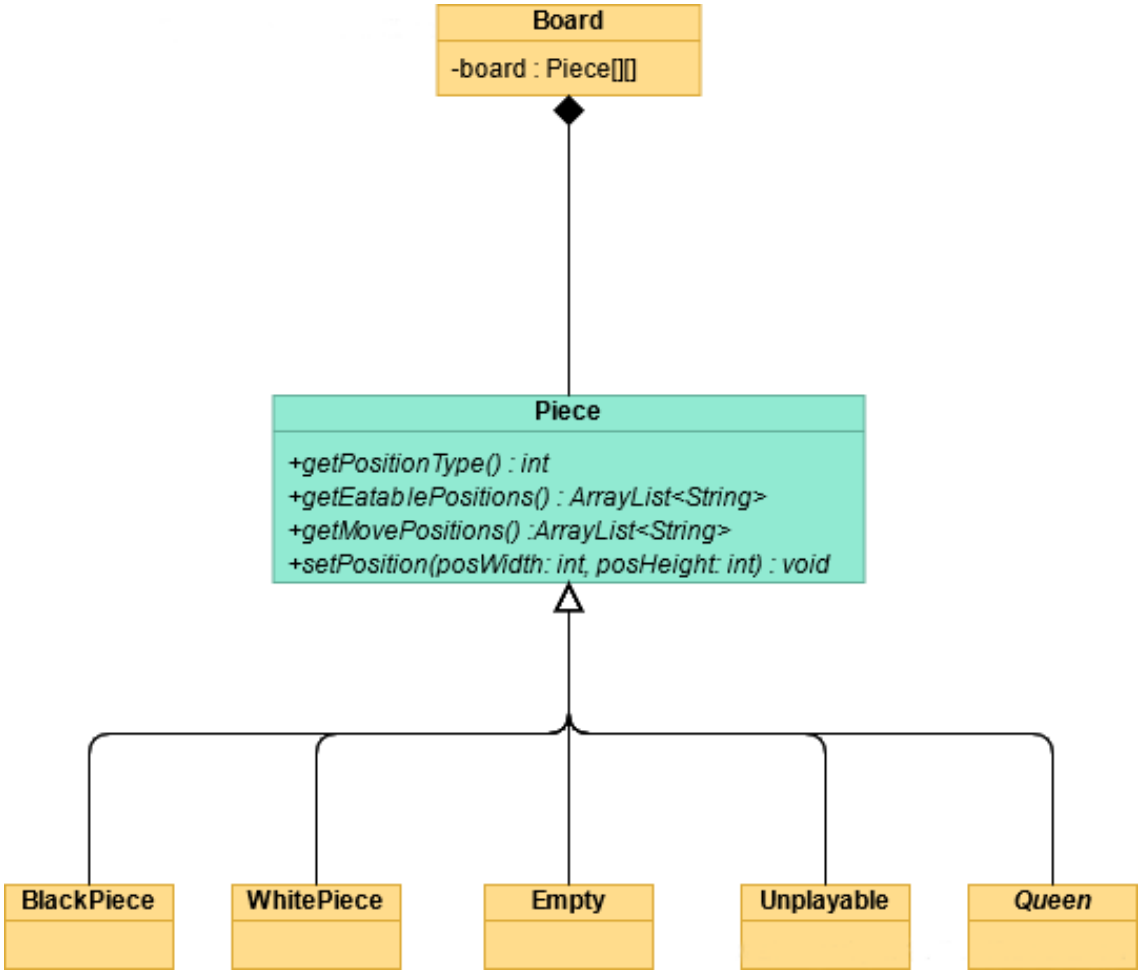

Figura 1 - Diagrama classes da interface Piece

2.1.2 Classe abstracta Queen

A classe Queen é classe que representa uma dama, esta classe é abstracta e implementa a interface Piece. Esta classe surge no facto de uma dama é quase igual entre uma peça preta ou branca a única diferença é que uma come peças pretas e outra come peças brancas. Nesse facto a classe possui três métodos abstractos o método *getPositionType* que será diferente para a peça branca e preta, *getAdversaryPieceType* que é o identificador da peça adversária e o *getAdversaryQueenType* é o identificador da dama adversária. Desta forma existem duas classes que estendem de Queen que é a WhiteQueen e a BlackQueen. Ambas as classes são simples onde chamam a super classe no construtor, declaram o seu identificador e qual o identificador tanto da peça como da dama aversária. A classe Queen possui ainda um método extra que é o *getSurroundings* que é chamado após uma dama ter comido outra peça, onde verifica na vizinhança de 4 nas diagonais se é possível comer outra peça.

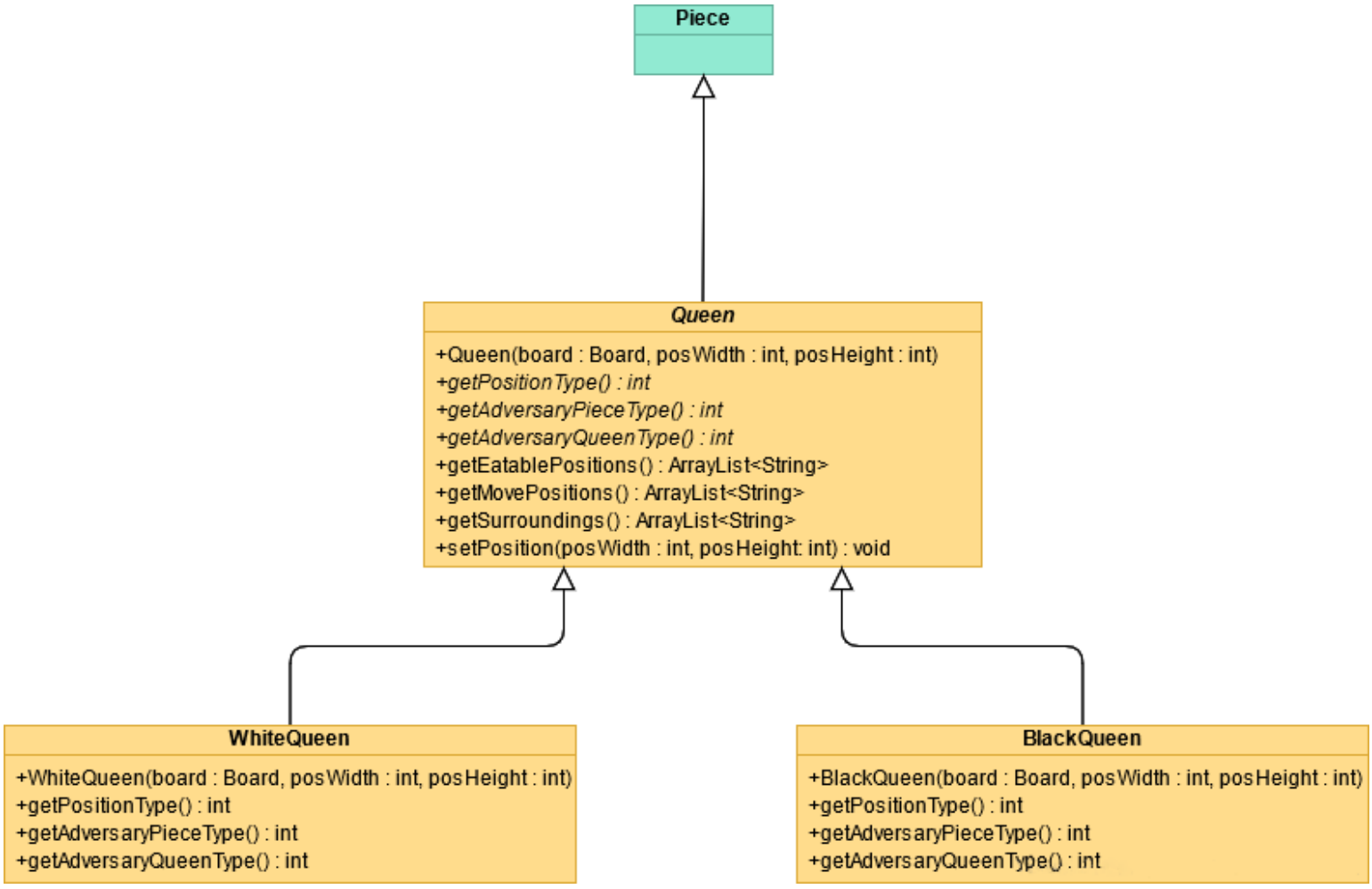

Figura 2 -Diagrama classes da classe Queen

2.1.3 Classe Board

A classe Board é a responsável por todas as operações no array do tipo Piece. Esta classe cria o array com o tamanho dos atributos *width* and *height* da classe, remove as peças do tabuleiro quando são comidas, movimenta as peças no tabuleiro e promove peças a damas. A partir dos métodos implementados nas classes que implementam a classe Piece a classe consegue obter as peças que podem comer outras e movimentar-se, esta estrutura de dados é um HashMap onde a key é a posição currente da peça e o valor é a lista obtida pelos métodos *getEatablePositions* ou *getMovePositions*. Por fim a classe tem um método designado *getMostEatablePositions* onde a partir dos mapas referidos anteriormente e dos métodos implementados de cada peça é possível obter todos os caminhos que uma peça pode fazer para comer outras peças. Isto é feito através de um mapa onde se removendo caminhos já explorados e adicionando caminhos ainda não explorados. Por fim é obtidos uma lista com todos os caminhos possíveis de realizar de uma peça para comer outras peças inimigas. Além destes métodos possui alguns métodos *get* para obter atributos ou de ajuda a outros métodos.

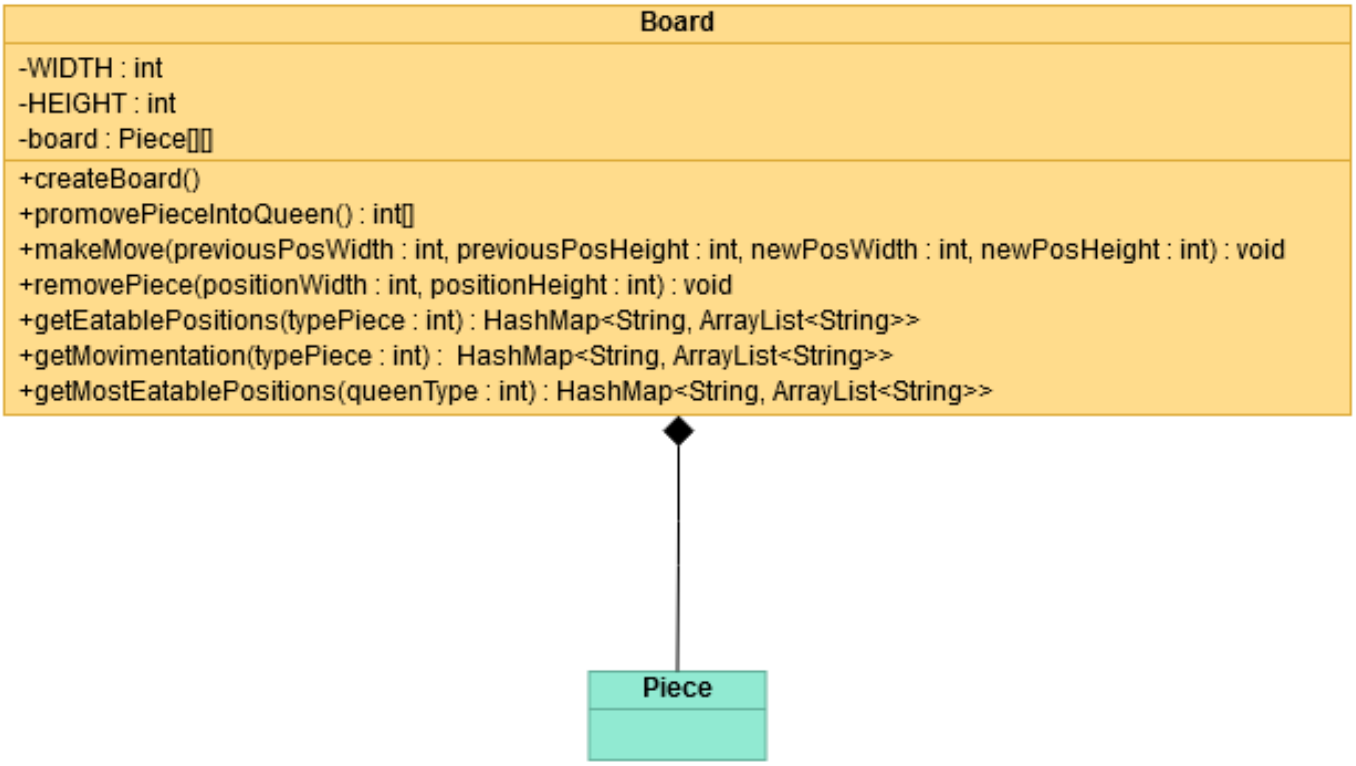

Figura 3 - Diagrama de classe da classe Board

2.1.4 Classe Engine

A classe Engine é uma classe de encapsulamento da classe Board, onde possui métodos que essencialmente chama métodos da classe Board com os devidos argumentos de forma a facilitar a utilização da classe Board e de forma a esconder toda a implementação desta classe, por ser também utilizada pelas classes que implementam Piece.

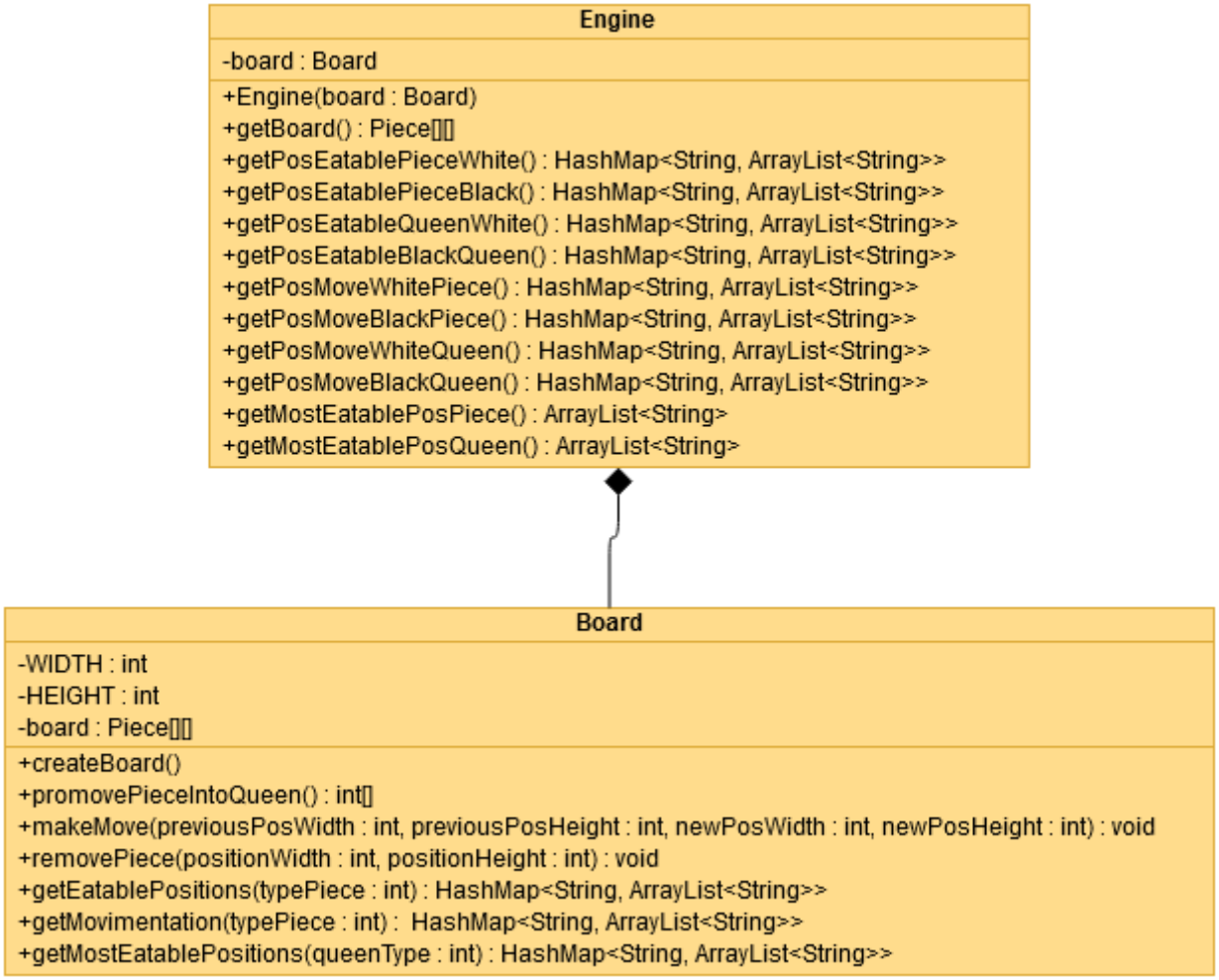

Figura 4 - Diagrama classes da classe Engine

2.2 Inteligência Artificial

A classe que representa a inteligência artificial é designada AI esta classe utiliza a classe Random para geração de número aleatórios para escolher valores. Esta classe possui então três métodos: *selectMovePosition, selectEatPositionRandom* e *selectEatPositionBiggerValue*.

O método *selectMovePosition* escolhe caso seja possível mover peças e damas metade da probabilidade para mover uma peça outra metade para mover uma dama. A partir dos mapas passados como argumento obtem as peças que é possível realizar movimento e das peças dentro do mapa escolhe uma aleatóriamente e caso seja possível realizar mais do que um movimento escolhe um aleatóriamente.

O método *selectEatPositionRandom* a partir das listas passados como agumento escolhe uma lista aleatoriamente sendo as listas das peças e das damas, caso estas tenhas tamanho superior a 0. Após a escolha da lista é escolhido um índice aleatoriamente, podendo ser o movimento que come mais peças ou não.

Por fim o método *selectEatPositionBiggerValue* escolhe sempre o índice entre todas as listas de movimentos que come mais peças. A prioridade para quando são de tamanho igual entre damas e peças é escolhida sempre as peças.

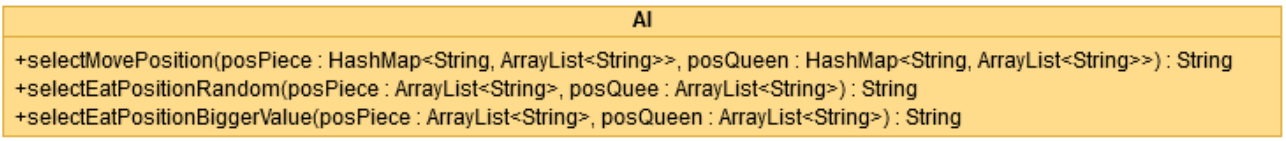

Figura 5 - Diagrama classes da classe AI

2.3 Interface gráfica

A interface gráfica foi desenvolvida segundo a bilbioteca Swing do Java. Esta biblioteca permite representar gráficamente botões entre outros elementos gráficos em forma de objetos. Para realizar esta interface é necessário existir uma classe que seja uma JFrame ou que possua um atributo do tipo JFrame. A JFrame é a classe responsável pelo tamanho da janela, nome da janela, a disposição dos elementos no layout entre outros.

A partir de um JPanel é possível então desenhar elementos na interface gráfica por exemplo imagens e também a deteção de clicks. A interface gráfica implementada é constituida de uma JFrame e duas JPanels. Os JPanels serão então o menu e o jogo propriamente dito.

A classe que é uma JFrame é designada MainFrame e é responsável então por criar a janela afetar o seu tamanho inicial e fazer a troca entre os dois JPanel. Esta classe recebe como argumento um objeto do tipo Engine e do tipo AI, contudo serve para depois enviar para o painel do jogo.

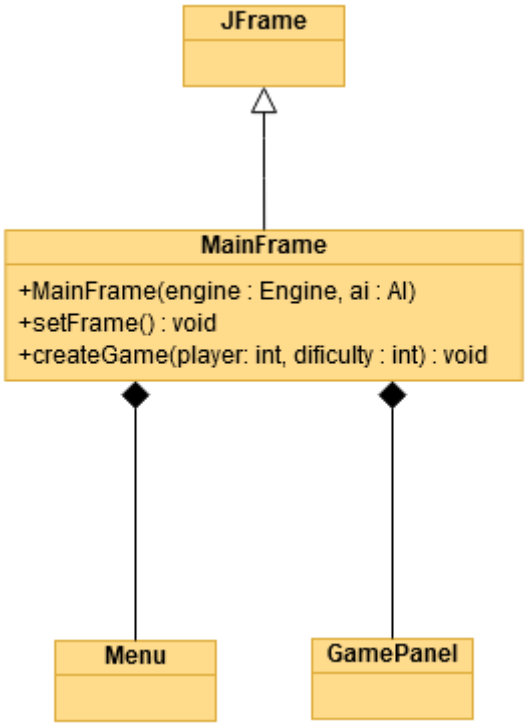

Figura 6 - Diagrama classes da classe MainFrame

2.3.1 Painel do Menu

O menu é uma classe que estende de JPanel e implementa MouseListener. A partir da reescrita do método paintComponent é possível desenhar componentes na interface onde no caso da implementação foi desenhado uma imagem de fundo e imagens de botões. Esta interface é possível aumentar e diminuir conforme o utilizador assim o desejar, onde a partir de uma referencia para a MainFrame é possível saber o tamanho atual do painel e assim é possível ajustar os elementos sempre no centro do painel. Este menu utiliza também uma classe auxiliar designada FilesObtainer que permite obter os ficheiros das imagens, da música, métodos de *get* para as imagens e lançar a música como Thread. O menu possui então o aspeto que se segue:

Figura 7 - Interface gráfica do menu

O menu é constituído então de 4 images que representam botões, através da implementação do MouseListener é possível saber se o utilizador clicou numa das imagens e assim realizar uma ação de acordo. O botão Exit termina o processo, o botão Rules mostra as regras do jogo em inglês como é mostrado na figura que se segue:

Figura 8 - Interface gráfica do menu a mostrar as regras

O botão Play music começa música lançada como Thread e muda o botão de Play music para Stop music, caso o utilizador clique em Stop music a música para e a Thread termina.

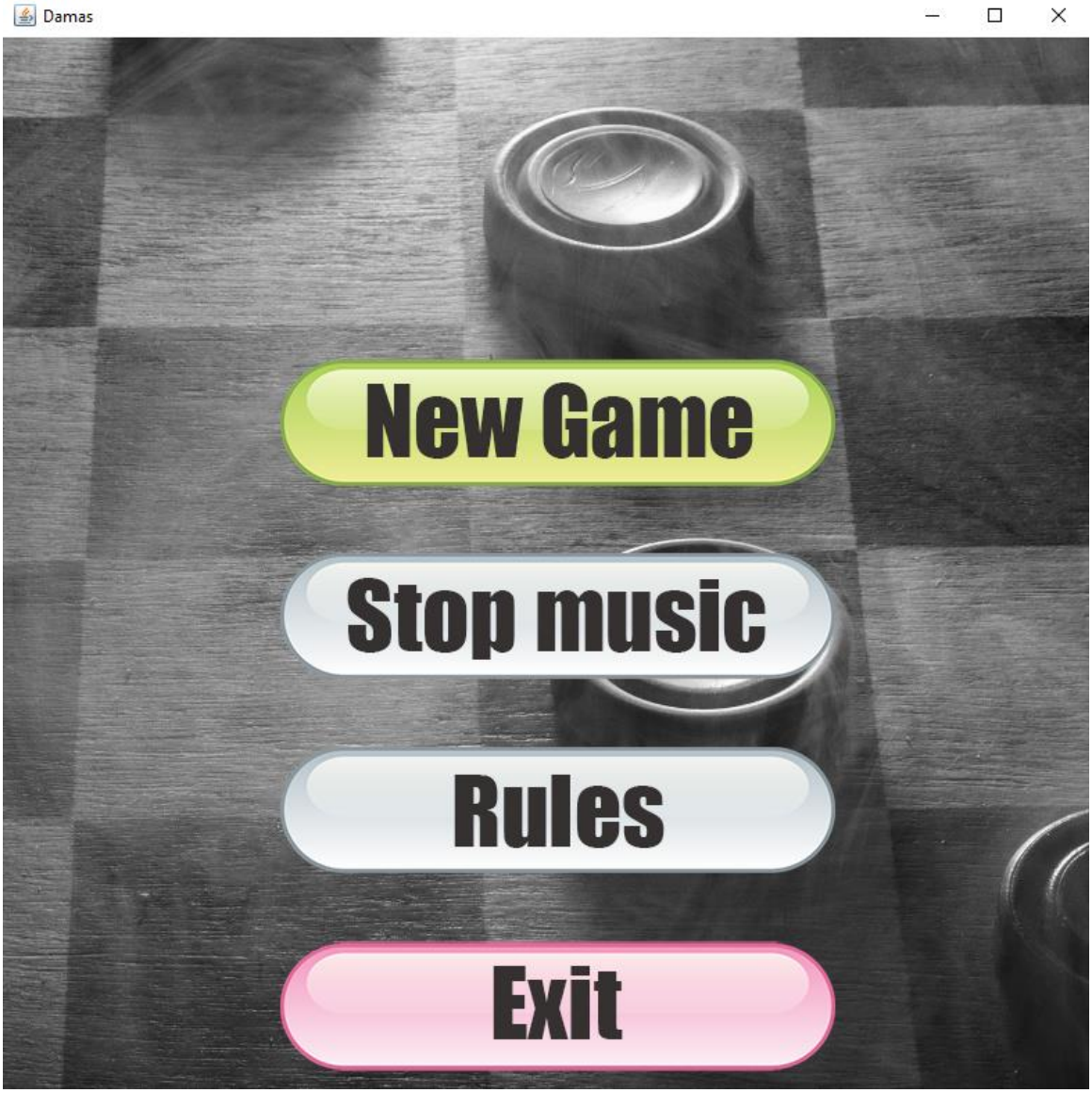

Figura 9 - Interface gráfica do menu quando a música está a tocar

Por fim o botão New Game mostra uma janela para o utilizador ser o primeiro a jogar ou o segundo, onde caso seja o segundo fica com as peças pretas e caso inicie ele fica com as vermelhas.

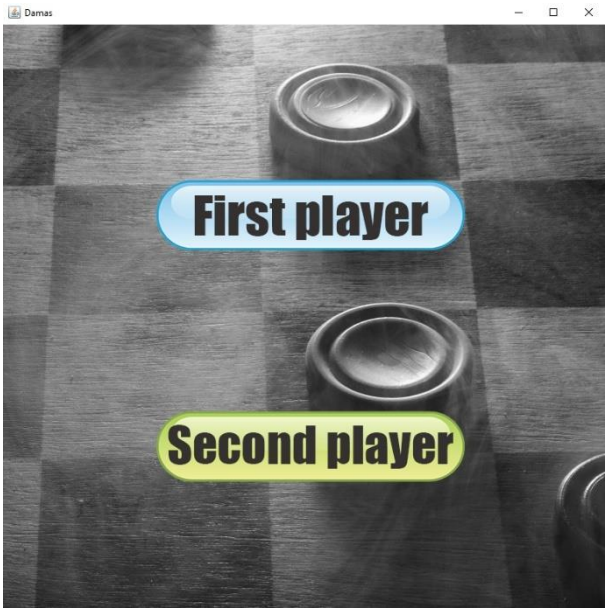

Figura 10 - Interface gráfica menu seleção da ordem do jogador

De seguida é pedido para selecionar a dificuldade, onde a dificuldade iniciante a inteligência artificial seleciona aleatóriamente uma peça para comer e no nível médio escolhe sempre o caminho onde come mais peças, como foi referido na classe AI.

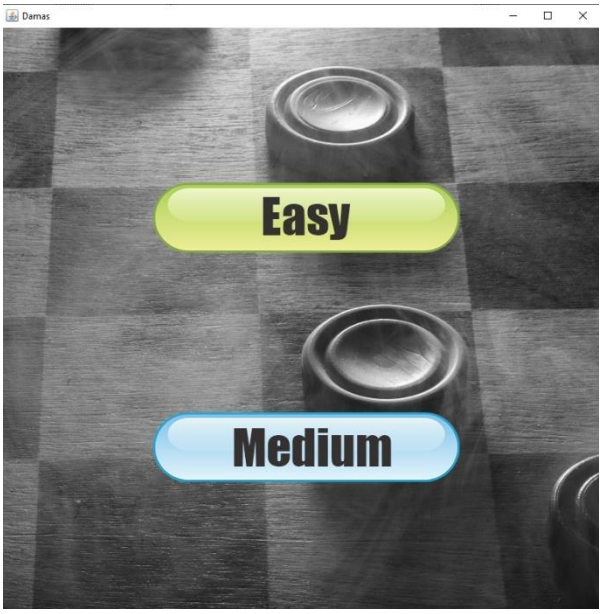

Figura 11 - Interface gráfica menu seleção da dificuldade

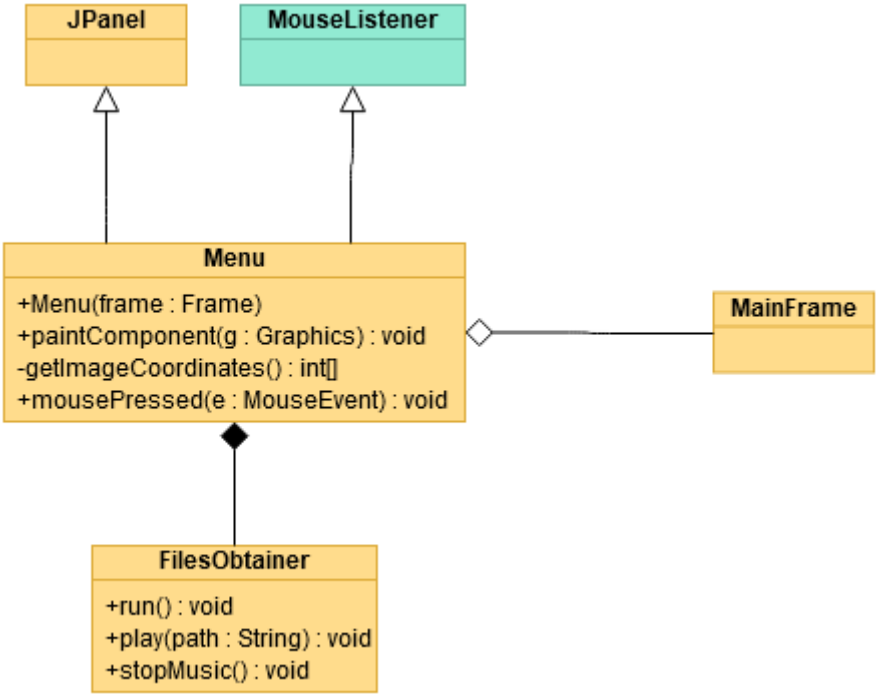

Figura 12 - Diagrama de classes da classe Menu

2.3.2 Painel do Jogo

O painel do jogo é feito pela classe GamePanel esta classe também á semelhança da classe Menu estende de JPanel e implementa MouseListener. Esta classe possui uma referência para a classe MainFrame também para conseguir centrar sempre o jogo no centro da janela. Esta classe possui uma referência para a classe Engine e para a classe AI de forma a conseguir através da interface gráfica realizar operações no motor do jogo. A classe Engine serve para obter a estrutura do tabuleiro, movimentos possíveis entre outros e a classe AI para obter a movimentação das peças pelo computador. Esta classe possui também uma classe auxiliar designada GraphicalOperations onde permite obter imagens e operações relacionadas com lógica do jogo.

A classe recebe como argumento se o jogador é o primeiro ou o segundo a jogar, quando é o primeiro ficas com as peças vermelhas e casos seja o segundo fica com as peças pretas e é feito primeiro uma jogada pela inteligencia artificial. A classe recebe também a dificuldade, onde caso seja o modo fácil utilizada o método da inteligência artificial onde a jogada utilizada é o método de comer que é sempre aleatório, enquanto se for intermédio o método de comer é sempre chamado o que come mais peças.

Toda esta mecânica é feita através de cliques do rato e através das coordenadas em píxeis do rato e do tamanho do tabuleiro é possível saber todas as posições para os quais é possível movimentar as peças.

Em relação á interface gráfica as posições que podem ser jogadas são postas dentro de um quadrado verde. Como se mostra na figura que se segue:

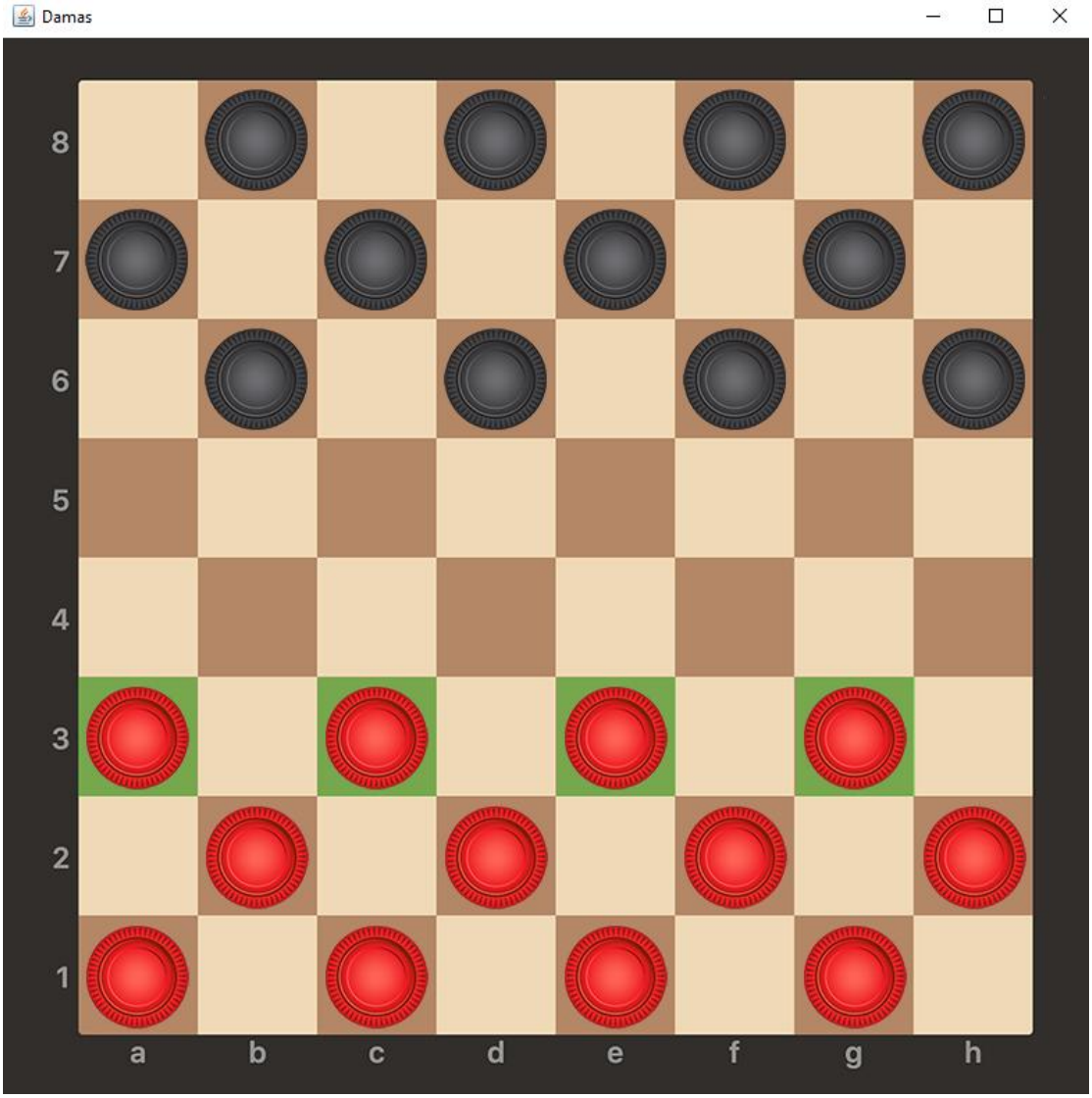

Figura 13 - Interface gráfica do jogo das posições que é possível jogar

Caso o jogador clique no quadrado aparece as posições para o qual é possível jogar com a posição selecionada. Como é mostrado na figura que se segue:

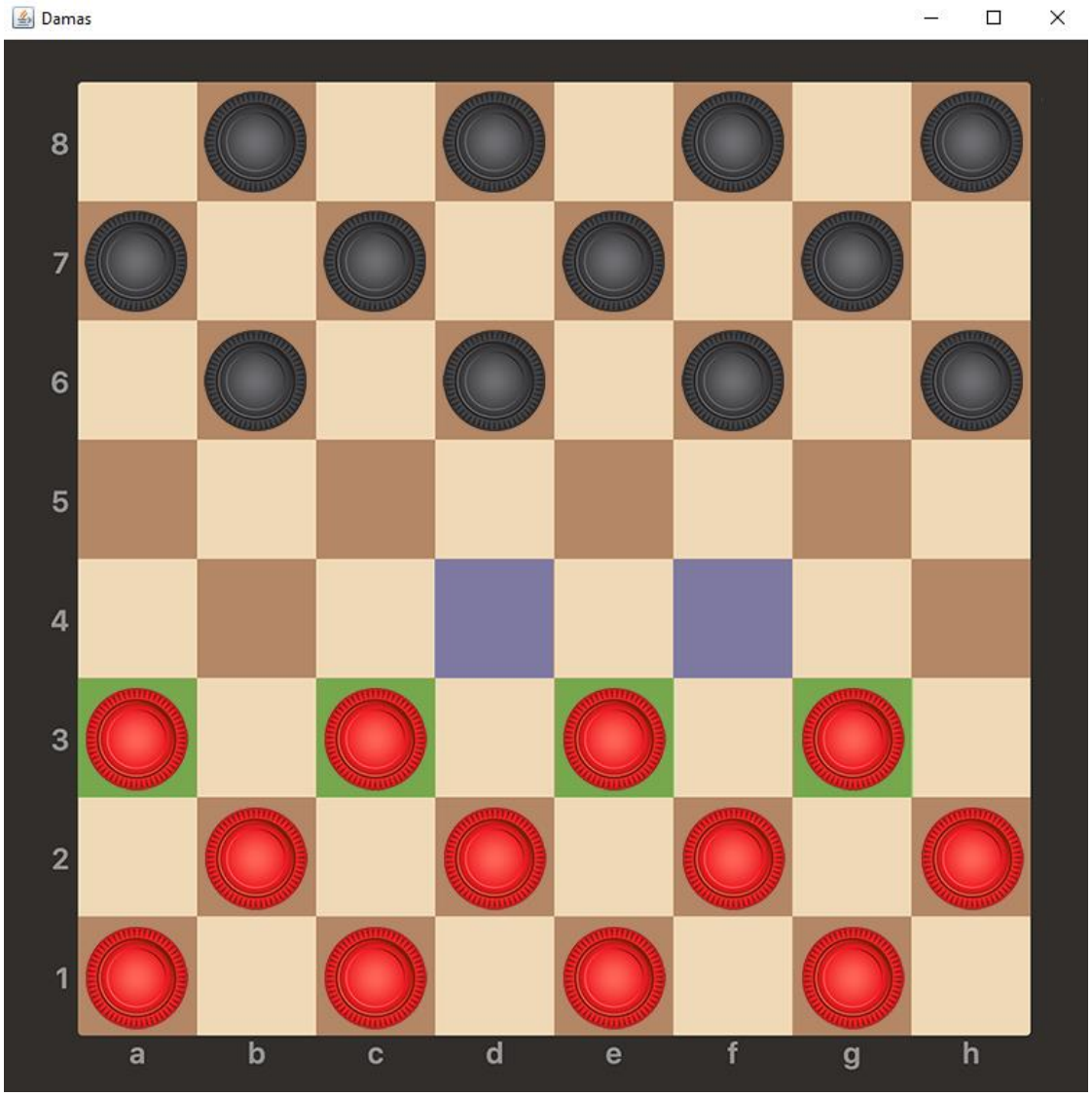

Figura 14 - Interface gráfica das posições que se pode mover

Caso o utilizador clique num dos quadrados azuis a peça irá mover-se para a posição clicada.

Caso o utilizador ganhe ou perca mostra uma mensagem de vitoria ou derrota e permite voltar a jogar.

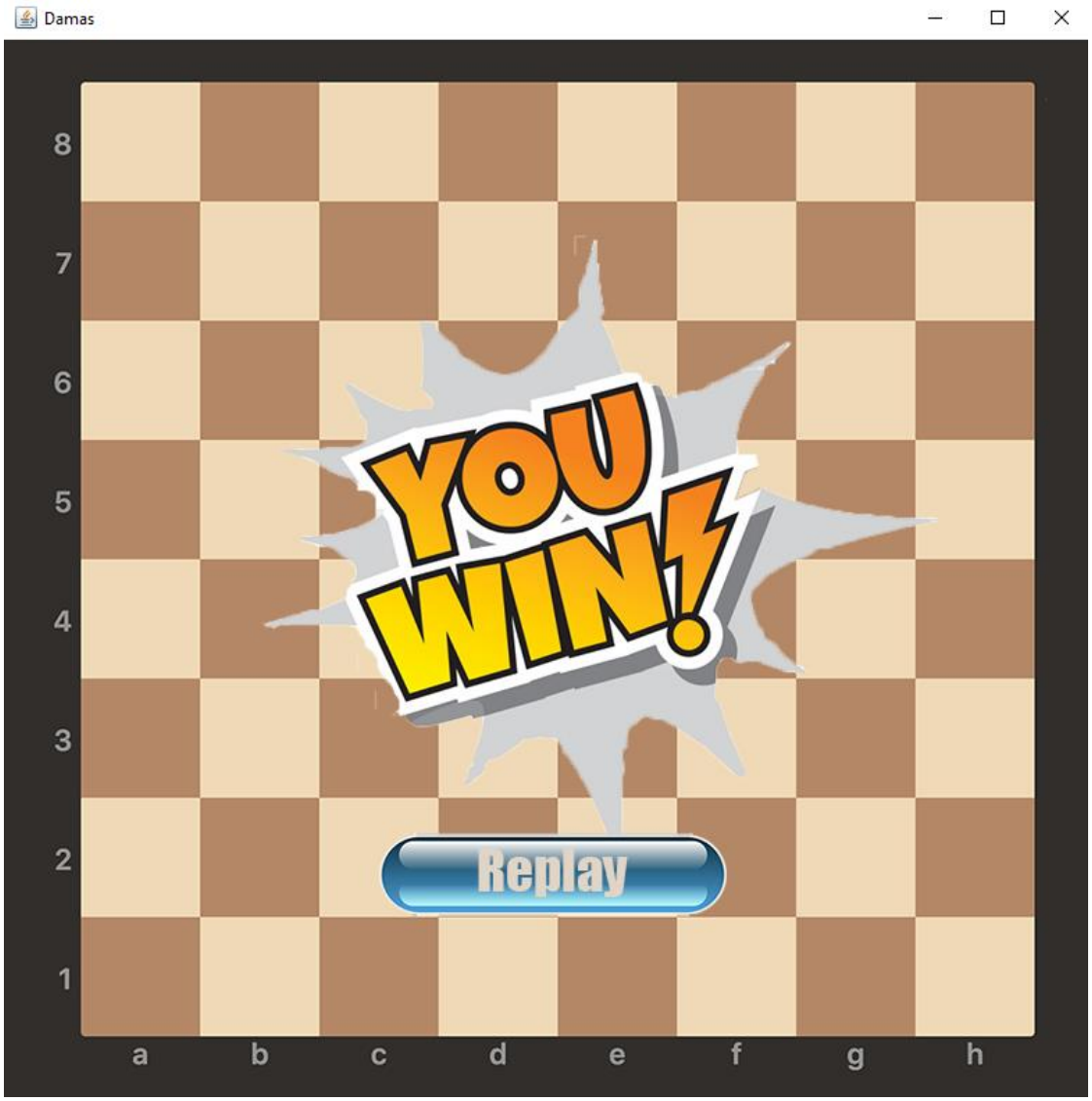

Figura 15 - Interface gráfica para a vitória do jogador

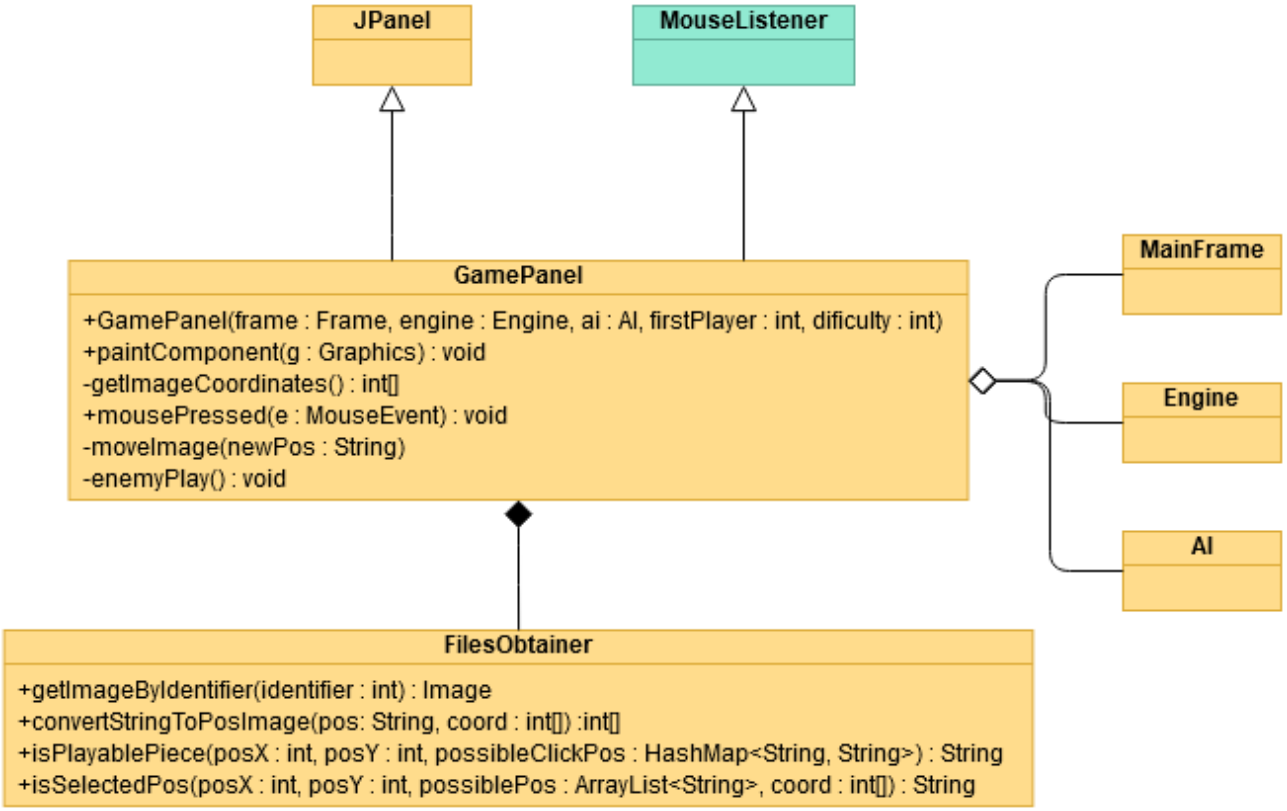

Figura 16 - Diagrama de classes da classe GamePanel

3. Conclusões

Em suma com a realização do presente trabalho prático aprendeu-se os seguintes conhecimentos:

- Modelar problemas por classes e dividir funcionalidades pelas diferentes classes.
- Utilização da biblioteca Swing para apresentar componentes gráficas.
- Utilização de Threads para tocar uma música em background.
- Utilização da classe MouseListener para deteção de cliques na interface gráfica.
- Separação da parte gráfica da parte lógica.

Em termos de implementação do sistema todo ele funciona e é possível reutilizar a parte lógica para outras implementações como por exemplo telemóvel. Em termos de interface do utilizador este possui um aspeto clássico e por isso apelativo. Todas as funcionalidades do sistema estão a funcionar na integra e tudo implementado pelo aluno.

4. Bibliografia

https://stackoverflow.com/questions/9625110/how-to-start-anonymous-thread-class

https://stackoverflow.com/questions/19125707/simplest-way-to-set-image-as-jpanel-background