Report for Scotland Yard summative coursework, by Joseph Cox (wz21309) and Oscar Martin Dickie (pj21274).

Summary

The cw-model portion of the project was finished passing all 83 tests without warnings in any of the classes we contributed to whilst compiling and running successfully. Within MyGameStateFactory, each attribute of the MyGameState class is initialised with its own initialisation method which performs the required checks and throws exceptions when one of them is not passed. The majority of these methods are simple, but initialiseAvailableMoves and initialise winner are more intricate.

initialiseAvailableMoves first checks if Mr X is in remaining. If he is, then all of the possible single moves Mr X can make are found using the makeSingleMoves method and put into a set of their own, as well as totalPlayerMoves. Then, if Mr X has any double move tickets, then the method makeDoubleMoves is used to find all of the possible double moves that can be made, by applying makeSingleMoves to the destination of each single move he can make and then compiling the pairs of moves into a double move, and returning the full list of double moves to add to totalPlayerMoves. Then, totalPlayerMoves is returned. If remaining does not contain Mr X, then each piece in remaining is matched with a player in detectives, and then the moves it can make are found with makeSingleMoves and are added to totalPlayerMoves. One all pieces remaining have gone through this process, totalPlayerMoves is returned.

initialiseWinner firstly creates a set of pieces containing just the detectives. Then one by one the detectives are checked if they have any tickets left and if they are in Mr X's space. Then, foundWinner is set to Mr X if no detectives have tickets left or if the travel log is full and the foundWinner is set to the detectives if Mr X is caught or Mr X cannot move anywhere on his turn.

The next important method is advance. After making sure the move it takes as input is legal, advance implements the visitor pattern to return an array of single moves. If the move is a single move, then this array only contains the move in question. If the move is a double move, then the array contains both of the moves that make up the double move. All moves within the array are added to Mr X's log using the updateLog function. Then, either resolveSingleMove or resolveDoubleMove is called depending on whether the array only contains one move or not. In both cases, updatedPlayer is set to the player who is making the move. Once this is done, if the move was made by Mr X, then newMrX is set to the updated player, and otherwise newDetectives is made as a new immutable set of the unmoved detectives and updated player. After this, remaining is updated with the updateRemaing method, and a new gamestate is created.

Within MyModelFactory, the observer pattern is implemented. Within the chooseMove method, the advance method of the state attribute is called, and then depending on whether the game is over or not, all observers will be notified of when a move is made and when the game is over.

The cw-ai portion of the project was attempted. We created a basic AI for Mr X using a scoring function to score all the possible moves that Mr X could make and then choosing the highest scoring move to make. The score is calculated using 6 different aspects of the move and the resulting board. These are the amount of adjacent nodes to the destination, the distance from the nearest detective at the destination, the average distance of the detectives at the destination, whether the move is a double move, whether the move is a secret move and how far away the destination is from the edge of the board. The pickMove function incorporates Dijkstra's algorithm to find the distances between all nodes on the board from the destination of each possible move and implements the visitor pattern to find the destination of the move in question. The visitor pattern is also used in the checkDoubleMove method to check whether the move is a double move or a single move.

Reflection

First of all, the cw-model portion of the project was completed with all tests passing and in a readable and efficient way. In particular, using makeSingleMoves to find all of the possible double moves for each single move's destination was an interesting way to reuse some of our code to solve the problem of finding all the possible moves that could be made without error. Furthermore, our use of the visitor pattern to return an array of single moves that could be processed separately with updateLog and then applied to the moving player was quite efficient and made Mr X's travel log as a whole much easier to implement. Finally, we were able to implement the observer pattern in a concise manner that is easy to understand from looking at our code.

Secondly, we did successfully program a functional AI for Mr X. Our implementation of Dijkstra's algorithm is efficient and allows all of the methods that need it quick access to the distances between the possible locations of Mr X and any node on the graph. Our use of the visitor pattern was necessary in order to differentiate between single and double moves when dealing with an array of moves of both types. Also, our score function taking into account so many factors other than the distance of the detectives and weighting them differently to prioritise some aspects of the board after the move more than others means that although our AI is only a simple score function, we believe it is quite effective and look forward to see how he performs against others in the near future.

Although proud of the work we have done, we would have liked to implement a much more advanced AI. The AI we have made only looks one move ahead, and although we have taken into account quite a few aspects of the game, there are some aspects that our AI does not take into account that we would have liked to implement. Improving our AI as it is, we would have liked to vary how the different variables are weighted based on the point in the game. For example, prioritising distance from the nearest detective on the turns before Mr X's location is revealed. As well as this, in its current state our AI is unlikely to use secret tickets in a strategic manner, and will almost always only use them to travel by ferry. We would have liked to take this into account, possibly by increasing the score of moves using secret tickets when detectives are nearing Mr X. With enough time, we would have liked to implement the minimax algorithm to make an AI that is able to look ahead into the future by more than just one move.