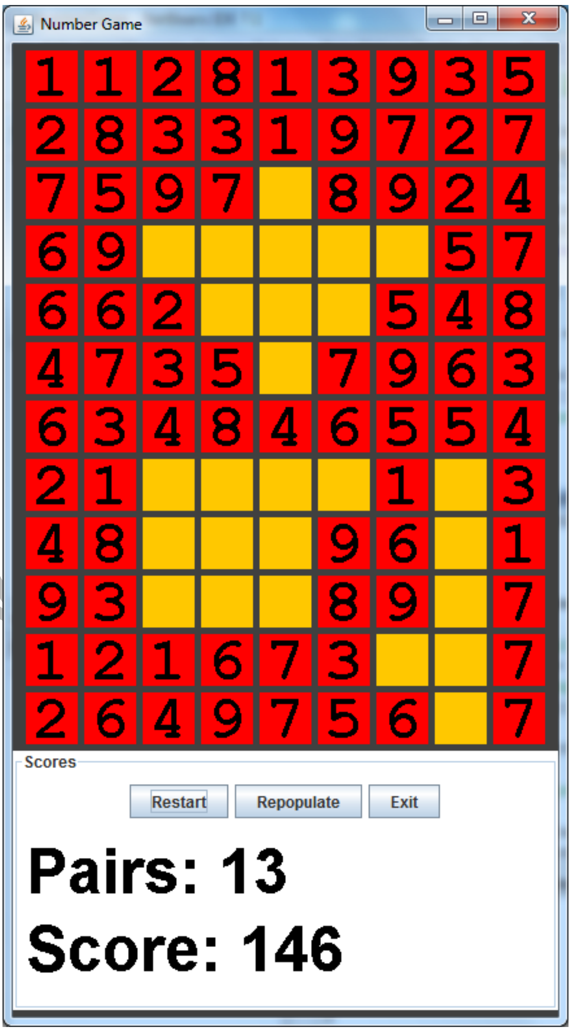
Assignment 2

Read this document **carefully**! Any clarification should be sought at the earliest opportunity. Information about the assignment and assistance, where necessary, will be presented during lectures and tutorials.

Numbers Game



Pairs: 13

Score: 146

Specification

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 to complete a numbers based puzzle 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 to the right) 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 right edge of the board to the subsequent row, figure below indicates legal combinations. 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 to the right). 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 1 and then five blank panels and another 1. These two 1's form a pair as there are no number cells between them.

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.

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, ie for the example above with the two 1's in the eighth row we would end up with the following calculation: 5 + 2 + 2 + 2 + 2 + 2 + 5 = 20 points. If a row is cleared then the player is awarded 25 points.

The player should be able to repopulate the game board when they choose. 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.

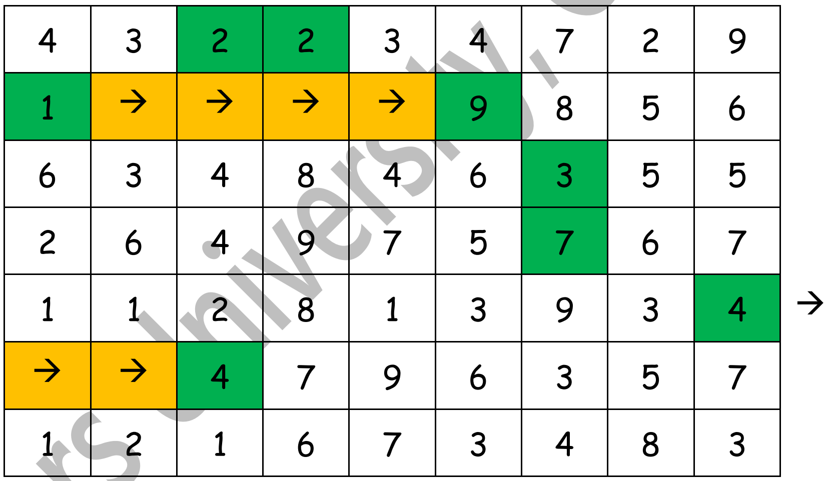
This will then write their score to a file called HighScores.hs, the entry should record the current time and day, the player's initials, the number of pairs, and the total score. There should be an option within the game to view the current high score list – this may be when the game first starts, when a new game is selected, 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 yellow.
- When the user moves the pointer off of a number panel it should be returned to its previous colour – ie. 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).

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



4	3	2	2	3	4	7	2	9	
1	→	→	→	→	9	8	5	6	
6	3	4	8	4	6	3	5	5	
2	6	4	9	7	5	7	6	7	
1	1	2	8	1	3	9	3	4	→
→	→	4	7	9	6	3	5	7	
1	2	1	6	7	3	4	8	3	

Deliverables

Much of the GUI content for this assignment will not be discussed in lectures until approximately week 10. 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,
- File writing,

- Class definition,
- GUI design and implementation,
- Event based programming,
- Mouse interactions, and
- Button interactions.

You will have an opportunity to work on this assignment during practicals throughout the semester. You may ask the demonstrator for some general direction but they will not be assisting in the development of this program. You will receive more information about the assignment as needed throughout lectures and practicals.

Remember to analyse the problem as necessary and design a solution that suits the deliverables. Treat this as a professional activity, so inline 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 inline 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 decision made and implement changes consistent with the code developed. Further discussion of this documentation will take place throughout the semester.

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 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 NetBeans 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 NetBeans environment, any issues running a program developed in another IDE will result in a penalty of 50% of your grade.

Requests for extension need to be presented as early as possible.

The project directory should be zipped and submitted through the Assignment 1 link on FLO.

The due date for Assignment 2 is 5:00pm 20 June 2014.