Plan of Attack

-Code pull requests will be under review by everyone  
-Daily Stand-up to ensure everyone is on track

|  |  |  |
| --- | --- | --- |
| **Dates** | **Task** | **Assigned To** |
| Nov. 18 | -Initial Meetup  -Read through guidelines/all games  -Discuss implementations for each game | P + N + Y |
| Nov. 19 | -Select Game  -Each draw rough UML sketch | P + N + Y |
| Nov. 21 | -Create Github Repo and secure a system for version control  -Make sure everyone has good understanding of version control/Github | N |
| Nov. 20 | [Meetup]  -Start drawing UML together on paper  -Compare with A4  -Discuss design patterns [Observer] | P + N + Y |
| Nov. 23-25 | -Draw UML and iterate constantly  -Submit UML | Y |
| Nov. 24 - 25 | -Create shell files using UML for all classes including hidden fields/methods in UML  -No implementation | P |
| Nov. 25 - 27 | -Code Cell Class  -code calls to ability | N |
| Nov. 25 - 27 | -Code Player + Ability Class  -move(), battle(), download() | P |
| Nov. 25 - 27 | -Code Link Class  -Text Display, Graphical Display -> Observer Pattern | Y |
| Nov. 27 | -Create Test Cases | P |
| Nov. 28 | -Group Testing | Y + P + N |
| Nov. 29 | -Code Fixes | Y + P + N |
| Nov. 30 | -Code Clean Up/ In Code Documentation | Y |
| Nov. 1-2 | -Potential Bonus Marks | Y + N + P |
| Nov. 1-2 | Final Design Documentation | N |
| Nov. 2 | Submission | Y + N + P |

Questions

**In this project, we ask you to implement a single display that flips between player 1 and player 2 as turns are switched. How would you change your code to instead have two displays, one of which is player 1's view, and the other of which is player 2's?**

We will have the same implementation in our code, except the fact that in our **Subject** class, there will be two instances of observers, 1 for each player. Then, whenever, player1 has a change in the game, only the parts allowed to be viewed by player 2, will be viewed by player, and vice versa.

For the scenario, where we are not sure how many players will be playing (*i.e.* may be 3 or 4 players, we do not know), we will change the code in **Subject** class. We can simply have a vector of observers, where the necessary changes are updated accordingly.

**How can you design a general framework that makes adding abilities easy?**

Abilities, so far, can be classified into two categories:

1. Abilities which can affect the cell
2. Abilities which can affect the link of mine and/or opponent

When adding a new ability we will add the Ability to our Abilities enum, deduce what type of object it modifies, add necessary fields to those classes to monitor ability flags, and data, the implement what to do when ability is used in our Game class.

**One could conceivably extend the game of RAIInet to be a four player game by making**

**the board a plus shape (formed by the union of two 10x8 rectangles) and allowing links to escape**

**off the edge belonging to the opponent directly adjacent to them. Upon being eliminated by**

**downloading four viruses, each links and firewall controlled by that player would be removed, and**

**their server ports would become normal squares. What changes could you make in your code**

**to handle this change to the game? Would it be possible to have this mode in addition to the**

**two-player mode with minimal additional work?**

For the board, we will have a 10x10 square, with each of the corner cells declared as invalid cells (where no abilities or links can go). For the edge scenario, we only need to change the move method inside to check if the edge boundaries cover the respective edges (for download when crossing edge)

Whenever a player is eliminated the destructor function would be called and that would remove all the links from the cell, as it would normally do for the 2-player game. However, when it comes to firewall, we would have to consider in such a way that destructor also destroys all the players firewalls on the board.

To solve this: Whenever a player cast a firewall in a cell, the player keeps record of the firewall it created, as a field (as a vector of *firewalls*). Now, when the player gets eliminated, we take care of the firewalls and destroy the firewalls accordingly. Similarly, the player who gets eliminated, will have their end of the edge server pots turned into normal cell, since server ports are just another cell with different functionality.

For the scenario, where we are not sure how many players will be playing (*i.e.* may be 3 or 4 players, we do not know), we can simply add to the vector of observers, where the necessary changes are updated accordingly. Now, 1 player sees everything according to the rule of the game, and not anything extra which may tip the player to cheat.