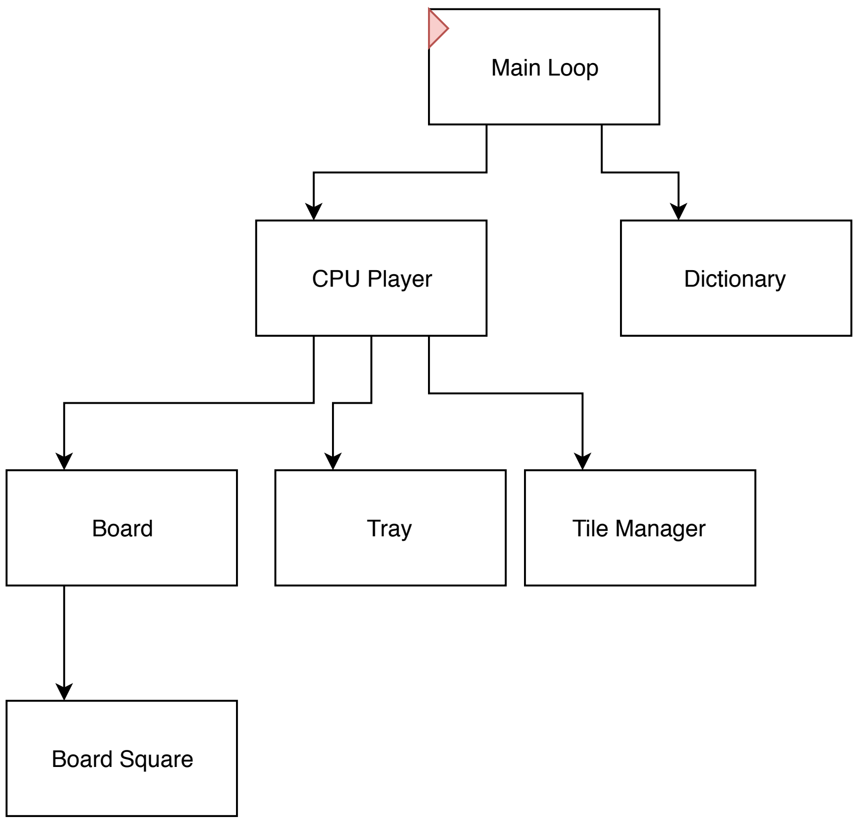
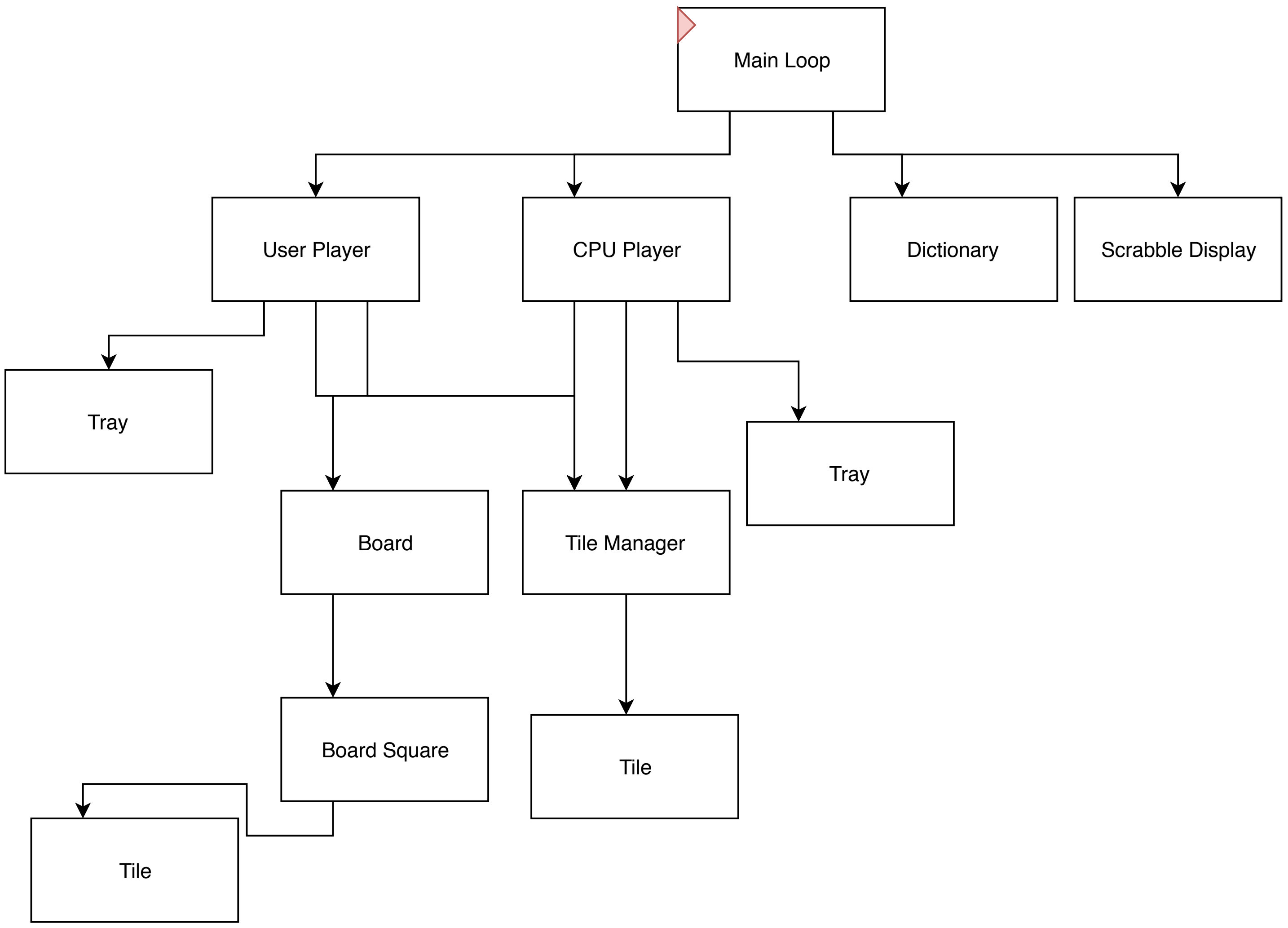
Command Line Solver



Scrabble Gui Version



Main Loop: The main loop of the game/command line solver, runs until reached EOF or until the user exits the GUI.

CPU Player: The cpu player for the game, always picks the best possible move, in the command line solver it is used for what play should be made, in the GUI the user plays against it, handles finding the best moves and then playing it

User Player: Handles some basic events for drag and drop, and sets up the tiles for the player so they can be dragged onto the board. Handles ending the turn, whether it was an exchanging turn or playing a move turn, shows feed back if the move is not valid using alerts

Tile Manager: Handles anything to do with tiles, like drawing or redrawing, also can get values of strings or of a character

Tray: Keeps track of what tiles the player has, every player has one. Handles anything a player should be able to do to tiles. You can show or hide a tray, which hides the tiles in the tray, or setup drag and drop for the tiles on the tray

Scrabble Display: Handles the fxml initialization and the set up for the user player, like setting some events to functions on the user player. Can also update the score display for the computer and the user.

Board Square: The object that represents a square on the board, can either just be multipliers or can contain a tile when that square has been played on. Handles drawing itself and playing tiles on it

Tile: The tile that represents a scrabble square. Handles drawing itself , can be hidden, can also setup drag and drop functionality on itself with a couple of call back functions. Used for almost everything in the game

Dictionary: The dictionary is implemented in one of two ways in my program. The first is a simple TRIE the second is a DAWG. The Trie has faster initialization but the DAWG has other aspects that are faster. The dictionary can be search to see if a word is in it you can also get the root node of the dictionary which can be used for transitioning when looking for valid moves on the cpu player. The Trie and Dawg each have their own node that extends from a base DictNode class. The nodes are able to transition, check if it is a word if you stop at that node, and get the map of transitions for that node. These basic functionalities are what makes up the search of the CPU player and by using transition maps for the Dictionary and a two part backtracking algorithm the cpu can generate all possible moves in less than a second usually on my pc.