

Programação Orientada a Objetos

Licenciatura de Engenharia Informática



Manual Técnico – Projeto Fase 1



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 por consultar as suas pontuações, continuar um jogo gravado, ver o top 10 de pontuações, 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> getColumnInteger(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 “A9”, 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 perceptível para os utilizadores. Retorna o erro numa frase perceptível 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 ErrorCode 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 HashMap<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 verdadeiro (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

Class Hierarchy

- java.lang.**Object**
 - BlockuDoku.**Block** (implements java.io.Serializable)
 - BlockuDoku.**Board** (implements java.io.Serializable)
 - BlockuDoku.**FileHandler**
 - BlockuDoku.**GameFileHandler**
 - BlockuDoku.**PlayerFileHandler**
 - BlockuDoku.**Game** (implements java.io.Serializable)
 - BlockuDoku.**GameManager** (implements java.io.Serializable)
 - BlockuDoku.**Player** (implements java.io.Serializable)
 - BlockuDoku.**PlayerManager** (implements java.io.Serializable)
 - BlockuDoku.**Program**
 - BlockuDoku.**Score** (implements java.io.Serializable)
 - BlockuDoku.**Screen**
 - BlockuDoku.**Square** (implements java.lang.Comparable<T>, java.io.Serializable)
 - java.lang.**Throwable** (implements java.io.Serializable)
 - java.lang.**Exception**
 - BlockuDoku.**ExceptionHandler**

Enum Hierarchy

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