COMP8702 – Research Coding Assignment

This assignment will be an individual investigation assignment where you will apply all the skills and knowledge that you have gained through the course of this topic, (and to some extent a research component for the tasks we will not cover), to complete a Numbers based puzzle game.

Counts for 10% of the total mark, Due: 5pm, 1 June, 2018

Do not leave this assignment to the last minute. There is significant material that needs to be researched in order to produce an adequate assignment.

Read this document carefully! Any clarification should be sought at the earliest opportunity.

Specification: Number Game

The game itself allows players to find matches between pairs of numbers or two numbers that add up to 10. The objective of the game is to clear the game board and score the highest possible score. The game board (see Figure 1) is made up of a matrix of 12 rows and 9 columns. Each of the cells within the matrix has a random number (1 to 9) allocated.

The player is able to select neighbours within the same row, column, or by wrapping around the edge of the board to the previous or subsequent row, Figure 2 on page 3 indicates legal combinations. **Note:** the player must be able to select the neighbours in any order i.e.

- left-to-right or right-to-left
- top-to-bottom or bottom-to-top
- wrapping around the right edge of the board and moving down a row or wrapping around the left edge of the board an moving up a row

If a match is found then the panels for those numbers are "removed" from play (they are set to blank panels – orange on the image Figure 1). Pairs can then be made through the blank panels providing the player with more points. On the figure to the right you can see that in the eighth row there is a 5 and then four blank panels and another 5. These two 5s form a pair as there are no number cells between them.

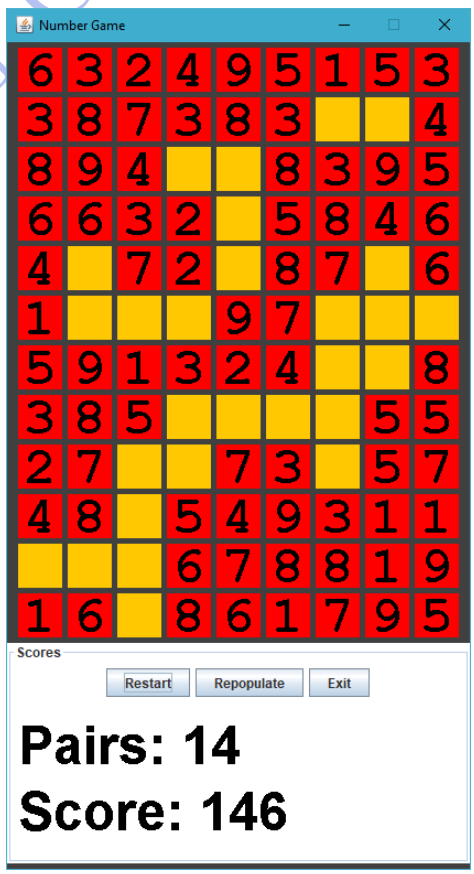

Figure 1: Screen shot of the Number Game

If the player clears an entire row then this blank row is removed from the game board and the subsequent rows are moved one row up the board. This will result in a blank row at the bottom of the game board.

Scoring

The game will keep track of the players score by identifying the total number of pairs as well as the associated score for the matches. When a player matches a pair of numbers (or two numbers that add up to 10) they are awarded 5 points for each panel – therefore a match is equivalent to 10 points.

If the match is formed across blank panels then the player gains an additional 2 points for every blank panel between the matches, i.e. for the example above with the two 5s in the eighth row we would end up with the following calculation: $5 + 2 + 2 + 2 + 2 + 5 = 18$ points.

If a row is cleared then the player is awarded 25 points.

Other Features

The player should be able to repopulate the game board when they choose, however *a penalty of 20 points should be deducted from the score.* Repopulating the game board will result in a random number of blank spaces being reset to a random number. A player may choose to do this if they feel they are stuck and cannot complete any more matches.

The player should also be able to restart the game, clearing all scores and resetting all numbers on the panels. When the player selects restart they should be asked if they wish to save their score first. This will then write their score to a file called `HighScores.hs`, the entry should record:

- the current time (hh:mm) and date (dd/mm/yyyy),
- the player's initials,
- the number of pairs, and
- the total score

There should be an option within the game to view the current high score list – this may be when the game first starts, when a new game is selected, or as an additional window that is always displayed while the game is running. This display of high scores should be an ordered list of the highest to lowest scores. It should only record the last 10 highest scores. If two or more scores are equal then it should be displayed in date order with the oldest scores appearing highest in the list.

The player will also have the opportunity to exit the game. The player should be given the same opportunity to save their score when they quit. A suitable mechanism for this capture of high scores may be to use dialog boxes.

To aid the usability of the game the following key functionality that must be included:

- When the user selects a number panel it should be recoloured green.
- When the user hovers their mouse over an **active number panel** that panel must be coloured white.
- When the user moves the pointer off a number panel it should be returned to its previous colour – i.e. red if it hasn't been selected, green if it has been selected.
- Only allow a maximum of two panels to be selected at one time – if a match does not occur then both panels are de-selected (and appropriately recoloured).

Figure 2 is an example of part of a game board showing legal matches (coloured green).

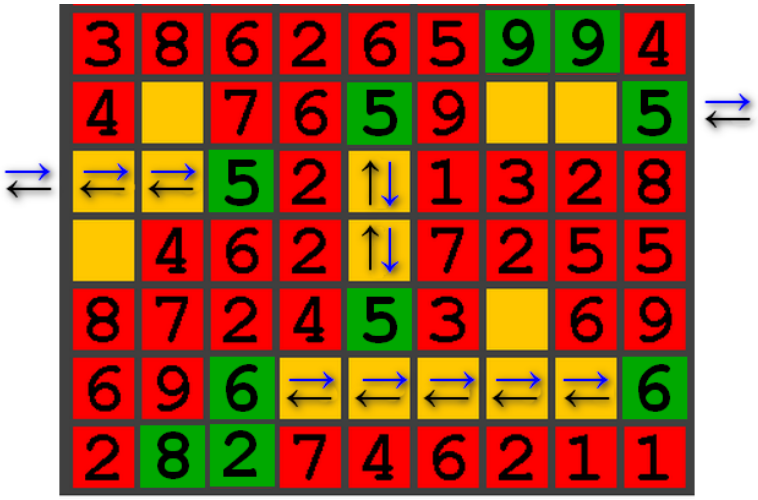

Figure 2: Example of legal matches on a game board

Deliverables

Much of the GUI content for this assignment will not be discussed in lectures, however there is an enormous volume of information detailing GUI development on the Internet. You should prepare for this assignment by designing the logic for this assignment as pseudo code before you begin looking at the GUI development.

This application will assess your knowledge of the Java programming language by testing your ability to use:

Expressions Decisions Looping Method design and invocation Parameter passing Class definition	Covered in lectures
File writing GUI design and implementation Event based programming Mouse interactions Button interactions	Research component

Remember to analyse the problem as necessary and design a solution that suits the deliverables. Treat this as a professional activity, so in-line comments, documentation and application output should be presented appropriately for the target audience. You need to determine the structures you will use. This is an application that will reuse elements significantly and so your development should consider this.

Your code should contain appropriate in-line comments to assist future developers. Include a README.TXT file with relevant technical details (author, development date, version, project objective, method definitions). You are also asked to provide a document that details the class structure and the relationships between your classes. This document should be written so that a future programmer is able to interpret the decisions made and implement changes consistent with the code developed.

You will be awarded marks for the level of completion of your program. This assignment will be marked out of a total of 150 marks as follows:

Display the matrix of NumberPanels – 5 marks

Randomly allocate and display numbers on the NumberPanels – 5 marks

Mouse hover interaction – 10 marks

Mouse selection interaction – 10 marks

Calculation of scores for neighbours and removal of NumberPanels from game – 10 marks

Calculation of scores involving blanks – 10 marks

Calculation of scores with row wrap around functionality – 10 marks

Display of scores within the ScorePanel – 5 marks

Removal of blank rows – 10 marks

HighScore file stored, sorted, retrieved and displayed – 10 marks

Restart and Exit button functionality – 5 marks

Repopulate button functionality – 10 marks

Appropriate documentation for class design – 10 marks

Overall class design and implementation – 20 marks

Style (coding conventions and whitespace) – 20 marks

While the above marking system provides a general structure, the design of your code has to adhere to the specification, therefore there are components that may be necessary to implement, aside from the above assessable elements, that are not given any marks. ***This means design your code for the specification not the marking structure.***

Development should be consistent with the *jGrasp* environment, any issues running a program developed in another IDE will result in a penalty of 50% of your grade.

Submission

The work you submit should adhere to the guidelines presented in this document. If your code does not compile you will be awarded 0 marks. Code that does not compile will not be investigated further. You will not be given an opportunity to resubmit working code. This is a programming assignment and the responsibility for submitting a working product is yours. Development should be consistent with the *jGrasp* environment, any issues running a program developed in another IDE will result in a penalty of 50% of your grade.

The project directory should be zipped and submitted through the MIT Assignment link on FLO. The due date is **5pm, 1st June, 2018**

Plagiarism

The assignment is an individual piece of work. Any students who submit similar submissions will be suspected of academic dishonesty and subject to the applicable policy. This also applies to assignments with significant amounts taken from outside sources with no source reference. Each student should carefully ensure that the assignment is in their own words and any material directly sourced from the work of others, is properly acknowledged. Paraphrasing does not remove the obligation to cite the source. The citing of sources gives the report/argument credibility and establishes that scholarly work has been carried out.