<name>

<how objects will be stored>

<how stored objects information (name and function) will be identified from existing text file>

<how stored objects will be called identified (as above) and called during game from new list>

The difference between blue and orange is the lists that are setup. Blue is the process of taking pre-existing list and creating objects based-off that. Orange is the process of taking a newly created list of objects/numbers/etc and getting a specific piece from that.

Hard-Code:

Hard-code every piece as object, this means there is no flexibility and automation if the text file is changed, the object creation has to be manually changed with any new text file inputted

Creation of hard-coded if-statement or switch-case checks to determine the identify of a piece in the text file, then a hard-coded object will be created

Player does not enter co-ordinates, instead they enter the hard-coded ID of a piece (example White Rook 1, ID; WR1 or White Rook 2, ID: WR2)

Highest efficiency: n objects + n piece checks\*each iteration through file + 1 selection check in new file

Poorest efficiency: n objects + n\*n piece checks\*each iteration through file + 1 selection check in new file

Python method (2D Logical List):

All objects are stored in a single 2D list. This 2D list is identical to the text file in terms of object positioning, objects in the 2D list are logically located where they are found in the text file and blank spaces in the text file will be blank in the 2D list.

If-statement or switch-case checks used to determine the team and role of each piece (min: team found at 1st check, max: team found at 2nd check) (min: role found immediately at 1st check, max: role found at last check at 6th check since there are 6 roles.

Player enters co-ordinates to directly target a desired piece

Efficiency: n objects + variance of (1-2 team checks)+(1-6 role checks) piece checks\*each iteration through file + 1 selection check in new file

Highest number of piece checks is 8, lowest number of piece checks is 2

a method of displaying the visual board to the player with the 2D or 1D unstructured logical list method is for each iteration of the list, call the object’s method. This method will simply return its visual information which can be directly printed out, example Rook class can have a method which simply returns “WR” or “BR”, this will eliminate the need to have a list with “WR” and “BR” contained within it and instead the list can be used to contain objects.

Role-Based Object List:

There are only 6 objects, each corresponding to their unique piece and all objects are stored in a 1D list. Each object then has its own list of co-ordinates which represents the various numbers of this object on the board, example Rook object will contain 4 sets of co-ordinates, 2 for white rooks and 2 for black rooks.

The 6 objects will be pre-created. If-statements and/or switch-case checks will be performed on the text file to identify piece of the object and then pass these co-ordinates into the appropriate object’s method. This method will create and store the co-ordinates for future calling.

Player enters co-ordinates, a separate visuals-only list that used to print to the player is already created and the co-ordinates are checked against that list. From this the piece and therefore the object can be identified. Then the object’s appropriate method is called.

Efficiency: 6 objects + variance of (1-2 team checks)+(1-6 role checks) piece checks\*each iteration through file + 1 selection check in new file