Software Engineering Group Project
Design Specification AUM group

Author:	dkm4, nah37, jty
Config Ref:	SE_JC_DS_01
Date:	2018-04-26
Version:	1.6
Status:	Release

Department of Computer Science
Aberystwyth University (Mauriitus Branch Campus)
Flic-en-flac
Mauritius

CONTENTS

1. INTRODUCTION

1.1 Purpose of this Document

The design Specification document tracks the necessary information required to effectively define architecture and system design in order to give the development team guidance on architecture of the system to be developed.

1.2 Scope

This document specifies the design requirement for the JoggleCube game. It indicates the main design aspects to be used in the design process of the game. This document should be read by the project manager, project team, and develop team.

1.3 Objectives

The objective of this document is to provide the design specification to the project members, so as the design requirement are met according to the standards of the Design Specification Document.

2. DECOMPOSITION DESCRIPTION

In this section of the document, we will describe the packages that make up the JuggleCube game. The packages have been separated according to their use and role in the application so that the main components dealing with one aspect of the game are in one package. Some of the packages are furthermore divided into sub-packages in order to group the modules accomplishing similar tasks. This is also to allow a clear structure of the code making up the JoggleCube game.

The gameFrames package, as the name suggests, is the package responsible for holding all the modules that make up the interface and also the data that is required for it to function properly. Also, with the help of the other packages, the modules making up the gameFrame package, it will be responsible for *generating* the frames that make up the game and which will be displayed to the user to interact with.

The second main component, the gameIcons package, as its name suggests, is the package that holds the icons referenced in the modules that make up the gameFrames package. This package is mostly to separate the external components that will be loaded apart from the Java code and to hold all of these components in one package for clear structure.

The gameLogic package, the third main component, is the package that holds all the modules that enable the functionality of the game. It interacts with the interface that was made from the gameFrames and gameIcons package. Its main purpose is to provide a structure to build the game on as well as the required functionality to be able to play the game as required. The package is moreover made up of one sub-package, the interfaces package, which holds the interfaces implemented by the modules in the gameLogic package.

2.1 Significant Classes

2.2 GameMainClass module description

Background:

There is quite a bit of functionality, for e.g. enabling the user to check for a word and then adding it to the list of accepted words for that game, clearing a word selection via the assigned button and loading saved games as well as saving played games, that had to work with several components from other modules that a separate module for those was required to keep the project low in coupling. This is also our main module which starts up the game and loads the starting the screen.

Description:

Identification	GameMainClass
Purpose	The purpose of this module is to handle all the remaining interactions taking place throughout the game
Function	This module was made to hold all the other methods and properties that have to deal with the entire game and not only one grid or other sub-component to maintain a low coupling

2.2.1 Grid Module Description

Background:

The Grid module extends the Swing component JPanel as well as makes use of other methods and attributes to create a component. This component is one of the most important components that the user will interact with. It will enable the user to select Tiles to make up a word. It will also responsible for the proper generation and population of the tiles with random letters that make up the grid. However, when a game is loaded, it ensures that the saved tiles are correctly loaded and populated in the grid.

Description:

Identification	Grid
Purpose	The aim of this module is to model the grids that are used to make up the game
Function	This module has been made to ensure accurate and random generation of random letters at the start of a new game as well as proper population of the grids when a previous game is loaded. It also handles user input via keyboard and mouse and deals with the adjacent tiles that have to be highlighted

2.3 Letter Module Description

Background:

A letter is the component that the tile displays on the grid. This has been made so that it is easy to associate the values related to each letter in the population of letters, like the maximum number of occurrences of a letter allowed in a grid and the scrabble score of a letter. It has a toString method that is used when a previously saved game is loaded and only one getter method for the maximum number of a letter allowed in a grid.

Description:

Identification	Letter
Purpose	The aim of this module is to model a component that will be represented in each tile and which will have its own properties
Function	This module was made to provide a clean and easy way to deal with the component that has to be represented in a tile and since a normal Swing component would not be sufficient, the Letter module has been implemented

2.3.1 LetterPopulation Module Description

Background:

This is an enumeration of the letters that make up the letter population. It is a key-value pair where the values stored are the maximum number of occurrences that specific letter can appear in one grid as well as the scrabble value of the letter itself.

Description:

Identification	LetterPopulation
Purpose	The purpose of this module is to provide a collection of all letters that can be used in the game
Function	This module was made to provide an easy way to associate the letters with the properties that every letter must have and which must, in turn, be represented on each tile

2.3.2 PositionInGrid Module Description

Background:

This module has been made since in order to properly populate the grid at the start of every game, be it loading a previously saved game or a new game, the position of the tiles in each grid have to be taken into consideration and doing so with the modules readily available would have caused a lot of strain on the game overall.

Description:

Identification	PositionInGrid
Purpose	The aim of this module is to model the position of a tile in a grid
Function	This module has been made to have an easy way to deal with the positioning of the tiles in the grid

2.3.3 Tile Module Description

Background:

The Tile module models a component that extends the swing button component to make it a better fit for the purpose which it is being used for in the JuggleCube game. The added properties include, but are not limited to, defining whether the tile can be selected by the user, whether the tile is currently in the selection made by the player and providing the position of the tile in the grid that it belongs.

Description:

Identification	Tile
Purpose	The aim of this module is to model a component that will extend the properties and functionalities of a standard button
Function	This module has been made to provide a suitable component to work with which will make up the grids

2.4 Mapping from requirements to classes

In order to ensure properly fulfilling the requirements needed, a table mapping the functional requirements onto the classes which contribute to those requirements is shown below:

Requirement	Classes providing requirement
FR1	gameMainClass, startMenu
FR2	playGame, Grid, Tile, Letter, LetterPopulation
FR3	loadGame, playGame, Grid, Tile, Letter, LetterPopulation
FR4	playGame, gameMainClass, gameTiming
FR5	selectGame
FR6	gameScoreBoard
FR7	playGame, Grid, Tile
FR8	playGame, gameMainClass
FR9	playGame, gameMainClass
FR10	playGame
FR11	playGame

3. DEPENDENCY DESCRIPTION

3.1 Component Diagrams

Since there are no other programs, and the JoggleCube game was written all in Java, the component diagrams shown below are about describing the relationship between the relevant modules in the project. These also show the inheritance dependencies between the modules.

Click on play game:

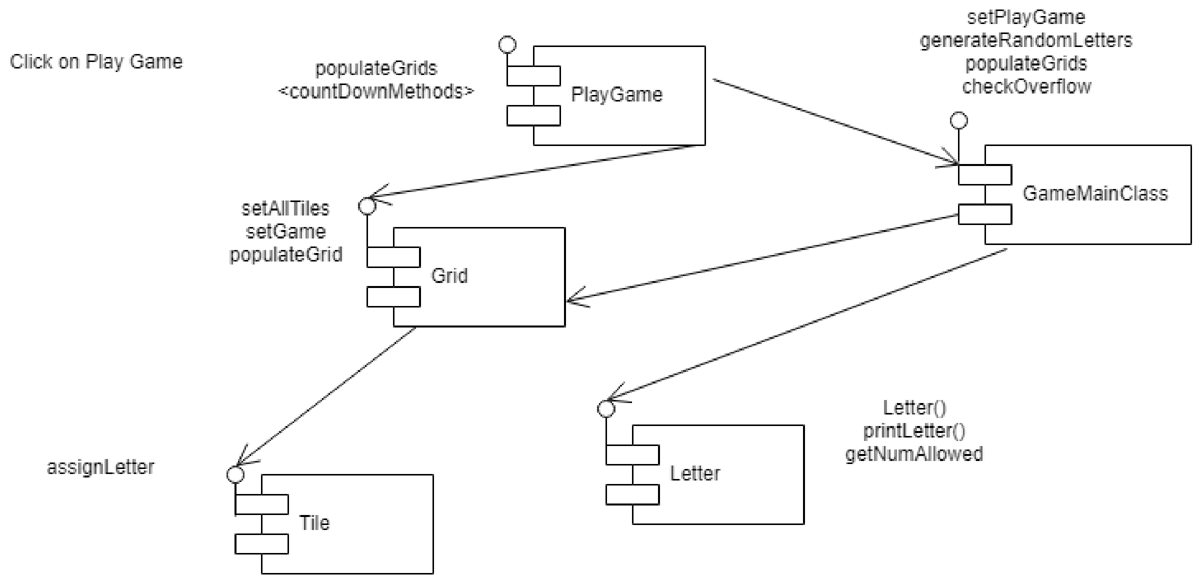

Fig 1 component diagram for playing a game

Click on load game:

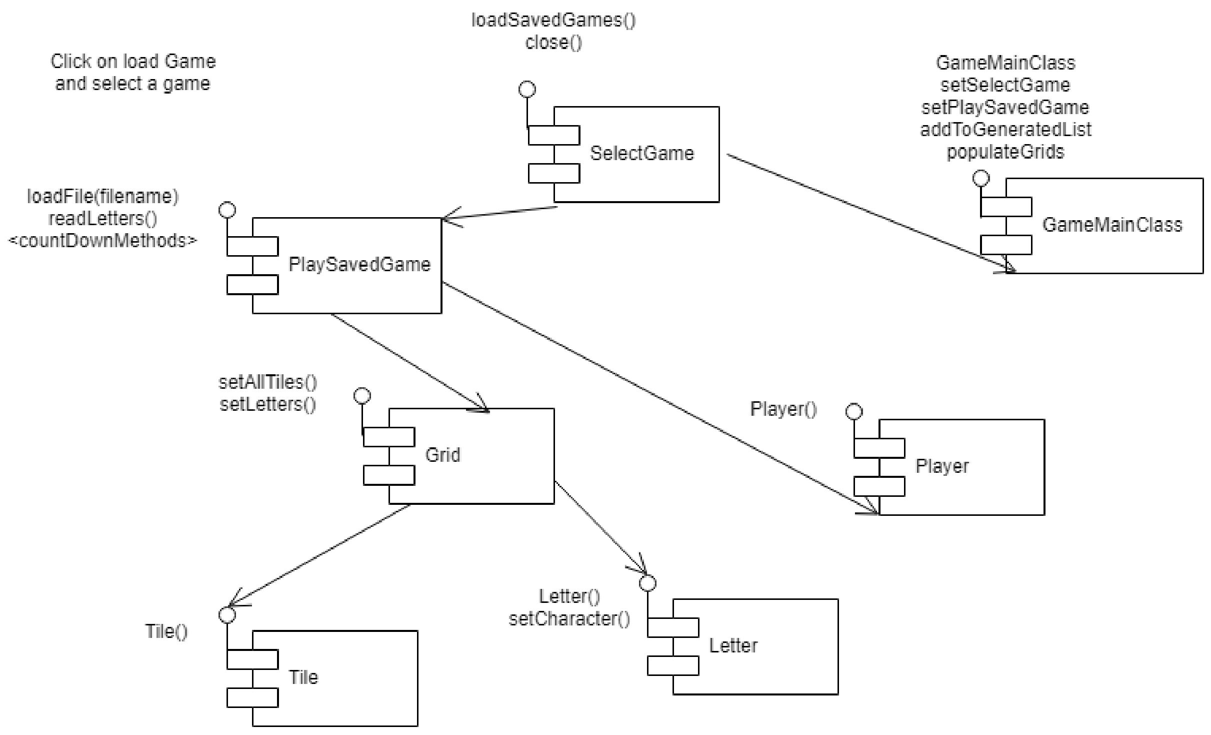

Fig 2 Component diagram for loading a game

View scoreboard:

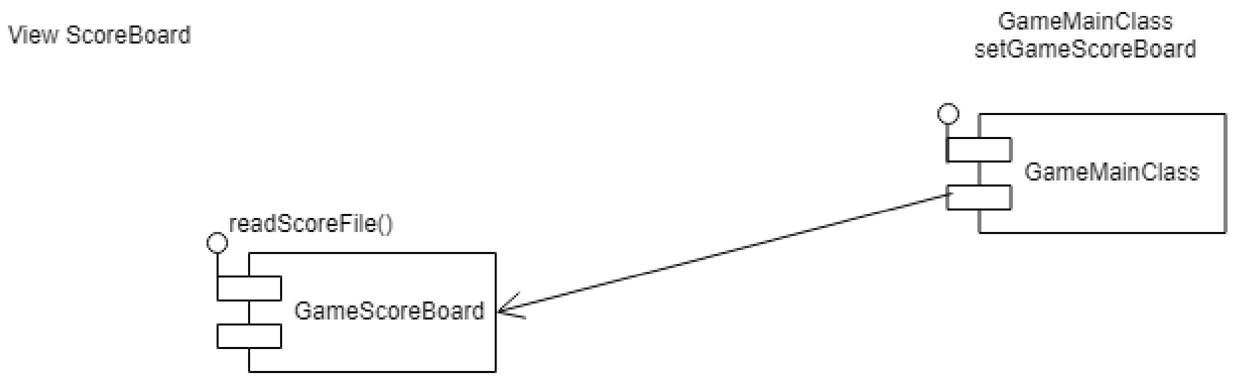

Fig 3 Component diagram for viewing the scoreboard

4. INTERFACE DESCRIPTION

4.1 gameMainClass

public class gameMainClass
public static void main(String[] args)

gameMainClass is the main class of the program. It is a public class which manage all the back-end function in the JoggleCube . The main class is a public static void type since it there is no object to invoke therefore it has a static method to allow invocation from class and it does not return any value. It accepts arguments of type string in a collection. The main class allow the startMenu to be loaded.

public void setStartMenu(StartMenu startMenu)

It is a public class with parameter startMenu that is used so that there is only one instance of the StartMenu frame is use.

public void setPlayGame(PlayGame playGame)

It is a public class with parameter playGame that is used so that there is only one instance of the PlayGame frame that is used and will be manipulated. It also sets the set the starting grid when game start.

public void setPlaySavedGame(PlaySavedGame playSavedGame)

It is a public class with parameter playSavedGame that is used to set game frame and grid of the saved game.

public void setGameHelp(GameHelp gameHelp)

It is a public class with parameter gameHelp that is used to set up the gameHelp form.

public void setScoreMenu(gameFrames.ScoreMenu scoreMenu)

It is a public class with parameter scoreMenu from the gameFrames package that is used to set up the highscore board form.

public void setSelectGame(SelectGame selectGame)

It is a public class with parameter selectGame that is used to set up the selectGame form on the game menu.
public List<Tile> getSelectedTiles()

public void showPlayGame()

It is a public class with no valid output that is used to show the type of game (save or new) being played at that instance.

public void showGameScoreBoard()

It is a public class that show the score board once the function has been clicked on.

public void setPlaySavedGame()

It is a public class with no valid output that set up and initialize the save game.

public void showPlaySavedGame()
It is a public class that is called when the user chooses to load a save game and it is use to set the game grid of the save game and start the countdown.

public void showStartMenu()
It is a public class that is used to set up the start menu making it visible to the user.

public void generateRandomLetters()
It is a public class that is used to generate the letter from the Enum list file to populate the game grid.

private Boolean checkOverflow(LetterPopulation someLetterEnum)
It is a private class with parameter someLetterEnum that is produce a boolean output which check if the letter generated is exceeding the maximum amount allowed by the requirements.

public void populateGrids()
It is a public class that is used to populate the 3 game grid with letters randomly from the list of letter available.

public void addToGeneratedList(List<Letter> listToAdd)
It is a public class with parameter listToAdd of type List<Letter> which is used to add the generated letter to a List for it to saved list.

public void updateCurrentWord(java.awt.event.ActionEvent evt)
It is a public class with parameter evt of type ActionEvent which is used to update the word field where the current word shows when the user clicked on a letter tile.

private void setSelectableIfNotAlreadySelected(PositionInGrid position, Grid ownerGrid)
It is a private class with parameter position and ownerGrid of type PositionInGrid and Grid respectively. It is used to check if a tile is already selected or not.

public void handleTileAction(ActionEvent evt, JLabel warningLabel)
It is a public class with parameter evt and warningLabel1 of type ActionEvent and Jlabel respectively. It is used to light up a tile once it is pressed.

public void clearTileSelection()
It is a public class used to unselect the whole game grid of the already selected tile.

public void clearCurrentWord()
It is a public class used to clear the tile that is unelectable once the user try to click on it.

public void addCurrentWordToList(JList list)
It is a public class used to add a word to the list of correct word for the player

public Boolean isValidWordThenUpdateScore(JLabel scoreLabel)
It is a public class with parameter scoreLabel1 of type Jlabel1 which returns a Boolean output used to check if the word input by the user is valid or not and then update the score according to amount of score it holds.

public void wrongWord(JLabel warningLabel)
It is a public class with parameter warningLabel of type JLabel which is used to check if the word input is wrong then output an error message.

public void wordAlreadyInList(JLabel warningLabel)
It is a public class with parameter warningLabel1 of type JLabel which is used to output an error message if the word input by the user is already in the list of word entered previously.

public Boolean isNotAlreadyInList(JList currentWordsList)
It is a public class with parameter currentWordsList of type JList which returns a Boolean output if the word input by the user is already in the list.

public void printCurrentWord()

It is a public class used to print out the current entered by the user to the list.

public void countdown()
It is a public class used to countdown the amount of time left for the user to play until the end.

public void setSelectableInNeighbourGrids(Tile tileInOriginGrid, List<Tile> selectedTiles, Grid ownerGridOfTile)
It is a public class with parameter tileInOriginGrid, selectedTiles and ownerGridOfTile of type Tile, List<Tile> and Grid respectively. It is used to set up the selectable tile from a neighbour grid.

public List<Grid> getNeighbourGrids(Grid ownerGrid)
It is a public class with parameter ownerGrif of type Grid with an output of type List. It allows the user to click on a neighbouring tile.

public Boolean newOrSavedGameCheck()
It is a public class used to return a boolen output to check is the game is a new game or a saved game.

4.2 startMenu

public class startMenu extends javax.swing.JFrame
The startMenu is a public classs which extends the jFrame class to allow the the jFrame design to run in the program.

public startMenu()
It is a public class which is use to initialize the core components of the jFrame.

public void close()
It is a public class used to control the termination of the game. Once this function has been clicked on the gameplay will ask the user if he/she wants to end and close the game.

private void initComponents()
It is a public class with no return value which is use to initialize the core components of the gameplay design.

public void actionPerformed(java.awt.event.ActionEvent evt)
It is a public class which extends an actionevent evt and returns no values. The actionPerformed is use to monitor the action being performed once pressed or clicked.

private void jbnewGameActionPerformed(java.awt.event.ActionEvent evt)
It is a public class which returns no values and is use to control the action once the jbutton newGame is pressed.

private void jbhelpActionPerformed(java.awt.event.ActionEvent evt)
It is a public class which returns no values and is use to control the action once the jbutton help is pressed.

private void jbSettingsActionPerformed(java.awt.event.ActionEvent evt)
It is a public class which returns no values and is use to control the action once the jbutton settings is pressed.

public static void main(String args[])
The main class is a public static void type since it there is no object to invoke therefore it has a static method to allow invocation from class and it does not return any value. It accepts arguments of type string in a collection. The main class allows the function to be called.

4.3 playGame

public class playGame extends javax.swing.JFrame
The playGame is a public classs which extends the jFrame class to allow the the jFrame design to run in the program.

public playGame()
It is a public class which initialize the game components and allows the creation of a new game grid. It create the object for the playMenu.

private void initComponents()
It is a private class which is use to called the variables and return no values since it is void type.

public int getSize()
It is a public class which is required to return a integer value for the grid panel to compute.

public String getElementAt(int i)
It is a public class with a parameter integer i and it returns a string value from the grid for the user to know what element a single grid contains.

public void actionPerformed(java.awt.event.ActionEvent evt)
It is a public class with parameter evt as an ActionEvent and outputs no value. It performs all the action to be performed once a button is clicked.

private void populateGrids()
It is a private class to be called only in a specific scope of the code and it is use to generate the game grid with letter for the user to play.

private void addWordBtn(java.awt.event.ActionEvent evt)
It is a private class with parameter evt of type ActionEvent which is used to set up the addWord button and it function on the game screen.

private void clearWordBtn(java.awt.event.ActionEvent evt)
It is a private class with parameter evt of type ActionEvent which is used to set up the clearWord button function on the game screen.

private void handleTileAction(java.awt.event.ActionEvent evt)
It is a private class with parameter ActionEvent evt to handle the selection of the user once a game tile has been clicked the other grid will also light up the respective game tile.

private void changeViewBtnActionPerformed()
It is a private method that is used to change the view of the game grid once the button is clicked.

public javax.swing.JTextField getCurrentWordField()
It is a public method which returns an output of type swing.JTextField used to get the placing of the current word field.

public void updateCurrentWordField(String word)
It is a public method with parameter word of type string which is used to update the corrent word field.

public Grid getGrid()
It is a public method used to set up the grids on the game screen.

public int getScore()
It is a public getter method used to getting the score of the game.

public Grid setGrid()

This a public setter method used to set up the game grind in the game.

public int setScore()
This a public setter method used to set up the game score in the game.

public void run()
It is a public class with no values to be output and it is use to link the gameplay button to the main menu of the game.

4.4 playSavedGame

public class PlaySavedGame extends javax.swing.JFrame

It is a public class which extends the JFrame and is used to implements the load game options that prompt the user to choose the saved file to load and play.

public PlaySavedGame()

It is a public class that is use to build the form of the playSavedGame and populate the game grid.

public void setFilename(String filename)

It is a public method with parameter filename of type string used to set the name of the file.

public void loadFile(String filename)

It is a public method with parameter filename of type sting used to load the save file to the game.

private List<Letter> readLetters(Scanner infile)

It is a private class with parameter infile of type Scanner which returns a List<Letter> output used to read the letter that make up the grid of each grid.

public void endSavedGame()

It is a public class which is used to end the saved game and saved the file with all of its details.

public javax.swing.JTextField getCurrentWordField()

It is a getter method used to get the word in the current field of the game.

public void updateCurrentWordField(String word)

It is a public class with parameter word of type string that is used to update the current word field.

public Grid getGrid()

It is a getter method for getting the grid in the game screen.

public int getScore()

It is a getter method used for getting the score of this game playing.

public void setScore(int score)

It is a public class with parameter score of type integer which is used to set the score of this game to the save text file.

public void setCountdown()

It is a public class used to set up the countdown of the game.

4.5 gameScoreBoard

public class gameScoreBoard extends javax.swing.JFrame
It is a public class which extends the jFrame and it is the main class for the game board.

public gameScoreBoard()
It is a public class use to build object needed for the main class.

private void initComponents()
It is a private class for the scoreboard scope only and is used to initialize the variables needed for the gameScoreBoard class.

private void readScoreFile()
It is a public class that load the highscore content from the save text file.

public void run()
It is a public class which return no values and is use to create and displays the game score board for the game

4.6 selectGame

public class selectGame extends javax.swing.JFrame
It is a public class which extends the jFrame and is the main class for the selectGame class.

public selectGame()
It is a public class which initialize the game components and create the object for the class selectGame
public void close()
It is a public class which is used to close the opening window on the game screen.

private void loadSavedGames()
It is a private class that is used only in this scope that is used to load the saved game.

public void actionPerformed(java.awt.event.ActionEvent evt)
It is a public class with parameter evt of type ActionEvent that is generated by the Swing used to monitor the action performed on a button.

public boolean run()
It is a public class of type Boolean which ask the user if he/she really wants to exit the class

4.7 scoreMenu

public class scoreMenu extends javax.swing.JFrame
It is a public class which extends the jFrame and is the main class in scoreMenu.

public scoreMenu()
It is a public class used to initialize and build the object for the scoreMenu class

public void checking()
It is a public class that is used to check the game type of the game and called the appropriate method according to each type.

public void setGameType(boolean isNew)
It is a public class with parameter isNew of type boolean that is used to sets up the game type.

public boolean isNew()
It is a public class that checks whether the game is a new game or a loaded game.

public void setFilename(String filename)
It is a public class with parameter filename of type string that sets the name of the file from which the game was loaded.

public void setPlayers(ArrayList<Player> previousPlayers)
It is a public class with parameter previousPlayers of type ArrayList<Player> that gives the details of all the players who has played the saved game.

public void getCopyPlayedGame(Grid grid1,Grid grid2,Grid grid3,int score)
It is a public class with parameter grid1,grid2,grid3 of type Grind and score of type int that is use to get all the details of the game being played.

private boolean isAmongBest()
It is a private class in this scope which return a Boolean output that states whether the game score of a specific game is among the top ten.

private void updateScoreBoard()
It is private class in this scope that is used to update the game highscore if isAmongBest return true and save the file.

public void saveNewGame(File filename)
It is a public class with parameter filename of type File that is used to the game the user is playing at that time and also write the player name and score in the text file.

public void loadScoreBoard()
It is a public class that is used to load the highscore from the text file to the game UI when a player played a saved game.

public void saveLoadGame()
It is a public class that is used to save the loaded game by populating the grid and writing the player name and score.

private String characterCheck(JTextField textField,JLabel label)
It is a public class with parameter textField and lable of type JTextField and Jlabel Respectively. It is used to check if a textField contain any special character or whether it is empty when the save-file is saved.

public void close()
It is a public class that is used to close the current save game file window.

private void initComponents()
It is a public class with no return value and is use to initialize the components needed for the creation of the scoreMenu class.

4.8 selectGame

public class selectGame extends javax.swing.JFrame
It is a public class which extends the jFrame and is the main class in selectGame.

public selectGame ()
It is a public class used to initialize and build the object for the selectGame class

private void initComponents()
It is a public class with no return value and is use to initialize the components needed for the creation of the selectGame class.

public void run()
It is a public class which output no value and is use to create and display the selectGame form and linked to the gameMenu.

4.9 gameHelp

public class gameHelp extends javax.swing.JFrame
It is a public class which extends the jFrame and is the main class in gameHelp.

public gameHelp()
It is a public class used to initialize and build the object for the gameHelp class

private void initComponents()
It is a public class with no return value and is use to initialize the components needed for the creation of the gameHelpclass.

public void run()
It is a public class which output no value and is use to create and display the gameHelp form and linked to the gameMenu form.

5. DETAILED DESIGN

In order to choose between JavaFX and Swing as graphics and media packages to use for the JoggleCube game, two experimental projects have been undertaken; one in JavaFX and one in Swing. After working on both experimental projects for a little while, it was decided that for the purpose of this project, Swing would be a better choice. However, this also meant that in order to implement the designs that we were working on, we would have to make custom Swing components. This is what made up most of the second experimental project.

5.1 Significant algorithms

Making the custom components so that they are an appropriate fit for the needs of our project meant creating custom Swing components. This was made possible by using the already existing Swing components which had the most resemblance to the components we would require and implementing our own functionality on top of that. And so, the Tile and Grid modules were made to do exactly that, extending the JPanel and JButton components respectively.

In order to deal with the letter population which each had their own properties (maximum number of occurrences allowed in one grid, scrabble score of letter), we tried to implement those through Lists. But after experimenting, we realised that it would quickly become a very tedious task to keep up with that. That is when we found Enumerations to be a perfect fit for that element of the project.

Implementing the required methods to enable the proper handling of the tiles in the grid when the user selects one tile was the task that required the most time to implement since it was working with several components at once.

Eventually, we were able to write a short and maintainable algorithm for that purpose which is also extended on in the project.

5.2 Significant data structures

A class relationship showing the entity relationships between classes is shown below in the gameLogic package is shown below.

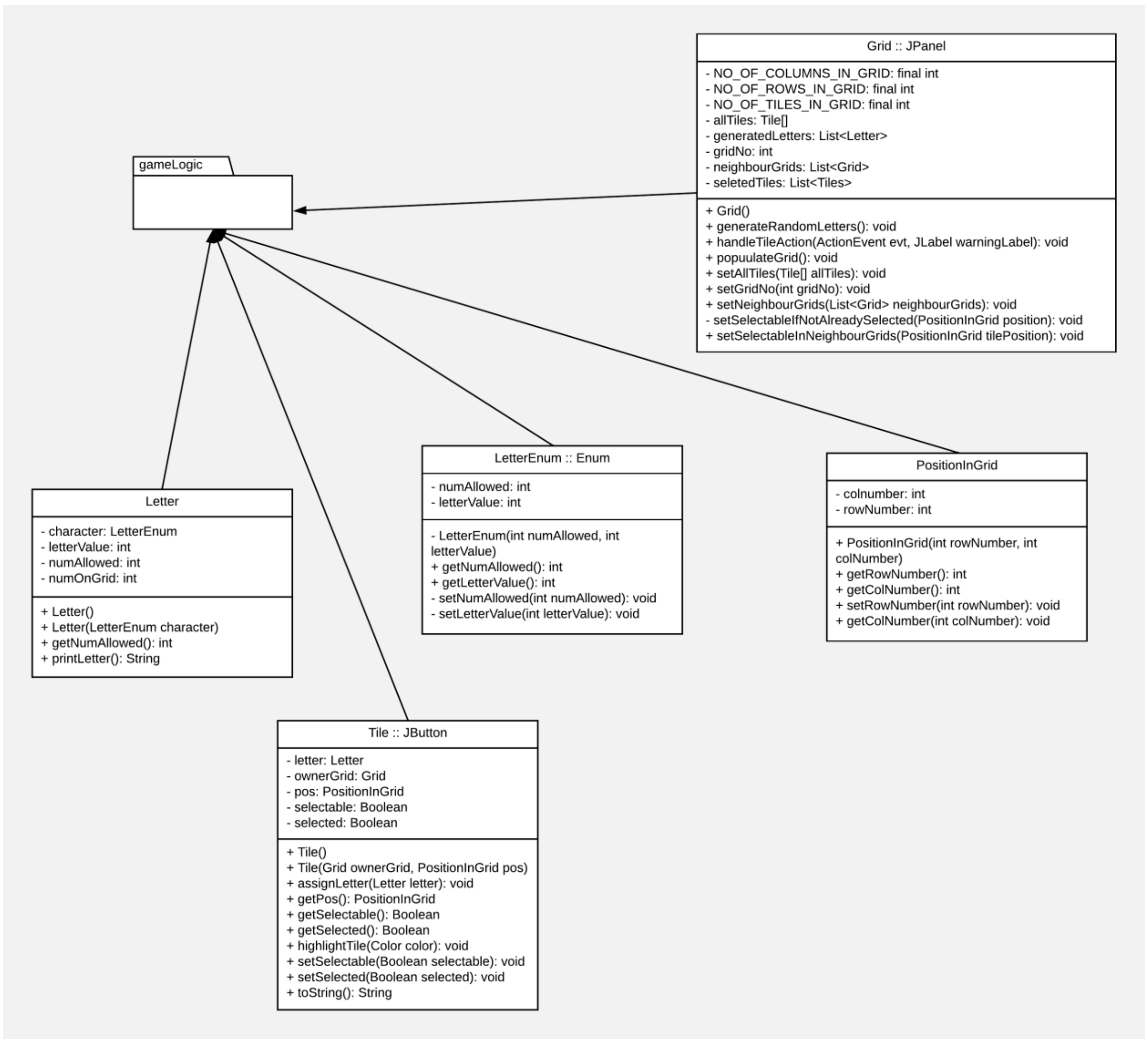
gameLogic
Grid :: JPanel
- NO_OF_COLUMNS_IN_GRID: final int
- NO_OF_ROWS_IN_GRID: final int
- NO_OF_TILES_IN_GRID: final int
- allTiles: Tile[]
- generatedLetters: List<Letter>
- gridNo: int
- neighbourGrids: List<Grid>
- seletedTiles: List<Tiles>
+ Grid()
+ generateRandomLetters(): void
+ handleTileAction(ActionEvent evt, JLabel warningLabel): void
+ popuulateGrid(): void
+ setAllTiles(Tile[] allTiles): void
+ setGridNo(int gridNo): void
+ setNeighbourGrids(List<Grid> neighbourGrids): void
- setSelectableIfNotAlreadySelected(PositionInGrid position): void
+ setSelectableInNeighbourGrids(PositionInGrid tilePosition): void
Letter
- character: LetterEnum
- letterValue: int
- numAllowed: int
- numOnGrid: int
+ Letter()
+ Letter(LetterEnum character)
+ getNumAllowed(): int
+ printLetter(): String
LetterEnum :: Enum
- numAllowed: int
- letterValue: int
- LetterEnum(int numAllowed, int letterValue)
+ getNumAllowed(): int
+ getLetterValue(): int
- setNumAllowed(int numAllowed): void
- setLetterValue(int letterValue): void
PositionInGrid
- colnumber: int
- rowNumber: int
+ PositionInGrid(int rowNumber, int colNumber)
+ getRowNumber(): int
+ getColNumber(): int
+ setRowNumber(int rowNumber): void
+ getColNumber(int colNumber): void
Tile :: JButton
- letter: Letter
- ownerGrid: Grid
- pos: PositionInGrid
- selectable: Boolean
- selected: Boolean
+ Tile()
+ Tile(Grid ownerGrid, PositionInGrid pos)
+ assignLetter(Letter letter): void
+ getPos(): PositionInGrid
+ getSelectable(): Boolean
+ getSelected(): Boolean
+ highlightTile(Color color): void
+ setSelectable(Boolean selectable): void
+ setSelected(Boolean selected): void
+ toString(): String

Fig 4 Class diagram showing the data structures in package gameLogic

The one for the gameFrames package is below.

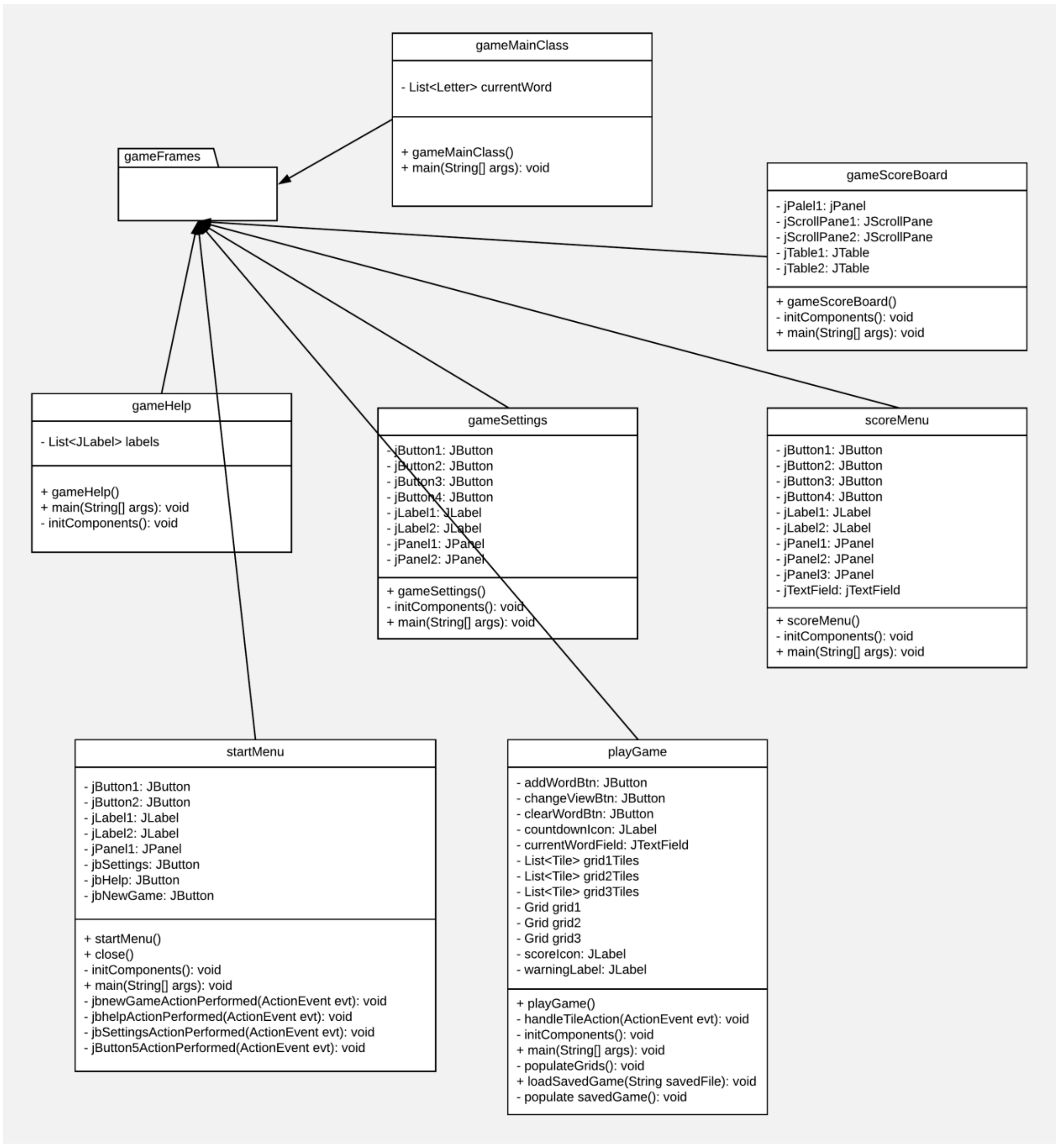

Fig 5 Class diagram showing the data structure in package gameFrames

5.3 Files structure

5.4 High score file

The high score file is a simple text file and it stores the name and the score of the ten best players. The first line is the numbers of players details stored in the file. The next two lines is the name of a player followed by its score. The player's details are sorted in descending order. The example below stores the score and name of the 6 best players.

```
6
Niman
930
Gin527
800
Minajdg
552
Nxgmkl37
550
Mila
400
Ndbg9
205
```

5.5 Saved game file

The saved game file is a simple text file. The letters in each row in the grids are stored on the same line. Each group of three lines corresponds to the letters in a grid, starting from the beginning of the file. It is followed by the number and the details of all the players who have played the game. The player's details are stored in two lines and they are sorted in descending order. The first line is the player's name and the next one is its score. The example below shows the structure of the saved game file.

```
abc
def
ghi
jkl
mno
pqr
stu
vwx
yza
6
Niman
930
Gin527
800
Minajdg
552
Nxgmkl37
550
Mila
400
Ndbg9
205
```

6. REFERENCES

[1] Software Engineering Group Projects: General Documentation Standards. C. J. Price, N. W. Hardy, B.P. Tiddeman. SE.QA.03. 1.8 Release

DOCUMENT HISTORY

Version	*CCF No.*	*Date*	*Changes made to document*	*Changed by*
1.0	N/A	2018-04-19	Main parts	nah37
1.1	N/A	2018-04-26	Interface description	jty
1.2	N/A	2018-04-26	Files structure	nah37
1.3	N/A	2018-04-26	Decomposition description	dkm4
1.4	N/A	2018-04-26	Dependency description	dkm4
1.5	N/A	2018-04-26	Significant algorithms & data structures	dkm4
1.6	N/A	2018-05-11	Updated and verified the document to QA document standard	jty