CS3098 Progress Report: Group C

INTRODUCTION

For our game board implementation of Settlers of Catan, we have been coding a basic game setup. This includes code to initialise a randomised game board, data structures to store data about edges and intersections, storing the resource cards and development cards and allowing the players to place their first roads and settlements.

Using this setup, we can then start implementing the game logic itself, allowing users to actually play the game.

TOOLS

We are using Eclipse to write our code on, as we are most familiar with this IDE, and it also allows us to easily implement JUnit tests.

In terms of source control and project management, we are using GitHub and Trello. The majority of the team have used these tools in past team projects and so we felt that it was the best decision to use these again. Using Trello greatly helps organising tasks and also keeping collaborative documents in one place. To communicate outside of labs, we use a Facebook group chat. We felt that this was better than using Slack as we all use Facebook more and so would be more likely to check for messages.

For inter team collaboration, we use a sub channel on the stacs Slack. This helps to keep communications in place pretty well. We also use GitLab for the shared protocol. Using git for source control helps keeps it consistent with our own teams source control also, and so this was a good decision.

CODE

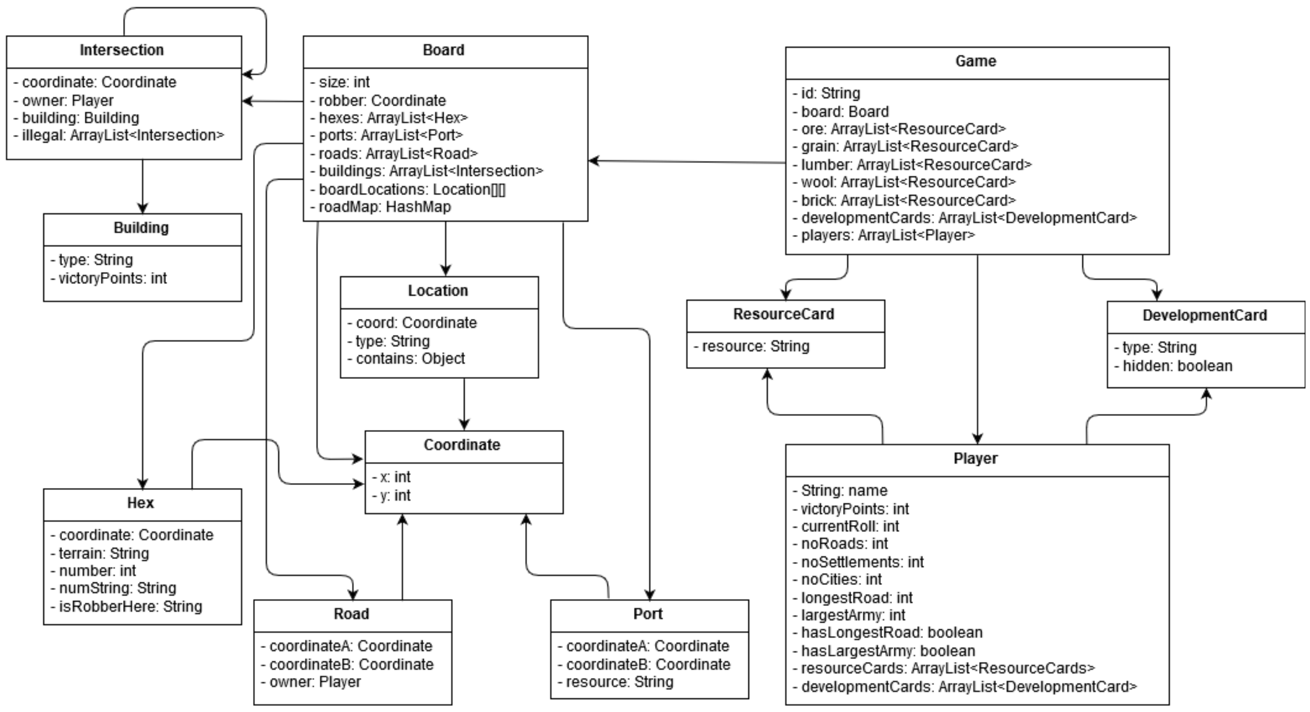
We are using Java to write the logic of the game board itself. We feel that this was the best choice as all members of the team are most knowledgeable about Java and so we will be able to achieve a better end product.

For testing our game logic we will use JUnit, as again we are most knowledgeable with this package.

GAME LOGIC

To implement the game logic we thought it would be best to fully understand the game first. As a team, we played ‘Settlers of Catan’, as well as playing online and mobile versions, and tried to discuss ways we could code each part and what difficulties we would face. From this, a document was written which split the rules into sizeable chunks.

Each team member chose a section of the rules that played to their strengths, and we began writing pseudocode to have an outline of what our code would look like. Looking at the pseudocode written, we decided on suitable classes that we would need to make an efficient solution. Our class diagram is shown below:



Once we had this outline in place, we used paired programming to start coding the game logic fully.

We began to Pair program and below is an outline of some of the logic that we were able to implement at this point and time.

Hex board design

Initially we have 19 Hexes. All hexes have three attributes that are visibile in the initial UI. So each Hexagon has a unique coordinate in order to define its position on the board. Also, it has a label for the resource it represents (P = Sheep, F = Wood or Forrest, M = Mountain, and H = Clay, and D= is for desert. Each hexagon also displays a value 2-12 representing the dice roll value that will activate the resources distribution. There is also a field that is empty for all but one hex. This field, “isRobberHere”, is used to represent whether or not a certain hex contains the Robber. If the hex contains a robber, the activation code or number for that Hex will no longer work until the robber is removed.

Roads

In the initial design, each road is an object which is represented by two coordinates and a player or owner of the road. Currently the road is represented by 1 (which is hardcoded) meaning that a road exists on the board. However, we will change this feature as we develop the board to make 0 represent no road and 1 represent a road.

Intersection

The intersections of the roads or the hexagons are represented by “o”t. The “o” is to represent the owner. Each owner will be given a significant ID concurrent to their color (Blue = B, Green = G , Red = R, and White = W.) However, at this point instead of colors we have identified the players with special characters. The colors will be implemented later on. Before the intersection has a construction on it the road will contain a blank before the t and if a construction is build the intersection will be represented with B, G, R, or W to represent the owner. “T” is used to show what kind of construction is available on the intersection. If it is a town the intersection will be labeled “t” and if it is a city the intersection will be labeled “c”.

Ports

The ports are constructed on the boards of the hexes. To implement the ports, we labeled them as P-. The – represents the edge the port is pertaining to. A more complex implementation of the port will available later.

Resource and Development Cards

An array list was established to hold the development and resource cards. The development cards were added according to the rules. There are 25 development cards: 14 Knights, 2 road building, 2 year of plenty, 2 monopoly and 5 victory point cards. Once these were added to the list the order was randomised so the deck was shuffled. 5 resource card decks were made which store 19 cards for each resource.

Set Up

The implementation of the board set up was also started. Features such as getting the number of players and letting the players place the first roads and settlements. Statements were printed to ask the users how many players there were. From this each player was allowed to choose a unique ID from a selection of characters (!, #, %, &, @, *). This would be the character to show where this player had placed roads and settlements. The players then roll the dice, which is done by generating two random numbers between 1 and 6. The players then play in order of highest roll first, and begin placing their first roads and settlements.

AI

As we are a joint honours group, we have not made an AI at this point in time.

UI

We are currently working on using the console to print out the board using an ASCII UI. At a later date we will improve this to use a graphical 2D UI which will look a lot more professional.

The board is printed as a series of hexes that resembles the Settlers of Catan board. Inside the hexes, specific data is shown. This is the coordinate of the hex, the terrain and number token placed on the hex, and whether the robber is on the hex or not.

To represent roads, settlements and cities, we allow players to have a unique character to represent them. This character is printed on edges and intersections that they own, so players can easily see where tokens have been placed.

To allow players to interact with the board, we use a test based interface. A message is printed to the console asking the players for an input, which is read in and so players can play the game. This allows players to specify coordinates to roll the dice and build their initial roads and settlements. Eventually, this interface will let the player buy and play development cards, as well as trade with other players.

INTER GROUP COORDINATION

A member of our group, Taylor, has been attending the biweekly inter group meetings. At these meetings the groups discuss what the protocol should be.

The first few weeks we decided on what messages should be sent to the clients, and what computation should be carried out by the server and by the clients, to ensure every client has the same version of the board at all times, to avoid syncing issues. The coordinate system that each team would use was also decided.

In later weeks we started to write the protocol, using protocol buffers. This was a good choice as it gives a more concrete class system that the teams should use, making the implementations easier to coordinate.

Our team has also been cooperating with the other two joint honours teams, and we plan to meet up and discuss our implementations. As we all are currently implementing the game logic, we feel that it would be a good idea to decide on an interface for the other teams’ AI to interact with. We also think that creating a universal set of tests would be a good thing to cooperate on as each team’s game logic should be the same.

TESTING

We do not currently have tests in place to test the game logic. We feel that writing unit tests will be a good idea, as it will make it easy to pinpoint errors within the logic we have written.

FUTURE

We plan to have a working version of the game board and logic ready for the semester 1 demo. We plan to do this by further using paired programming and cooperating with other joint honours teams.

Looking forward towards the next semester, we plan to start looking at migrating out logic from an ASCII board to a more responsive UI. We also plan to start working with servers and get the board to work with other team's boards and AIs by using the protocol that has been decided on in the inter group meetings.

CONCLUSION

To conclude, we have made good progress in implementing the setup for the game board, by creating methods to randomise hexes, terrains and numbers as well as creating classes to store information about road and settlement location. We also have an text based UI in place, that will be further fleshed out before the end of the semester.

We hope to show you the board we have created in action in the demo.