COMS10017 – SCOTLAND YARD PROJECT

Patrick Lee

**PART 1 – MODEL**

Outline

The first task was to build the ‘Game State’ and ‘Model’ Factory classes such that it would pass all the given tests to ensure they would act as foundations to the Scotland Yard game, essentially creating implementation for an object “game state” which would provide all details about a particular game state including the arrangement of pieces on the board and the moves available to them. This was linked with the “model” implementation which could register and notify any external observers about game events such as moves and if the game has ended.

Our implementation was completed successfully, passing all 83 of the given tests and making use of a variety of techniques and OOP concepts learned in the past few months.

Helper Functions

In addition to implementing overridden methods from the “GameState” interface, we developed several helper functions within the surrounding “MyGameStateFactory” class to break down some of the more complex problems and reduce the size of the “MyGameState” class and its constructor. Helper functions were created for the following specific tasks:

* Given a list of detectives, determines if they are all distinct and valid. Returns Boolean.
* Given a location and list of detectives, determines if the location is occupied by a detective. Returns Boolean.
* Given a list of Player objects, returns an Immutable set containing the each Piece object linked to each of the Player objects.
* Given a Move object, uses the Visitor Pattern to return an Immutable Set of the destination(s) reached/passed through in that move.
* Given the MrX Log, a GameSetup object and a Move object, returns a new MrX Log (As an Immutable List of Log Entries) which contains an entry for the most recent move made my MrX.
* Given a Player and Move object, returns the same Player object but updated so that their location and tickets reflect the move they have just made.
* Given a MrX Player object and list of detectives, returns an Immutable Set containing all detective Piece objects and the MrX Piece object.
* Given a list of Player objects, list of detectives and a GameSetup object, determines if the the given players can move or not. Returns Boolean.
* Given a GameSetup object, list of detectives, Player object and Integer representing a location value, returns an Immutable Set of all Single Moves the Player can make.
* Given a GameSetup object, list of detectives, Player object and Immutable Set of single Moves, returns an Immutable Set of all Double Moves the Player can make.

OOP Techniques

* Classes, objects, attributes, methods and encapsulation – Attributes of objects instantiated from the “MyModel” and “MyGameState” are initialised in the constructors and only accessible via the getters and setters which we implemented based on the interfaces for each class, thereby ensuring attributes can only be modified and retrieved via specific methods.

Evaluation