Documentation for Assignment 1 (Team Squash)

<u>Classes written for this Assignment</u>
Frame
Tile
Pool
Player
PlayerTest

Important points to note;
- Make sure the assets folder is present with the charset.conf file within it. This file holds information of which characters are valid tile values.
- The 0 tile is a VALID tile that represents the 'blank' tile.

<u>Frame docs</u>

Frame()

The only available constructor for this class in the current revision. It initialises the ArrayList which backs the Frame, but does not link the Frame to the Pool to be used for the furation of the game. The ArrayList backing the frame is persitent for the duration of the game, and so no functionality is exposed to allow for it to be re-initialised after this has been done in the constructor (however, the design of 'getLetters()' does allow this to be achieved, though this might change in future)

hasLetter(<Tile or char>)/hasLetters(<some collection>)

An overloaded method with an assortment of definitions, all of which share a common functionality. They return a boolean (primitive) value, indicating whether a given letter or series thereof are present in the Frame. It is assumed that the Frame is going to be used in a single-threaded context. These methods are not written to be synchronised, as the underlying implementation of the Frame's data, as an ArrayList<Tile>, is also not synchronised. No exception thrown on failure (as boolean is returned instead).

refill()

A simple method which interacts with the Pool API to refill the Frame. No return value, exception thrown if refill fails.

consoleRender()

A simple method which uses the Arrays.toString() method to print out the current state of the Frame, by calling upon the toString() method of each tile. Printing is sent to stdout.

render()

A placeholder method which will be used later to allow for interaction with javaFX to display the tile in a GUI environment.

removeAll(<some collection>)

An overloaded method which allows a collection of Characters/Tiles (List<Character>, char[], String etc.) to be removed from the Frame. Since the hasLetters() methods are almost all based on collections, there is no signature provided for this method which takes only a single item to be removed as an argument (e.g. remove()). This is done in order to ensure that the Frame's API will force developers to adopt good practice.

isEmpty()

Standard method used to check whether a Frame is empty. Adheres to naming convention of the Collections Framework.

getLetters()

Returns a reference to the ArrayList which this Frame is backed by. The ArrayList returned is not a copy, and so alterations thereto will affect the values stored in the Frame. This behaviour might be changed in a later update to ensure good programming practices are adhered to. (i.e. so that one does not attempt to circumvent the built-in protection mechanisms of the Frame class) getLettersAs<some collection>()

An assortment of methods, all of which return a collection which contains he state of this Frame. Some of the collections returned are copies, whereas others are not, and so they must be used with due care paid to the comments found with their definitions. This behaviour is also under review.

add(<Tile or char>)

A method to allow a single Tile to be added to this Frame. Exception thrown if insertion fails. Alternate version with no exception thrown in event of failure is being looked into for future releases.

addAll(<some collection>)

A method which allows for an array of Tiles to all be added to the Frame. In its current implementation, Tiles from the provided Tile[] array will be added to the Frame until either the array has been consumed, or the Frame is full. An exception is thrown if the Frame reaches capacity before the array is consumed. However, this method exhibits fail-fast behaviour. The elements of the provided Tile[] array which had already been placed into the Frame will not be removed before the exception is thrown. A fail-safe alternative is being considered for a future release.

setPool()

A simple utility method to allow for the value of the ‘pool’ instance variable to be set.

Please note: Usages of the word ‘collection’ above do NOT refer to the Java Collections Framework, unless explicit references thereto are made. Instead the canonical meaning is used.

Tile Docs

Tile(char value)

The only constructor available in the current revision. Initialises a Tile with a value equal to the one passed in as an argument to the constructor. The value is stored as a Character object.

getValidCharactersSet()

Returns a reference to the final class variable VALID_CHARACTER_SET which is a HashSet<Character> containing all valid characters for the current game, as defined in the *assets/charset.conf file. All valid characters must be placed on their own line. A space character is used to represent the blank tile.*

tileArrayFromCharArray()

A static method which takes a char[] array as input, and returns a Tile array, containing references to Tiles whose values match the value of the corresponding char in the input array.

tileArrayListFromCharacterList()

A static method which takes a List<Character> as input, and returns an ArrayList<Tile>, containing references to Tiles whose values match the value of the corresponding Character in the input array. Since the Tile(<char>) constructor is called, changes made to the returned ArrayList will not propogate to the argument itself.

Player Docs

Player()

The default constructor for this class. Creates an instance of Player with 3 variables: username, score and frame. Username is set to an empty string by default. Score is set to 0 (starting position of game), and the frame is created (but not yet filled).

Player(String name)

The overloaded constructor of this class takes a String as argument. The player’s username is set by the name argument passed to the constructor. Score is set to 0 (starting position of game) and the frame is created (but not yet filled).

resetUser()
A simple method that resets the player’s data by setting the string to an empty string and setting the score = 0.

setUsername(String name)
A simple setter method to set the username. Takes a String name as argument and sets it equal to username.

setScore(int value)
A simple setter method to set the score. Takes an integer value as argument as sets it equal to score.

increaseScore(int value)
This method takes an integer value as arugment as increments the player’s score by that value. This method will be used to increment the user’s scores after each turn of the game is played. If a valid word is played on the board, the score will be calculated from the tiles and added to the current player score.

getScore()
A simple getter method to return the value of a player’s score. Takes no arguments.

getFrame()
Takes no arguments. Returns an instance of the frame to allow for access to a player’s tiles.

displayUsername()
A simple method that returns the player’s username as a String. Takes no arguments.

Pool Docs

initPoolTiles()
Inserts all the drawable tiles & their respective quantity into the poolTiles HashMap.

resetPool()
Removes all tiles inside the pool by clearing the poolTiles HashMap containing all the tiles & their respective quantities. After this it then re-initializes the pool, refilling all the respective quantities of the tiles.

displayPool()
Displays the tiles in the pool, their values, and their respective quantities.

getTilePoolCount()
Returns the count of pools in the tile, i.e the summation of the quantities of each tile in the pool.

getTileValue(Character tile)
References the tileValues HashMap which stores the score value of each tile depending on their Character & returns this score value.

getAvailableTiles()
Gets a list of tiles in the pool who’s quantity is greater than 0, so that it can be drawn.

drawRandTiles(int number)
Draws ‘x’ random number of tile(s) specified by the argument ‘number’, and returns an ArrayList containing these tiles.

isEmpty()
Checks if the pool is empty by checking the quantity of each tile in the pool.

PlayerTest Docs

main(String[] args)
PlayerTest contains the main method that drives the tests for each of the above classes.

The main method contains testing that involves creating instances of Pool, Player and Frame.