Scotland Yard Model Report

This documentation will elaborate the methods and its usage throughout the implementation of ScotlandYardModel.java. We agreed not to add any extra classes in the implementation methods of ScotlandYardModel since understanding and availability of functions from other classes provided suffice the purpose of the project. Upon completing the project, We have made use of the tests provided in junit and instructions in example workflows to build up the program in a consistent and progressive manner. The following implementation and methods are all within ScotlandYardModel.java.

First, We started initialising all the required attributes in the constructor. For all the self-developed helper functions or attributes, we have assigned the access modifier 'private' to encapsulate them within the class. Next, we proceeded by giving implementations of the methods in the interface ScotlandYardView. We encountered the concept immutables for which we made the returning values of the method immutable with method provided by JDK/skeleton code and optional to ensure that there is always a returning value even if the parameter of the method is not in the scope of the game.

In the accept method of the callback, we implemented visitor design pattern,observer design pattern,polymorphism as well as overloading. This is evident from the visit method in the MoveVisitor which shares the same name but different parameter type. In addition, we also divide the process of notifying spectators into smaller functions such as *notifyOnMoveMade(), notifyOnGameOver()*, etc. Likewise, we made createNewTicketMoveForMrX(Ticket x,int y), playerIncrement(), isRevealRound(),etc as DRY helper function. This makes the code in shorter and increases the reusability of the functions. For the order of function executions, we followed the UML diagram provided and debugged as we saw fit.

Last but not least, something worth noting is that we have assigned PassMove to MrX when it cannot move to make the condition check for isGameOver even easier. Also, we tailored and restructured the program by moving the implementations of isGameOver to getWinningPlayers as they both share same conditions for returning value.

AI Report

Our AI is based on Djikstra's algorithm, mainly selecting the move that is the furthest away from the detectives by counting the number of steps it takes to reach the detectives. We had to implement a new visit function in order to retrieve the locations of the detectives. This is through the overriding of accept (Move move) method. The AI also forces us to re-do the graph of the game and visit class which allows us to utilize classes of OOP and objects to organize our code. The AI algorithm also considered the priority of tickets utilize. For example, it will select a move which do not use doublemove since there are limit tickets on doublemove and doublemove should be saved as "last resort" in certain situations.

We have explored the possibility of implementing Minimax algorithm with alpha beta pruning but due to time constraint for both of us and approaching exams, we are forced to stop half way since there are a lot of implementation details we didn't figure out in the planning steps. These includes creating our own "BoardState" or the Scotland Yard view of the game. We also couldn't decide between creating a game tree of the "BoardState" itself or game tree of the scores generated by the scoring algorithm. Further Explanation of the files in AI can be found in README.txt in the AI part.

Personal Reflection (Dhammatorn Riewcharoon):

This project has taught me not only the basics of object oriented programming, but also how to cooperate with another member to complete the same goal with the use of git. I am certain that this project has broaden my understanding of OOP concepts such as polymorphism, inheritance, encapsulation, Lists etc. Overall, I think my partner and I have done a great job as a team in communicating and working together in one direction. I have learnt that explaining your own code is an important part of being a developer.

Sometimes me and my partner will work separately and write our own code in our own ways. We will then have to sit down and discuss which one to use, weighing their pros and cons before deciding which one to push onto git.

In addition, this project lets me explore the use of git and its pros and cons in working together on the cloud. It is a process of learning how to use git through out the development of the project. We had some fail attempts to push and pull each other's code, leading to making new repository or reverting the version of git. This taught me that before pushing or committing version, one has to communicate and agree on one version to prevent such problems down the road.

Moreover, I thought about creating extra classes which would have demonstrated that we both have a deep understanding of the OOP concepts.

Planning the AI part allows me the explore more into Object-Oriented concepts and algorithms such as Dijkstra's Algorithm and Breadth First Search. Creating my own classes, valid move objects, overriding the accept method and creating superclass to subclasses.

Personal Reflection (Yap Kar Hor):

Completing ScotlandYardGame has not only consolidated my programming skills in general but also enhanced my understanding of Object-Orientated Programming, in particular pivotal Java concepts. Doing this project made me realise that working alone is never a good choice especially if you are stuck at the bottleneck. For example, trying to debug on my own might just take half the effort if i work with my partner or consult TA during the lab session. It came to light to me that exchanging thoughts and opinion with people is a more efficient and productive means in dealing with a big project.

Besides, throughout the development of the project, i have learnt how to effectively utilise version-controlled git application to update our respective code as well as communicate to agree on certai different implementations. This enables us to review our previous code if any undesired changes have been made and keep track our progress. Other than that, i also learnt that instead of debugging the program just for the sake of passing the tests provided, approaching the problem through logical interpretation based on solid understanding of the program flow helps one to make huge progress at a later stage.

One of the biggest challenges we faced was not recognising the dependency of a function on another function which caused the inconsistency of the results of the spectators being notified. After due discussion in vain, we turned to the forum for help and managed to get constructive advice on the possible cause of the outcome.

In a nutshell, working through the project from scratch is indeed a great experience as we are able to implement most of the concepts taught in lecture in a practical and engaging way.