Code Files:

Server Side:

- *Server*: Server does 90% of the work. Server receives and sorts incoming messages to decide if they are a new client or old. For old clients the Server updates the track based on their message. For new clients Server sends all the relevant track data to get them initialized.
- *Client*: Holds information about clients that are connected to the server.
- *Track_Data*: holds all the necessary information about the expected state of the track.
- *INetworkUtils*: holds the constants for the types of messages being sent and received and the port to use.
- *Program*: starts the server (main).

Client Side:

- *Track*: Main driver object. Invokes most other objects with updates and communicates through Client.
- *Train*: Stores speed, username, location of sprite, section in, and current target. Moves when Track tells it to. When the current target is reached Track will pass it a new target received from the path stored in the section the train is currently in.
- *INetworkUtils*: Utility interface that holds the constants for the types of messages that can be sent and the IP and port of the server.
- *Client*: manages the sending and receiving of messages from the server.
- *Section*: Stores and iterates through its Path as told by Track. Locks when entered unlocks when exited. Stores if the current train is traveling Path in reverse or not.

- *Path*: Stores waypoints through a section in order. Iterates from 0 - N when asked by section normally. Iterates from N - 0 when asked by a reversed section.
- *Waypoint*: Contains its own order in the track and a location.
- *Endpoint*: Bookend for a section. Contains direction facing (destination to use), destinations (sections the endpoint leads to), and whether it is active.
- *Turnout*: Inherits from Endpoint. Changes its direction when clicked if active.
- *Passthrough*: Inherits from Endpoint. Placed in corners where there are only two sections touching. Not clickable tries to allow trains through automatically when told by the Track it should flip.
- *IDirections*: Interface to store constants which translate Up, down, left, and right to ints

Extra Information:

Starting the Server: To start the server there are a few steps. First you will need to cd to: */home/shepjc/Capstone/publish***.** Once you are in the correct folder you need to run the command: *dotnet --roll-forward Major Capstone_Networking_Project_Test_0.dll***.** Using this command the server will terminate when you log out or time out of your shell; to keep the server running 24/7 add *nohup* to the beginning of the command. When using nohup make sure to mark the pid that the server is running on so that you can kill it later on with the command: *kill -9 PID_HERE***.**

Changing the Configuration of the Track: Before changing the track remember that a section can only go one direction up, right, down, or left. It CANNOT turn a corner because then the section would go two different directions and the reverse traversal algorithm would not work. If you want a section of track to have a corner add two sections attached together by a passthrough.

Changing the track configuration has a few steps. My first step is to add the new tiles from the tile pallet window into the tile map so that you can align your endpoints and waypoints easily. Next I usually duplicate an existing section. Rename the new section and change the order to be the next available. Adjust the position of the waypoints in the section and add more if need be. Remember to change the id numbers of the waypoints such that they do not overlap with existing ones (these are important for when setting up other trains on track it must have a unique waypoint to target). Next set the orientation of the section to match the direction of flow from the lowest indexed waypoint to the highest. So if 15 is at the leftmost edge and 17 is at the right the orientation of the section is right. Next adjust the endpoints attached to this section so that it is possible to enter the section that would entail changing the destinations variable for the turnouts attached to the new section so that if the section is above the up part of the array holds the section index (order). Remember when adjusting that the destinations are 0:Up, 1:Right, 2:Down, 3:Left; the same as a compass. Create signals for the section if new ones are needed and if not make sure to adjust the signal array so that it references the ones bookending the new section. After completing these steps the new section should be traversable.

My Stable Track Configuration: