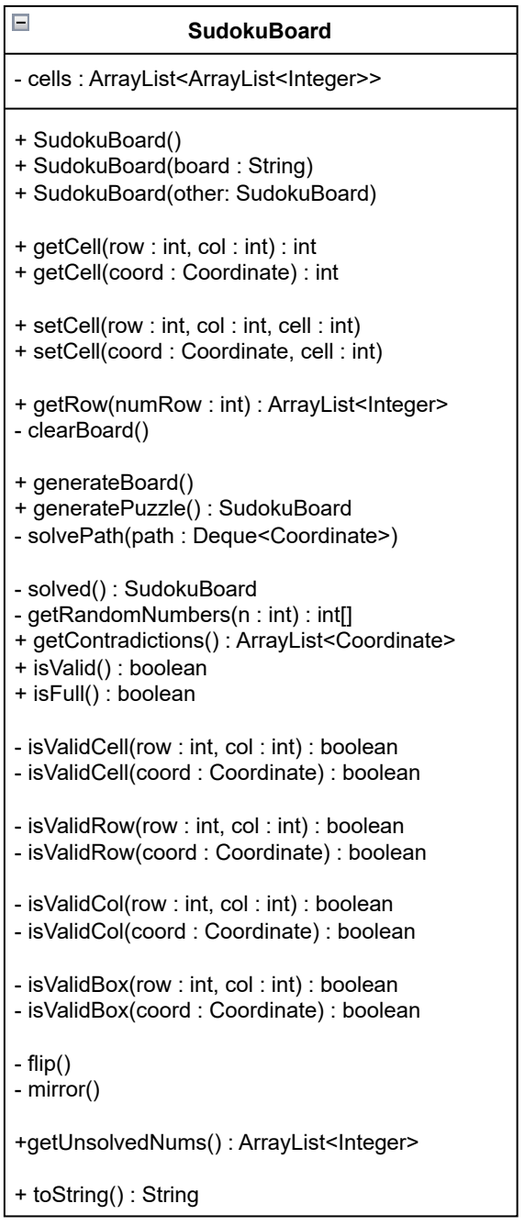
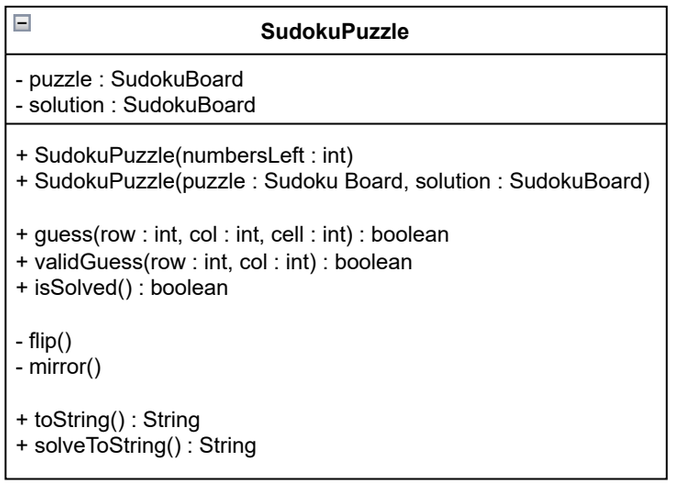
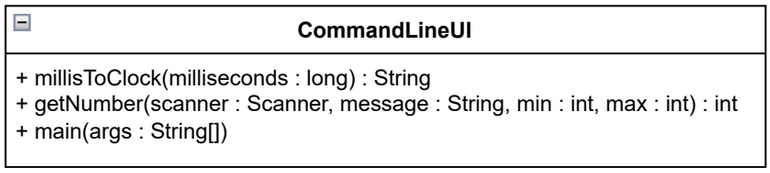
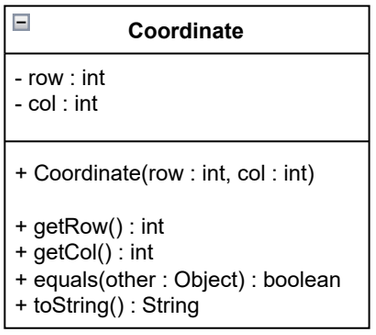

SudokuBoard
- cells : ArrayList<ArrayList<Integer>>
+ SudokuBoard()
+ SudokuBoard(board : String)
+ SudokuBoard(other: SudokuBoard)
+ getCell(row : int, col : int) : int
+ getCell(coord : Coordinate) : int
+ setCell(row : int, col : int, cell : int)
+ setCell(coord : Coordinate, cell : int)
+ getRow(numRow : int) : ArrayList<Integer>
- clearBoard()
+ generateBoard()
+ generatePuzzle() : SudokuBoard
- solvePath(path : Deque<Coordinate>)
- solved() : SudokuBoard
- getRandomNumbers(n : int) : int[]
+ getContradictions() : ArrayList<Coordinate>
+ isValid() : boolean
+ isFull() : boolean
- isValidCell(row : int, col : int) : boolean
- isValidCell(coord : Coordinate) : boolean
- isValidRow(row : int, col : int) : boolean
- isValidRow(coord : Coordinate) : boolean
- isValidCol(row : int, col : int) : boolean
- isValidCol(coord : Coordinate) : boolean
- isValidBox(row : int, col : int) : boolean
- isValidBox(coord : Coordinate) : boolean
- flip()
- mirror()
+getUnsolvedNums() : ArrayList<Integer>
+ toString() : String


SudokuPuzzle
- puzzle : SudokuBoard
- solution : SudokuBoard
+ SudokuPuzzle(numbersLeft : int)
+ SudokuPuzzle(puzzle : Sudoku Board, solution : SudokuBoard)
+ guess(row : int, col : int, cell : int) : boolean
+ validGuess(row : int, col : int) : boolean
+ isSolved() : boolean
- flip()
- mirror()
+ toString() : String
+ solveToString() : String


CommandLineUI
+ millisToClock(milliseconds : long) : String
+ getNumber(scanner : Scanner, message : String, min : int, max : int) : int
+ main(args : String[])


Coordinate
- row : int
- col : int
+ Coordinate(row : int, col : int)
+ getRow() : int
+ getCol() : int
+ equals(other : Object) : boolean
+ toString() : String

Kinds of Collections:

1. List – ArrayList as a method member in SudokuBoard line 16
2. Queue – ArrayDeque in SudokuBoard.generateBoard() line 189

Design Changes:

1. SudokuBoard.solvePath() - We added this method to reduce repetition. We realized that the algorithm used to make a filled board was very similar to the algorithm used to solve a board. So, we created this method so that we could define a path for the algorithm to take in solving/filling the board, and it would try to fill in the board in that manner. If we did not have this method, we would have had to write the same thing twice.

2. SudokuBoard.getUnsolvedNums() - We recognized that most sudoku games tell the user what numbers are left to solve. So, we added this method to read the board and give which numbers are left on the board. This is used in SudokuBoard.toString() to nicely display for the user which numbers they have left to guess.

3. CommandLineUI.millisToClock() - We also recognized that most online sudoku puzzles time you. So, we added this feature to convert milliseconds to hh:mm:ss display for the user to quickly tell how fast/slow they were able to complete the puzzle.

Note for the difficulty levels:

1. Easy, medium, and hard will generate boards quickly in about less than 5 seconds. Impossible, however, will take a bit longer. In 100 tests it took around 43 seconds on average, but this could be longer depending on which laptop is used.