Report for Scotland Yard Coursework

Part 1 Coursework Model:

In the first part, we implement data attributes which is needed to hold information related to the feature of the game : Game setup holding graph game, MrX moves', moves holding the current possible moves, MrX , list of all detectives players, remaining player in the game and MrX trace.

Next, we implement constructor to keep up to date game once the current state of the game change. We did this by using this.??=?? command, which update all the attributes described in the first part once one of the player makes a move.

Now, let's talk about getter method, we use getter method to return correct value of attribute variable we want. For instance, we use get available move to determine all possible MrX in the current turns and get MrX travel log to check track and know current situation of MrX. All of this will be used for further part of implementation.

For build function, it is a point we need to check about the invalid information, including whether MrX is an actual MrX ; whether the mrX present in the game; do we have correct number of MrX and detectives or not; Do all the player have different location?; Do all player has valid tickets or not?; and, Moves is empty when it should be empty or not.

To get all possible moves, we implement 2 separated functions myself: make single move a make double moves. This function can be done by iterate through all possible move in case of double move or just next move in case of single move.

Let's move on to the advance part, for this part, we use visitor pattern, which can be implemented by dividing into 2 cases: single move and double moves, to access source and destination of the current move. Then, we add used tickets in the current turn to the log entry and update the state of the game.
The last function of the first part is getting winter, we just check whether max have made a certain number of moves or not, detectives run out of ticket or move and MrX is captured or

not. If at least one of these holds, then we end game return winner (check whether it is MrX or detectives).

In the second part (Model observer Test), we set up the state game setup, all players (MrX or detectives), board game and list of observers as attributes. Then, we can initialise all of the attributes according to the state of the game.

To register new observer, just add an observer to the observer set. On the other hand, to unregister new observer, do the opposite by remove an observer from the observer. Aside from this, we also need to check edge cases where input information is invalid.

For the function to move, we just use get winner function the check if winner(s) exist or not.

If so, we can return game over event. Else we can return move made event.

For build function, we just return my model function.

Part 2 Coursework AI:

My AI use the distance base strategies with we choose to move by using the move with the longest sum of distance between MrX and all detectives. I used Dijkista algorithm to calculate distance from each detective to MrX. This algorithm can be implemented by setting source and destination using visitor pattern as the previous part of the coursework (part 1). Then we can iterate through each possible move in get available move. Then check which move is the shortest. If there are several possible moves, then we are going to choose one at random.

Merit: moves is chosen to depend on the current state of the game and will not produce any random behaviours. I think this is much better than choosing random moves.
Limitation: I do not 100 % consider different between single and double tickets, which those differences may result in making in much more optimal moves. Only thing I consider in the availability of the ticket.