COMP07027 Introduction to Programming

Programming Project – 2014/15 (Open Knight's Tour)

Banner ID Name **Yu-Ching Ho**

Point	Description	Comments	Score
1	The program follows the project specification in providing a description of the problem, the user instructions, and the initial state of the board. (Knight is initially on a1 in a 3x4 board).	On the whole this was fine, though it would have been good to explain to the user how to interpret the board display. (That is, that a number in a square represents a move number and that the highest number displayed is where the Knight is currently located). You allow the user to choose the starting square, which is nice touch but it would have been good to warn them that if they choose a square on the "b" or "c" files as the starting square that there is no possible tour of the board!	8/10
2	The program defines a suitable set of classes and variables to represent the program state. This includes representing the board and where the Knight has visited and where the Knight is currently located.	It would have been sufficient to use a 2D array with 3 rows and 4 columns to represent the squares – you make the array 4x5 to also include the labels for the ranks and files. This simplifies the output (though, given that you represent an empty square by a hyphen, you could have simplified it even more – see point 4) but means that the array is performing two different roles – one, involving the whole array, to explicitly store what is displayed and one, involving just part of the array, to track the program state. To your credit, however, this arrangement does have the advantage that the array row index numbers exactly match the rank numbers entered by the user, and that the file "a" maps to a column index of 1 rather than 0. The program relies on the String used to store the last user input to separately keep track of and display the Knight's current position.	8/10

		As noted under point 4 this does not work correctly when the user enters an invalid move. The program declares a local variable to count the moves to allow the program to determine when the tour has been completed.	

3	For each move, the program gets the user input and checks that, if the input is a move, that the move is a legal Knight's move and that it does not revisit a square that has already been visited.
	
	If the input is a request to quit the program displays a suitable message and exits.
		The program checks that a square has not been previously visited with the condition:
		
		`If (mapBoard[secondNumber][toNumber] == "-")`
		
		This works, but only because of an optimisation the JVM does to reduce the number of String objects in memory.
		
		(It maintains a pool of String objects which it reuses when the same String literal is used again in the program text).
		
		You should not use the == operator to compare Strings. You should have done what you did when checking if the user input is a "Q" and used the String.equals() method to do this comparison:
		
		`if (mapBoard[secondNumber][toNumber].equals("-"))`
		
		The program does correctly handle a request to quit. You did not get as far as coming up with a way for the program to determine if the move entered is a valid Knight's move, but it was good that you checked that the input square is valid (i.e. is on the board).
			6/10

4	For each valid move, the program updates the program state (i.e. moves the Knight) and displays the new program state.	This works fine except for the point noted in point 3 above that the program cannot determine if the move is a valid Knight's move and will update the program state even if it is not.	8/10

Given that the 2D String array used to represent the board represents an empty square with a hyphen and this is what is displayed if a square is empty, there was no need to check, in your drawMap() method, whether a square contains a hyphen – you can just display it. That is, instead of writing:

```
If (mapBoard[a][b] == "-") {
    TextIO.put(" -");
}
else {
    TextIO.put(" " + mapBoard[a][b]);
}
```

you should just have written:

```
TextIO.put(" " + mapBoard[a][b]);
```

(Again, you should not use == to compare the Strings but use the equals() method of the String class).

As pointed out in the project specification, TextIO does support outputting values in a fixed field width.

In your code, the on-screen columns do not line up for the 10th, 11th and 12th moves because these numbers contain two digits, while the hyphen and other moves are just one character long. It would have given a nicer output to take advantage of this feature and written, say:

```
TextIO.put(mapBoard[a][b], 4);
```

This would also have avoided the need to output a tab character.

(The tab does seem to line up the first digit of each move number when saving to file, but not when displaying on the screen, presumably because the default number of positions the tab moves the output is different in the two cases – it is neater, however, to have the second digit of the two-digit move numbers line up with the single-digit ones, rather than the first digit).

5	After each move, the program checks if the Knight has completed the tour and if so it stops the sequence and displays a congratulatory message.	This was fine.	10/10
6	Additional functionality (see handout for suggested possibilities).	The program allows the user to choose the starting square, and it saves the final position to file at the end.	8/25
7	Program structure. The functionality of the program is distributed across a number of methods with appropriate parameters and return types, and/or across a number of classes each of which encapsulate some aspect of the problem or solution.	The program defines two static methods in addition to main() – one to change the program output to a file, and one to draw the board. It would have been good, given that the former of methods these is named saveFile() to have this method not just switch the output but also actually save the moves by calling the drawMap() method, rather than having main() do this. It would have been good practice to restore the output to the standard output after doing this, though it is true that the program does not output anything further after saving the final position. It was good that variables you used to track the game state are all declared locally in main() – other than the 2D array used to represent the board. This could have been declared in main() as well and passed as a parameter to the two methods. It would have been possible to define a number of other static methods to take some of the work off the main() method – for example (from those functions that your program does implement) to check if a square was on the board, and to check if the user input was in the right format.	7/15

8	Report. The submission includes an individual report on the work done, how the program was developed and tested, and any areas where the program could be improved or further developed given more time and effort (**required**).	You submitted an interesting report indicating how you developed the program from its initial pseudocode description, and how you dealt with errors you encountered along the way. You indicated what you didn't manage to complete (checking for valid Knight moves), and presumably this is what you would have concentrated on if you had more time to work on the project. It was good that the program contains a number of comments, which helps reduce the amount of space in the report used to describe the program's structure and function.	7/10
Total	Total: 62/100		62%