Programação Orientada a Objetos

Licenciatura de Engenharia Informática

IPS Instituto Politécnico de Setúbal

Manual Técnico – Projeto Fase 2

Docente: Luís Cassaca

Alexandre Coelho, 190221093

Sérgio Veríssimo, 190221128

Breve Descrição

Este programa é um jogo de blockudoku, este é composto por 2 menus um menu inicial com 4 opções e o menu de jogo com 2 opções para escolher o modo de jogo. Nestes menus o jogador pode consultar as suas pontuações, continuar um jogo gravado, ver o top 10 de pontuações e reiniciá-lo caso seja necessário, jogar um novo jogo, escolher o modo, etc.

Classes/Interfaces implementadas

Block

Esta classe, como o nome indica, representa a criação de um objeto bloco no jogo do BlockoDoku.

Atributos

blockGameMode – Atributo do tipo GameMode, que indica o tipo de bloco, se este pertence ao modo básico, ou ao modo avançado;

Squares – Atributo do tipo ArrayList de Square, representa a quantidade de quadrados que o bloco poderá preencher.

Métodos

public Block (ArrayList<Square> squares, GameMode gameMode) – Representa o construtor desta classe, que recebe como parametro, uma ArrayList do tipo Square (ou de quadrados) e recebe um modo de jogo.

public ArrayList<Integer> getColumnsInteger(int y) – Este método verifica quais as colunas que possuem a linha recebida como parâmetro (int y). Retorna um ArrayList com o conjunto de colunas;

public GameMode getBlockGameMode() – Este método retorna o modo de jogo do bloco. Funciona como um *"getter"* para o atributo blockGameMode, referido anteriormente. Retorna o valor que se encontra no atributo blockGameMode;

public boolean compareColumns(ArrayList<Integer> columnsList, int value) – Este método compara as colunas da lista recebida como parâmetro, com o valor recebido. Retorna verdadeiro ou falso, consoante o resultado da comparação;

public int getLeftSquare(ArrayList<Square> objectsList) – Este método verifica qual é o quadrado mais a esquerda e retorna o valor inteiro da linha, representada por um Y;

public Integer getCurrentColumnIndex(ArrayList<Square> objectsList, int value) – Este método recebe uma lista de objetos e um valor, devolvendo o índice, onde se encontra uma coluna (representada por x) com o mesmo valor inserido nos parâmetros. Caso não exista nenhum valor de coluna igual ao valor passado por parâmetro, retorna null;

public String getBlockToDraw() – Como o nome indica, este método serve unicamente para gerar um bloco e devolver esse mesmo bloco;

public String toString() – Este método retorna as coordenadas X (colunas) e Y (linhas) de cada quadrado do bloco;

Board

Esta classe, serve unicamente para representar o tabuleiro, onde se vai realizar o jogo.

Atributos

header – Representa o cabeçalho do tabuleiro, representado por diversas letras identificadoras, pois as jogadas representam-se por "**A**9", por exemplo. Este atributo é do tipo ArrayList de String;

boardPositions – Este atributo, representa as posições no tabuleiro, como referido anteriormente, as jogadas representam-se por "A9", por exemplo, sendo que este exemplo representa uma posição no tabuleiro. Este atributo é do tipo Array bidimensional de Strings.

Métodos

public Board() – Representa o construtor da classe, que inicializa o tabuleiro. Dentro deste construtor são adicionados os cabeçalhos (header) e as posições no tabuleiro (boardPositions);

public String getBoardDraw() – Este método, como o seu nome indica, retorna uma String que contém o desenho do tabuleiro;

public ArrayList<Integer> checkRowPoints() – Este método percorre as linhas e verifica se estão completas. No caso de estarem completas, são adicionadas a um ArrayList de inteiros, que é retornado por este método;

public ArrayList<Integer> checkColumnPoints() – Semelhante ao método referido anteriormente, embora este, percorre as colunas e verifica se estão completas. O retorno consiste num ArrayList de inteiros, preenchido com as colunas completas;

public ArrayList<Square> checkSquarePoints() – Semelhante as anteriores, mas desta vez, este verifica se os quadrados grandes (conjunto de 3x3 no tabuleiro) estão completo. Retorna um ArrayList com os quadrados completos;

public int clearRow(int row) – Neste método, é feita a limpeza da linha que fora passada como parâmetro. Retorna um inteiro, que representa o número de linhas limpas;

public int clearSquare(Square square) – Semelhante ao método anterior, embora este recebe um objeto do tipo quadrado, e limpa esse mesmo quadrado passado por parâmetro. Retorna um inteiro, que representa o número de quadrados limpos;

public void addBlock(Block block, String insertedColumn, int insertedRow) – Este método, como o nome indica, adiciona um bloco ao tabuleiro, consoante o que é preenchido como parâmetros (o bloco, a coluna e a linha).

ErrorCode

Esta classe, representa um enumerado, que contem os erros que são possíveis de acontecer, no uso do BlockoDoku.

Contém os seguintes erros:

- **PLAYER_NAME_CANT_BE_NULL** – O nome do jogador não pode ser nulo;
- **CANT_PLACE_BLOCK** – Não é possível colocar o bloco no tabuleiro;
- **FILE_CANT_BE_NULL_OR_EMPTY** – O ficheiro não pode ser nulo ou vazio;
- **CANT_SAVE_GAME** – Não é possível guardar o jogo;
- **CANT_SAVE_PLAYER** – Não é possível guardar o jogador.

Métodos

public String toString() – O único método existente nesta classe, transforma o erro, que se encontra numa linguagem informal para os utilizadores, para uma linguagem mais percetível para os utilizadores. Retorna o erro numa frase percetível do tipo String.

ExceptionManger

Como o nome indica, esta classe serve como gestor de exceções do programa.

Atributos

error – Este é o único atributo que se encontra nesta classe, e representa um código de erro, proveniente da classe anteriormente mencionada (ErrorCode).

Metodos

public ExceptionManager(ErrorCode error) – Este construtor recebe um código de erro e inicia um gestor de exceções (ExceptionManager);

public ErroCode getError() – Este método, funciona como um *"getter"* para o código de erro.

FileHandler

Como a sua nomenclatura indica, esta classe serve unicamente para manusear ficheiros. Contém apenas um método.

Métodos

public static boolean fileExists(String filename, String extension) – Este método serve para verificar se o ficheiro passado como parametro (em conjunto com a sua extensão) já existem. Retorna verdadeiro (true) caso se verifique essa ocorrência de existência, retorna falso (false) em caso contrário.

Game

Esta classe, representa a classe de jogo. Aqui encontram-se muitos dos aspetos principais do desenvolvimento do jogo em si.

Atributos

gameModeBlocks – Este atributo privado representa um ArrayList de blocos que se encontram por ser ainda colocados;

currentBlocks – Este atributo privado representa um ArrayList de blocos para colocar no tabuleiro;

roundBlocks – Este atributo representa os blocos da ronda em que o jogo se encontra.

gameMode – Representa o modo de jogo em que o jogo se encontra.

board – Representa o tabuleiro onde o jogo se desenrola.

score – Como o nome indica, representa a pontuação.

round – Indica a rodada em que o jogador se encontra. É representada como um inteiro privado;

player – Representa o jogador que se encontra a jogar o jogo. Constitui-se de um atributo privado do tipo jogador (Player).

Métodos

public Game(GameMode gameMode, Player player) – Construtor da classe de jogo (game), que permite instanciar um jogo;

public Board getBoard() – Retorna o tabuleiro do jogo. Funciona como um *"getter"* do tabuleiro (Board);

public Player getPlayer() – Retorna o jogador que se encontra a jogar ao jogo. Funciona como um *"getter"* do jogador (Player);

public HasMap<String, Block> getRoundBlocks() – Retorna um Hasmap com os blocos da ronda. Funciona como um *"getter"* para os blocos da ronda (roundBlocks);

public int getRound() – Retorna um inteiro com a rodada que o jogo vai. Funciona como um *"getter"* para as rodadas (round);

public Score getScore() – Retorna a pontuação (Score) que o jogador tem. Funciona como um *"getter"* para a pontuação (score);

public Block player(String input) – Gere cada ronda e retorna o bloco escolhido pelo utilizador (definido no parâmetro deste método);

public boolean SaveGame(GameManager gameManager) – Recebe um gestor de jogo e adiciona o jogo ao gestor de jogos posteriormente guardando-os num ficheiro (games.bin);

public void generateRoundBlocks() – Este método serve unicamente para gerar os blocos da ronda;

public boolean addPointsToScore(Block block) – Verifica se as condições para obter pontuação são cumpridas e se sim, adiciona a pontuação ao "score" do jogo, retornando veradeiro (true);

public String getRoundBlocksToDraw() – Como a sua nomenclatura indica, este método, "desenha" os blocos da ronda. Retorna uma String com o desenho gerado;

public void createAdvancedBlocks(ArrayList<Square> squares) – Serve unicamente, para criar os blocos para o modo avançado;

public void createBasicBlocks(ArrayList<Square> squares) – Serve unicamente para criar blocos para o modo básico;

public String toString() – Retorna uma String com o modo de jogo, os pontos e o jogador.

Hierarquia de classes

Hierarchy For All Packages

Package Hierarchies:
BlockuDoku.Backend, BlockuDoku.Backend.Exceptions, BlockuDoku.Visuals

Class Hierarchy

- java.lang.Object
  - javafx.application.Application
    - BlockuDoku.Visuals.**Screen**
  - BlockuDoku.Backend.**Block** (implements java.io.Serializable)
  - BlockuDoku.Visuals.**BlockButtonUnit**
  - BlockuDoku.Backend.**Board** (implements java.io.Serializable)
  - BlockuDoku.Visuals.**DrawAlert**
  - BlockuDoku.Visuals.**DrawBlocksToPlay**
  - BlockuDoku.Visuals.**DrawBoard**
  - BlockuDoku.Backend.**FileHandler**
    - BlockuDoku.Backend.**GameFileHandler**
    - BlockuDoku.Backend.**PlayerFileHandler**
  - BlockuDoku.Visuals.**FileManager**
  - BlockuDoku.Backend.**Game** (implements java.io.Serializable)
  - BlockuDoku.Visuals.**GameScreen**
  - BlockuDoku.Visuals.**HomeScreen**
  - BlockuDoku.Visuals.**LoginScreen**
  - BlockuDoku.Visuals.**PersonalPointsScreen**
  - BlockuDoku.Backend.**Player** (implements java.io.Serializable)
  - BlockuDoku.Backend.**PlayerManager** (implements java.io.Serializable)
  - BlockuDoku.Backend.**Score** (implements java.io.Serializable)
  - BlockuDoku.Backend.**Square** (implements java.lang.Comparable<T>, java.io.Serializable)
  - java.lang.Throwable (implements java.io.Serializable)
    - java.lang.Exception
      - BlockuDoku.Backend.Exceptions.**LoadException**
      - BlockuDoku.Backend.Exceptions.**PlacingBlockException**
      - BlockuDoku.Backend.Exceptions.**SaveException**
  - BlockuDoku.Visuals.**Top10Screen**

Enum Hierarchy

- java.lang.Object
  - java.lang.Enum<E> (implements java.lang.Comparable<T>, java.io.Serializable)
    - BlockuDoku.Backend.Exceptions.**ErrorCode**
    - BlockuDoku.Backend.**GameMode**

Melhoramentos entre fases

De forma a colmatar alguns defeitos que existiram no desenvolvimento da fase 1, executamos alguns melhoramentos no desenvolvimento desta fase final, que é a fase 2. Estes melhoramentos são:

- Tratamento de exceções melhorado. Este tratamento retrata uma divisão de exceções por diversas classes, que não fora imposto no desenvolvimento da fase 1;
- Abstração de tabuleiro e blocos melhorada. As classes correspondentes ao tabuleiro e blocos, já permitem ser utilizadas para outros jogos que tenham um design semelhante (por exemplo Tetris);
- Correção de posicionamento incorreto de blocos. Na fase 1, era possível colocar blocos que entrariam em colisão com outros já colocados, sem existir qualquer erro. Nesta fase, se existir um posicionamento incorreto de blocos, existe uma exceção que avisa o utilizador desse posicionamento incorreto.

Manual de Utilizador

Página Inicial

Quando o utilizador executa a nossa versão do jogo, deparar-se-á inicialmente com esta página. Esta janela, consiste apenas numa pequena introdução, com um espaço para o utilizador colocar o seu nome. A colocação do nome é obrigatória, e permitirá ao utilizador poder guardar jogos com o nome inserido.

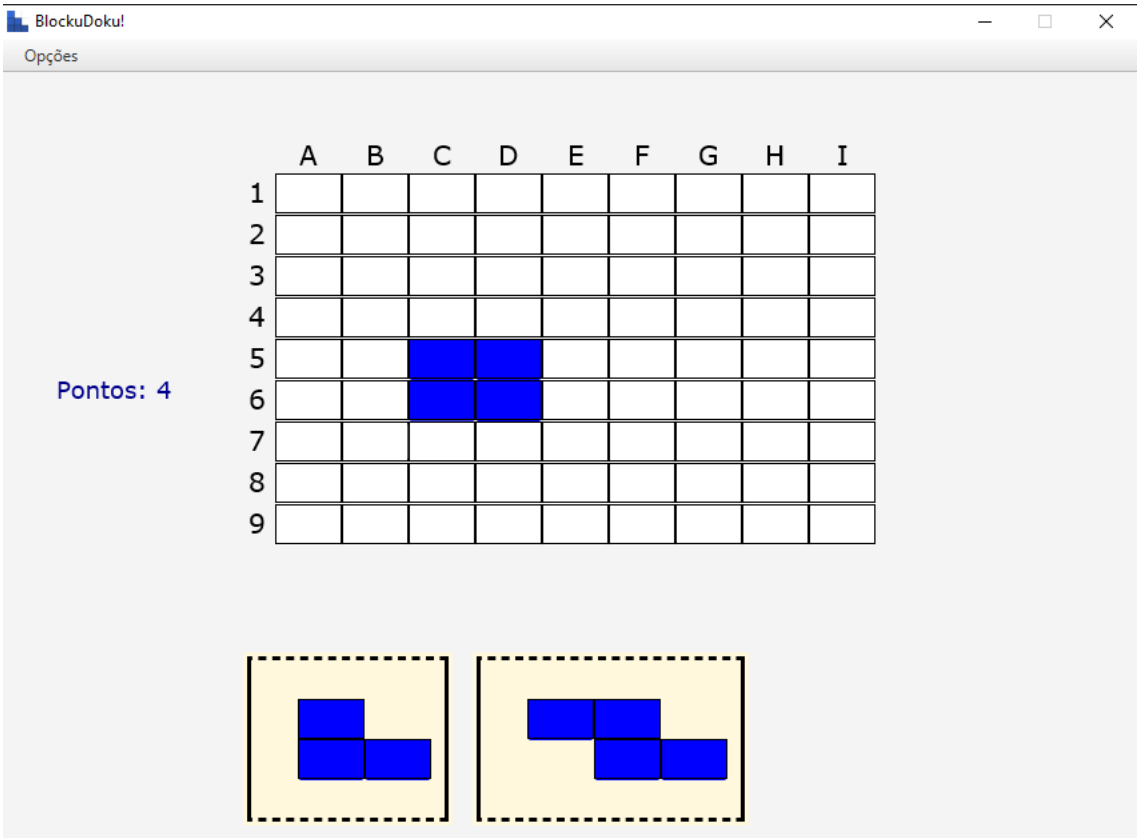
Após correto preenchimento dos campos, basta pressionar o botão “Entrar” e seguirá para a janela seguinte.

Menu Principal

Esta página, como a sua nomenclatura indica, serve para o utilizador navegar entre as janelas principais do nosso jogo. Contém uma mensagem de boas vindas ao utilizador, com o nome previamente inserido. É possível através dos botões, começar um novo jogo, abrir um jogo que fora guardado anteriormente, verificar as suas pontuações, consultar o top 10 e sair do jogo.

Caso o utilizador selecione “Começar novo jogo”, aparecerá uma janela pop-up que perguntará qual o modo de jogo pretendido, o modo básico, ou o modo avançado. Sendo que os blocos disponíveis, para o utilizador selecionar e colocar no tabuleiro, variam de modo para modo.

BlockuDoku!

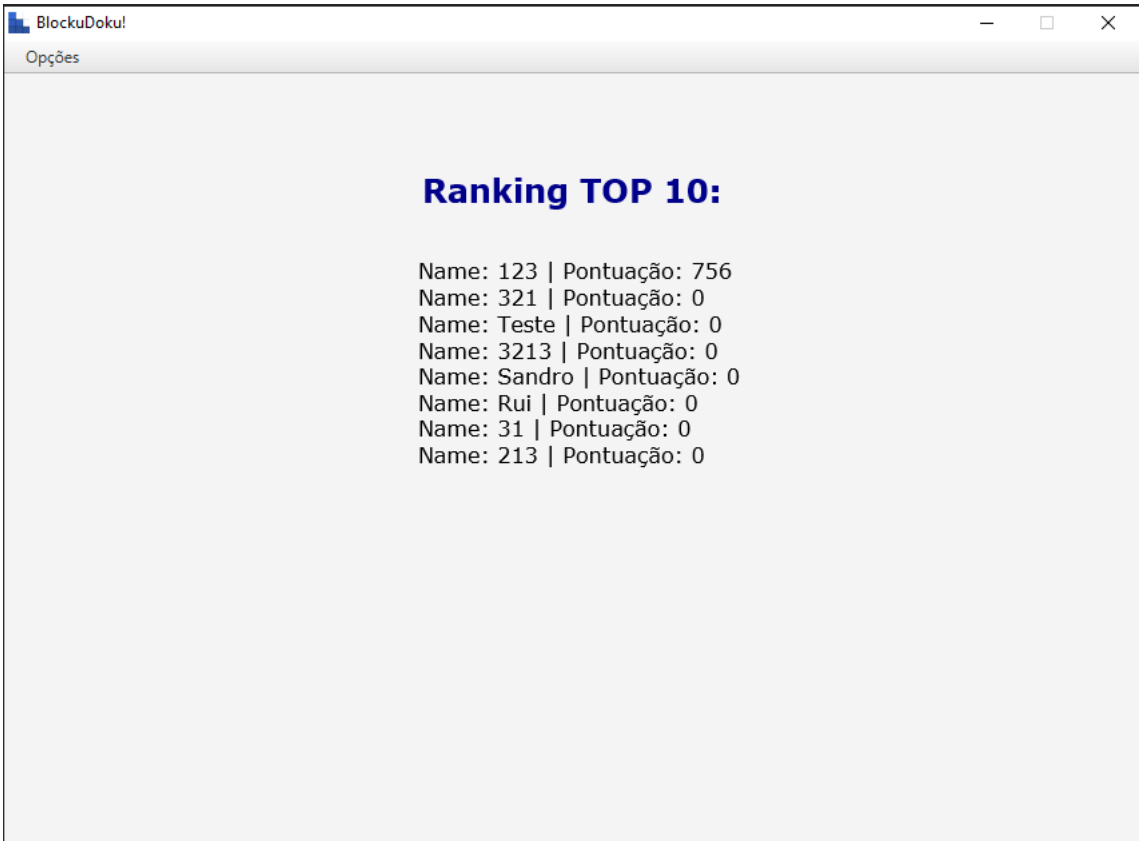


Esta janela contém aquilo que fora o foco principal da execução do projeto, o jogo em si. Nesta secção o utilizador está limitado a apenas jogar (que é o objetivo principal de um jogo), sendo que também pode consultar os pontos enquanto joga. A colocação das peças é simples, e consiste apenas num clique com o rato nas peças em baixo (qualquer uma) e noutro clique no

tabuleiro para a posicionar na posição desejada. Existe também um menu no canto superior esquerdo (menu “Opções”) que permite ao utilizador sair para o menu principal ou guardar o jogo.

Se o utilizador escolher a opção de guardar, aparecerá o explorador de ficheiros do Windows, com a opção para guardar o jogo onde o utilizador quiser.

Ranking (top 10)



Esta janela, possui a demonstração do ranking (mais propriamente, um top 10) dos jogos anteriormente jogados. Esta demonstração constitui-se do nome do utilizador e dos pontos efetuados no jogo por esse jogador. Este ranking encontra-se ordenado numa ordem descendente de pontos. Igualmente ao ecrã de jogo (BlockuDoku!), também possui um menu

no canto superior esquerdo, sendo que este apenas possibilita ao utilizador, poder voltar para o menu principal.

As minhas pontuações

Nesta janela, como a sua nomenclatura indica, é possível verificar as pontuações do jogador (que possui sessão iniciada). Esta lista de pontuações, constitui-se numa demonstração da data e hora do jogo guardado e da pontuação nesse mesmo. Também existe um menu no canto superior esquerdo, demarcado com “Opções”, que permite ao utilizador poder voltar ao menu principal.

Conclusão

Ao longo da realização deste projeto encontramos várias dificuldades, sendo a mais significativa a implementação de Unitests, visto que não tivemos muita formação sobre a criação e implementação destes testes. Visto termos esta dificuldade, não chegamos a implementar Unitests.

Nesta fase do projeto foi criado a parte gráfica do jogo Blockudoku, complementando a assim primeira fase do mesmo. Foram também corrigidos alguns problemas que existiam na primeira

fase do projeto, como por exemplo a divisão de exceções e o posicionamento do bloco. Pode consultar estes melhoramentos no ponto anterior de melhoramento entre fases.

Com a realização deste projeto, consolidamos várias temáticas que foram abordadas durante as aulas teóricas e práticas laboratoriais, podendo ver a sua utilidade e função num caso prático.