ISHIDO LOG

- Made Deck, and Board class as the main classes. The layout is that the whole screen has the Grid Layout and the board has the TableLayout. The model are separately stored in model folder.

- Made TableCoordinates class which will record the row and column number clicked in the TableLayout. There is no direct way to get the location. So, I made the class. Also, due to this I no longer have to deal with buttons and handle gigantic button id remembering or formatting.

- Deck holds the tiles generated and records it so that we don’t exceed the same kind for more than two times. It uses the 2 dimensional array.

- Board holds the basic entry in the grid with TileInfo objects. TileInfo consists of the color and the symbol of the tile. This class made it so much easier to handle the symbol values as a group.

Problems:

* I thought the color of characters were supposed to change. But it is color of the tile. **Possible solution**: change the background color of the TextView.
* The symbols are not coordinating with the ones that user chose.

**Possible Solution:** Check the clickedTile in the MainActivity and Deck class if both correspond to the same symbol notation.

1/31/16

* Solved the background color change
* Solved the weird symbol generation problem.
* Board class will calculate the score and also perform the check operation for the rules of Ishido. No big problem in this.

Todo:

Checking for if the game is over. **Possible Solution**: Maybe do isDone() functions in Deck and Board separately for more clarification.

Make the start Activity (like start screen)

* StartActivity made and designed.
* Problem arouse. The tile system was not working. It didn’t follow any rule. So, I spent like couple hours just trying to figure that out.
* I realized that while changing the color and symbol for user’s choice, I didn’t create a new TileInfo and rather just changed the color and stuff from current tile. This was the source of all of the errors. Fixed it.

2/2/2016

* Bug report: The orientation restarting the activity is unsolved.
* Included isDone() in Deck to check If we are out of all of the possible tiles.
* If yes, new Intent was created and it was directed to the game over page.
* Bug Report: In Game Over, it doesn’t exit the application on exit pressed. There are whole bunch of errors.
* There was problem with forever loop in generating random tile because of recursion. I made it a Boolean function and used isDone() of deck to check if the whole process is done.
* Bug Report: It doesn’t toast maxed out notice when user selects same combination for more than 2 times. **Solved**: forgot to put show() at the end.
* TODO: Make isDone() in Board

2/14/2016

* Added the FileAccess class that will handle and parse the file given to retrieve the strings of Board, stock and score.
* Added a new function in Board that will handle the board info parsed from the FileAccess. Also, since user does not have to click, I changed the board initially writing to screen to handle the filled tiles
* Made a stockQueue that is the linkedlist which handles the function of the queue of given stock. It is because we need to handle the given tiles from the first to last.
* Made the spinner and button for search options. Id is searchOK (button), searchHandle(function), searchChoice(Spinner)
* A bug found for automated system. The fillTile() checked for all of the side tiles to verify but it did not check if the tile selected is empty or taken.
* Now when we change the type of search we want to do, searchHandle () is called and it will check what type of search we are choosing. If the search type is being set for the first time, nothing happens. But if the search type is set for more than one time, then, the view of the table layout is deleted and new Board() is created in order to refresh the whole board for the new type of search.
* refreshSearchTable() is made for refreshing the search table everytime user wants a different types of search so that we can start from the beginning again. It creates new board, deck, searchModel, stock, and player.
* Finally done with DFS
* IMPORTANT CHANGE: Making a separate class for SearchModel creates a LOT of hastle and lot of annoying things to take care of for simple things. The View became very hard to update since we needed a unique condition for almost all of the variables and it was VERY VERY REDUNDANT.

2/15/2016

* BFS : lots of trouble. Tried doing the map. Got into a lot of hastle. Redid everything by replacing it with a new class TileTree that holds the TileInfo and TableCoordinates. This helps track the previous TileTree object that eventually helps us retract from the overall tree to the root node.
* Still have a lot of problems like tiles being replaced.
* POSSIBLE SOLUTION: Try to figure out when to fill the tile in the board and when not to. Only fill in the board if it is being placed in certain place like permanently. The guess of all of the tiles always being in a proper place is so wrong! Check for cell availability canFillTile() for everytime.
* Another error: check while assigning the master. It is not going through correctly. T3 is having T1 as master instead of T2.
* Assigning to the master the new tile from the queue was placed was wrong. Whenever we take a new tile from the queue, we refresh the table. It makes sense because it is actually a new search.
* Then, check for the parent tiles of the newly taken tile from the queue and place them on the board. Once placed, place the new tile. Then only check if we need to work with the new tile.
* Removed the stack of stationary files since it didn’t really make sense. We only push and pop one element. So, I changed it to a TileTree object.
* ERROR: The tiles were not being placed correctly. Once the available location was found, lets say (5,4), the column never started from 0 again even for next row. So, fixed it by changing the value of starting point of column to 0.
* TODO: Update player score accordingly.
* 2/19/2016
* - Added the bestFS() function.
* - Changed the name of the stack from dfsStack to searchStack hoping to
* make it usable for more search types.
* - Added an ArrayList that holds the Stack of TileTrees different for
* different scores. So, lets say we have 5 TileTrees in the board where we
* can get score of 2, the these TileTrees will be stored in a Stack in
* index 2.
* - This helped in pushing the TileTrees in open stack according to the
* order of score. (Taking first highest score).
* - Made a path stack which will hold the optimal end state path.
* CONCEPT: Take a new tile -- Find the available locations of it in the
* board -- If available, sort it according to the score and get the first
* location with highest score. -- Push it to the path -- Print it -- If
* no new location is found, then put the new tile back to stock and make
* the new tile as popped tile from open tile tree stack.

2/21/2016

* Tried a different approach for branch and bound search.
* For it we need to analyze all of the solution node (last leaf nodes in the given tree). So, I tried calculating the main algorithm first and then the printing part. It would take a long time if we have branch more than 4 since it is going to have a lot of them in the queue.
* Make performBNB() function where it will do the breadth first search until you find the solution node branch level. Once it reaches the solution node branch level, it will go through all of the solution nodes and then pick out the one that is going to produce the highest score overall.
* Added total score upto the particular node in the TileTree function. This helps us track the total score until that point. Helpful in all of the searches but just implemented in Branch and Bound. (too late to change it in the others).