CS360 Assignment 1
Knights Tour

Date Assigned: Thursday, February 5, 2004
Date Due: Tuesday, February 17, 2004
Total Points: 50

Goals:

The goal of this assignment is to get you up and running with the Java programming language. The problem itself is not too difficult but completing this problem in Java will familiarize you with the Java syntax.

The purpose of this assignment is to develop two Java programs to solve the Knight's Tour Challenge. You are free to choose either developing an application or an applet.

Part 1: One of the more interesting puzzles for chess buffs is the Knight's Tour problem, originally proposed by the mathematician Euler. The question is this: Can the chess piece called the knight move around an empty chessboard and touch each of the 64 squares once and only once?

The knight makes L-shaped moves (over two in one direction and then over one in a perpendicular direction). Thus, from a square in the middle of an empty chessboard, the knight can make eight different moves (numbered 0 through 7) as shown below.

	0	1	2	3	4	5	6	7
0								
1				2		1		
2			3				0	
3					K			
4			4				7	
5				5		6		
6								
7								

The board is represented by an 8-by-8 two-dimensional array, let's call it `board`. Each of the squares is initially zero. Each of the eight possible moves can be described in terms of both their horizontal and vertical components. For example, a move of type 0, as shown in the figure above, consists of moving two squares horizontally to the right and one square vertically upward. Move 2 consists of moving one square horizontally to the left and two squares vertically upward. Horizontal moves to the left and vertical moves upward are indicated with negative numbers. The eight moves may be described by two one-dimensional arrays, let's call them horizontal and vertical, as follows:

```
horizontal[0] = 2     vertical[0] = -1
horizontal[1] = 1     vertical[1] = -2
horizontal[2] = -1    vertical[2] = -2
horizontal[3] = -2    vertical[3] = -1
horizontal[4] = -2    vertical[4] = 1
horizontal[5] = -1    vertical[5] = 2
horizontal[6] = 1     vertical[6] = 2
horizontal[7] = 2     vertical[7] = 1
```

Now, let the variables `currentRow` and `currentCol` indicate the row and column of the knight's current position. To make a move of type `moveNumber`, where `moveNumber` is between 0 and 7, use the statements:

```
currentRow += vertical[moveNumber];
currentCol += horizontal[moveNumber];
```

You need to keep a counter that varies from 1 to 64. Record the latest count in each square that knight moves to. Remember to test each potential move to see if the knight has already visited that square, and of course, test every potential move to make sure that the knight doesn't land off the chessboard.

Your job is to write a program (name it Knight1.java) to move the knight around the chessboard. Your program should print:

- A board showing the order in which the squares were visited (similar to below)
- The total number of successful moves

	0	1	2	3	4	5	6	7
0	1	22		20				
1			2			19	12	
2	23		21		13			
3		3	14		18		8	11
4		24			9			
5		15	4	17			10	7
6	**25**				5			
7			16				6	

Part 2: One way to improve success is to use a heuristic (strategy) for moving the knight. Heuristics do not guarantee success, but a carefully developed heuristic greatly improves the chance of success. You may have observed that the outer squares are more troublesome than the squares nearer the center of the board. In fact, the most

troublesome, or inaccessible, squares are the four corners. Intuition may suggest that you should attempt to move the knight to the most troublesome squares first and leave open those that are easiest to get to, so when the board gets congested near the end of the tour, there will be a greater chance of success.

An accessibility heuristic can be developed by classifying each of the squares according to how accessible they are and then always moving the knight to the square (with the knight's L-shaped moves, of course) that is most inaccessible. The figure below shows a two-dimensional accessibility matrix with numbers indicating from how many squares each particular square is accessible. On a blank chessboard, each center square is rated as 8, each corner square is rated as 2 and the other squares have the accessibility numbers of 3, 4 or 6.

2	3	4	4	4	4	3	2
3	4	6	6	6	6	4	3
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
4	6	8	8	8	8	6	4
3	4	6	6	6	6	4	3
2	3	4	4	4	4	3	2

Your job is to write a version of the Knight's Tour program (name it Knight2.java) using the accessibility heuristics. At any time, the knight should move to the square with the lowest accessibility number. In case of a tie, the knight may move to any of the tied squares. Therefore, the tour may begin in any of the four corners. As the knight moves around the chessboard, your program should reduce the accessibility numbers as more and more squares become occupied.
Your program should print:

- A board showing the order in which the squares were visited (similar to below)
- The total number of successful moves

Submitting your programs:

Save both of your programs in a folder called (your PUNET ID) and place the folder in the CS360 drop folder by 9.25 AM on the assignment due date. You do not need to submit the .class files. Just submit your .java files.

Notes on grading:

- To achieve full credit for this assignment, you should complete both parts, print all the correct information, use a graphical user interface and follow coding standards.
- If you complete parts 1 and 2 correctly, print the correct information, follow coding standards but do not use a graphical user interface then you will achieve a maximum of 90% of the grade.

- If you complete part 1 correctly, print the correct information, use a graphical user interface and follow coding standards then you will achieve a maximum of 80% of the grade.
- If you complete parts 1 correctly, print the correct information, follow coding standards but do not use a graphical user interface then you will achieve a maximum of 70% of the grade.