Programming Tutorial 3

Susan Eisenbach

28 November 2007

This tutorial is about writing methods that would be needed to play an easy version of the game Soduko. Your code should make use of the following class declaration:

```
class Square{
   /* holds both the value displayed in a square on a soduko board (0
   * means that the square is empty) and whether it is possible
   * to put i in the square where i is the index of possibles, so
   * if (possibles[2]) then 2 could potentially go in this square and
   * if (!possibles[3]) the 3 cannot go in this square
   */
   int val = 0; //0 means empty, non-empty squares can hold any value between 1 and 9
   boolean[] possibles = { true, true, true, true, true, true, true, true, true, true};
}
```

The main method is the following:

```
void main(){
   Square[][] board = new Square [9][9];
   initBoard(board);
   placeNumbers(board);
   printBoard(board);
}
```

1. Write the method headers (the first line of the method) for
   a. initBoard
   b. placeNumbers
   c. printBoard
2. If initBoard were rewritten as a method that returned a board how would you write its method header?
3. How many elements are there in a board? What are the coordinates of the top left hand square? What are the coordinates of the bottom right hand square?
4. It is important to know if a board contains at least one empty space. Write a predicate hasEmptySpace that takes a board as an argument and returns true if and only if there is at least one empty space on the board. Remember empty spaces have the value 0.
5. Write the method printBoard which prints the board on the screen. For this question you only have to print the values of the squares or a space if the square is empty. You do not have to print out lines demarcating the squares or label the axis. The values should be separated by spaces. (You may make the board more elaborate if you wish.)
6. Write a predicate unique which takes an array of booleans as an argument and returns true if and only if exactly one element of the array is true.
7. After a number n is successfully placed on the board all other squares in the row r where it has been placed are no longer available (not possible) for having n placed on them. Write the method body for `void unSetInRow( int n, int r, Square[][] b)` to set possibles.
8. Write a class Coord to contain the coordinates of the Square being looked at, which are just the indices of the array that holds the soduko board.

Answers:

1.
   a. `void initBoard(Square[][] board)`
   b. `void placeNumbers(Square[][] b)`
   c. `void printBoard( Square[][] b)`
2. `Square[][] initBoard( )`

3. 81, (0,0), (8,8)

4.
```
boolean hasEmptySpace(Square[][]b){
   for (int r = 0; r < 9; r++){
      for (int c = 0; c < 9; c++){
         if (b[r][c].val == 0) {return true;}
      }
   }
   return false;
}
```

5.
```
void printBoard( Square[][] b){
   for (int r = 0; r < 9; r++){
      println(" ");
      for (int c = 0; c < 9; c++){
         if (b[r][c].val == 0) {print("   ");}
         else {print(" "+b[r][c].val + " ");}
      }
   }
}
```

6.
```
boolean unique( boolean [] list ){
   int count = 0;
   for (int i = 1; i < 10; i++){
      if  (list[i]) {count++;}
   }
   return count == 1;
}
```

7.
```
void unSetInRow( int n, int r, Square[][] b){
   for( int i = 0; i<b.length; i++){
      b[r][i].possibles[n] = false;
   }
}
```

8.
```
class Coord{
   // contains the coordinates of the Square being looked at
   // these are just the indices of the array that holds the soduko board
   int row = 0;
   int col = 0;
}
```