Front-End:

* 4 main categories in the GUI: main menu, about page, game board, and key pad.
  1. Main menu
     + One xml page and three buttons: “New Game”, “About”, “Exit”
       - Exit will close the window (but the program will run in the background)
       - About brings up a new activity with one Textview explaining the rules of the game (from Wikipedia)
       - New Game brings up and alert dialog to select difficulty level
     + All buttons on the main menu are set with onClickListeners
     + Color, text, and formatting done in xml files
  2. The about page brings up a simple “How to Play” dialog with fun facts about the game.
  3. Game board
     + The alert dialog is where the user can choose the puzzle difficulty, and once chosen it sends an integer value to a getNewSudoku function in the "Game" Java class.
     + This is where the program interfaces with the random value generator and returns a Sudoku game in an 81 value array.
     + The "Gameview" Java class is then called where it draws one square Sudoku board and one "Solve" button, and populates the board with the given values from the random Sudoku generator.
     + The solve button is where the program interfaces with the Sudoku solver, and if pressed it returns the array containing the puzzle with all valid values.  The puzzle is redrawn and the user is prompted.
     + If the solve button is not pressed, A selection rectangle is drawn and redrawn depending on what box the user has clicked, and once clicked it brings up another dialog containing the keypad.
  4. Keypad
     + The keypad is contained in its separate "keyPad" class and xml file, and is called whenever the user touches the valid boxes of the gameboard.
     + The keypad consists of the numbers 1 - 9, and one "hint" button.
     + Once a number is pressed, it sends a corresponding integer value to Game, where it checks to see if the number is valid or not for the given puzzle.  If it is valid, the number gets assigned into the puzzle array and is drawn into the board.  If it is not valid, the user is prompted and no other changes occur.

Back-End:

* The solving algorithm for our Sudoku app is based on Knuth’s Algorithm X which functions as follows:
  1. If the matrix “A” is empty, it is already solved
  2. Select a column “c”
  3. Select a row such that Ar,c = 1
  4. Include the row in a partial solution
  5. When a “1” is found, that row and column should be deleted from the matrix
  6. Repeat until matrix is empty
* For a more in-depth look about how the solver was implemented see comments in code
* The random Sudoku puzzle generator works in three steps:
  1. It takes an already solved Sudoku puzzle and shuffling the positions in a legitimate way.
  2. It chooses random locations for a number of starting tiles based on the difficulty level and removes all other tiles.
  3. It checks to make sure that every row and column has the minimum number of given tiles determined by the difficulty level. If there aren’t enough, it adds as many tiles as necessary and removes and equal number of tiles from the row or column with the most tiles.
* Easy puzzles have 40 to 49 given tiles and the lower bound for a row or column is 4 tiles.
* Medium puzzles have 32 to 39 given tiles and the lower bound for a row or column is 3 tiles.
* Hard puzzles have 27 to 31 given tiles and the lower bound for a row or column is 2 tiles.
* Completely random puzzles have 27 to 50 given tiles and no lower bound for a row or column.
* NOTE: Paper from MIT proved that the minimum number of tiles is 17. See references for elaboration.
* Shuffling uses three different techniques:
  1. Mutual exchange of two digits.
  2. Mutual exchange of two columns/rows in the same column/row of blocks
  3. Mutual exchange of two columns/rows of blocks
* 3 Objects: Board, House, Tile
  1. Board
     + This object generates a Sudoku game using a pre-solved game as a seed that is then shuffled. It can create easy, medium, and hard games, or can generate a game of random difficulty. This method of generation does not need to make use of zones, only rows and columns.
  2. House
     + This object holds a group of 9 Tiles in an array called "members." It also has a type, which can be 'r' for row, 'c' for column, or 'z' for zone.
  3. Tile
     + This object represents a particular tile in on the Sudoku board. It holds a value, and belongs to a row, a column and a zone, and an id number from 0 to 80.
* For a more in depth description of what each part does, see the comments in the code files Board.java, House.java, and Tile.java.

References:

* <http://en.wikipedia.org/wiki/Sudoku>
* <http://en.wikipedia.org/wiki/Algorithm_X>
* “Sudoku Puzzles Generating: From Easy to Evil” which was written by Xiang-Sun Zhang’s Research Group
* <http://www.technologyreview.com/view/426554/mathematicians-solve-minimum-sudoku-problem/>