Tic Tac Toe

Assumptions:
1. The game will be played between two players.
2. The game consists of 3x3 matrix box where once a players makes a mark in any cell, can not be undone.
3. End result of game will be either of any three: Player 1 wins, or Player 2 wins, or Undecided.

Data Structure and Algorithm
After thinking for sometime I decided to use ArrayList as a data structure to implement the game. Since the game involves checking of state of every cell(Position) after player marks a cell, we need to make frequent get operation to the data structure to compare state of each position. If we user LinkedList for this case then there would be overhead of going through head to required node every time we need to make get operation at an index. It costs O(n) time for every get operation. Rather if ArrayList is used then every get operation would be constant time, O(1) since ArrayList uses index based data structure. The same could be achieved by using two-dimensional array, but doing so would require much nested loops and same overhead of going through other indexes. Use of linear data structure (ArrayList) gave choice of using switch case on specific index to determine the path required to check for game state(to find if game is finished). Since the data indexes were very few and unlikely to change, I thought using a switch case would not be bad idea.

I used Position class to represent each cell in matrix and collection of such positions was maintained in GameRunner class to organize logic.

Algorithm for finding the game win situation was optimized such that the checking was made based on current marked position rather than checking every position for their state after player marks a cell. So if a player marks n-th cell then only those cell containing row and column (and diagonal if any) through n-th index will be searched to determine if game has been won. I maintained a counter to find if all cells are marked and still no one has won. Such condition will end game as undecided winner.

Libraries/Frameworks
1. Used Data Binding Library from Android SDK to bind UI components with model class.
2. Used Dagger for dependency injection of GameUtil instance in GameRunner class.
3. Used Robolectric for tests.
4. Followed MVP pattern for organizing the code.