MetroSim Design

* Organization
  + Passenger
  + PassengerQueue
    - Using <list>
    - A list of Passenger elements
  + Train
    - A vector of PassengerQueues
    - Each PassengerQueue in the train represents a car (organized by passenger destination)
  + Train Car
    - A PassengerQueue in the train
    - When a passenger gets on the train, they are added to the train car of their destination
  + Track
    - A vector of PassengerQueues
    - Each PassengerQueue in the track represents a station (organized by passenger station they enter the MetroSim at)
  + Station
    - A PassengerQueue in the track
    - When a passenger added to the simulation, they are added to the back of the queue of their “from” station
  + MetroSim
    - Includes 1 train and 1 track
* Functionality
  + When the train moves
    - Any passengers who are at the station that the train is about to leave from:
      * First, get added to the train in their respective cars
      * Second, get dequeued from the station
    - When the train arrives at the next station, any passengers in the car that matches up with the station:
      * First, are written to the output file
      * Second, are dequeued from the train car
* Classes
  + Passenger
    - Id
      * Passenger ID
    - From
      * Where the passenger enters the MetroSim
    - To
      * Where the passenger gets off the train
    - Print
      * Prints the passenger information
  + PassengerQueue
    - Id
      * Number of the station or train car it is meant to represent
    - Name
      * The name of the station or train car it is meant to represent
    - std::list<Passenger> passengerQueue
      * A list of Passengers
    - Passenger front();
      * Returns the passenger at the front of the queue
    - Dequeue();
      * Removes the first passenger from the queue
    - Enqueue();
      * Adds a passenger to the back of the queue
    - Size();
      * Returns the size of the queue
    - Print();
      * Prints all passengers in the queue
  + MetroSim
    - std::vector<PassengerQueue> train;
      * A vector of PassengerQueues (cars)
    - std::vector<PassengerQueue> track;
      * A vector of PassengerQueues (stations)
    - numStations;
      * The total number of stations on the track and number of cars on the train
    - currentStation;
      * The station that the train is currently at
    - numPassengers;
      * The total number of passengers to have been added into the simulation
      * Used to assign passenger ids to new passengers
    - output;
      * The output file to write to when passengers exit the simulation
    - MetroSim(string stationsFile, string outputFile, string commandsFile);
      * Creates a new simulation from a stationsFile, an outputFile, and a commandsFile
    - MetroSim(string stationsFile, string outputFile);
      * Creates a new simulation from a stationsFile and an outputFile
    - takeCommand(int argc, char \*argv[]);
      * Takes command from user input and executes the appropriate action
    - readStationsFile(string stationsFile);
      * Helper function for creating a new simulation, reads the stationsFile and creates the stations and train
    - writePassengerToOutputFile(Passenger passenger);
      * When a passenger leaves the simulation, write them to the output file
    - printMetroSim();
      * Prints the current state of the simulation
    - addPassenger(int arrivalId, int departureId);
      * Adds a new passenger to the simulation
    - moveTrain();
      * Moves the train to the next station
* DepictionA blue oval with black background

  Description automatically generatedA black background with orange squares

  Description automatically generatedA blue and green squares

  Description automatically generatedA black oval with blue and green squares

  Description automatically generated
* Schedule:
  + 10/12:
    - Create MetroSim, train, and track
  + 10/13:
    - Write functions for MetroSim, train, and track
    - Test the functions using basic tests to ensure they work under basic conditions
  + 10/15:
    - Work on functions involving reading commands, reading files, and writing to files
    - Test the functions using basic tests to ensure they work under basic conditions
  + 10/16:
    - Work on error handling
    - Ensure that any commands that can’t be processed are handled as such
  + 10/17:
    - Put everything together and start testing the entire program
  + 10/18:
    - Come up with as many tests for edge cases as I can
    - Test them
  + 10/19:
    - Ensure that everything is formatted correctly, and code is commented
    - Write README
  + 10/20:
    - Submit
* Edge Cases
  + Trying to add a passenger with a station id that is outside of the range of stations in the simulation
  + Dequeuing a passenger from an empty car
  + Adding a passenger to the simulation with the same arrival and departure stations
  + Entering commands that either lack information or are not commands that the simulation can take
  + When 2 passengers have the same from and to stations, making sure the first one that was added to the simulation is the first that gets printed or written to the output file