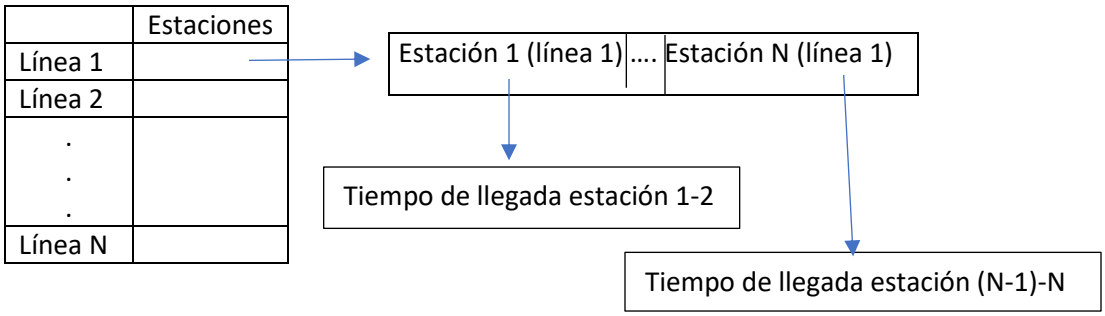
Informe Desafío II

Integrantes: Juan José Balvin Torres – Juan Pablo González Blandón

Análisis del problema y consideraciones para la alternativa de la solución propuesta

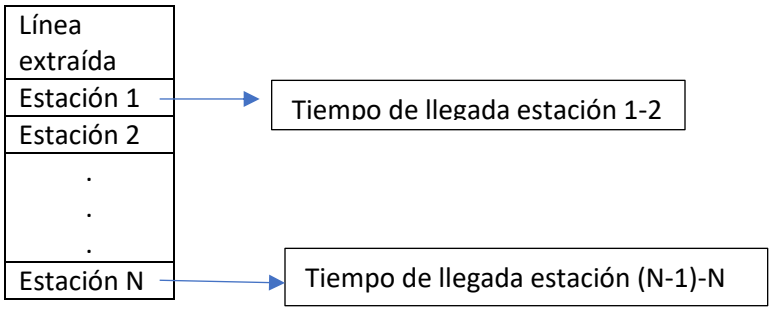
El desafío propuesto consiste en modelar una red metro con ciertas características expuestas en el PDF del desafío, teniendo en cuenta estas mismas y analizando como podemos llevar a cabo el desafío, la solución planteada es la siguiente: Pudimos identificar dos puntos clave para su desarrollo, estas son una vista general la cual llamaremos "red" y una vista un poco mas reducida la cual llamaremos "línea" ahora bien por que decidimos llevar nuestro enfoque a esas dos clases principales, lo pensamos de la siguiente manera: Necesitamos modelar un comportamiento total sobre la red metro y realizar ciertos cambios y operaciones sobre esta que influirán no solo en una línea sino en toda la red en general así que para ello es necesario modelar la vista general, sin embargo entre las operaciones que se deben de realizar sobre esta, no se incluyen aspectos mas internos de la red como las estaciones que tienen una pertenencia directa a las líneas de la red, para ello se modela la clase línea, para llevar control sobre estas.

Para implementar la solución la dividiremos en una estructura tridimensional, cuyo esquema es el siguiente:



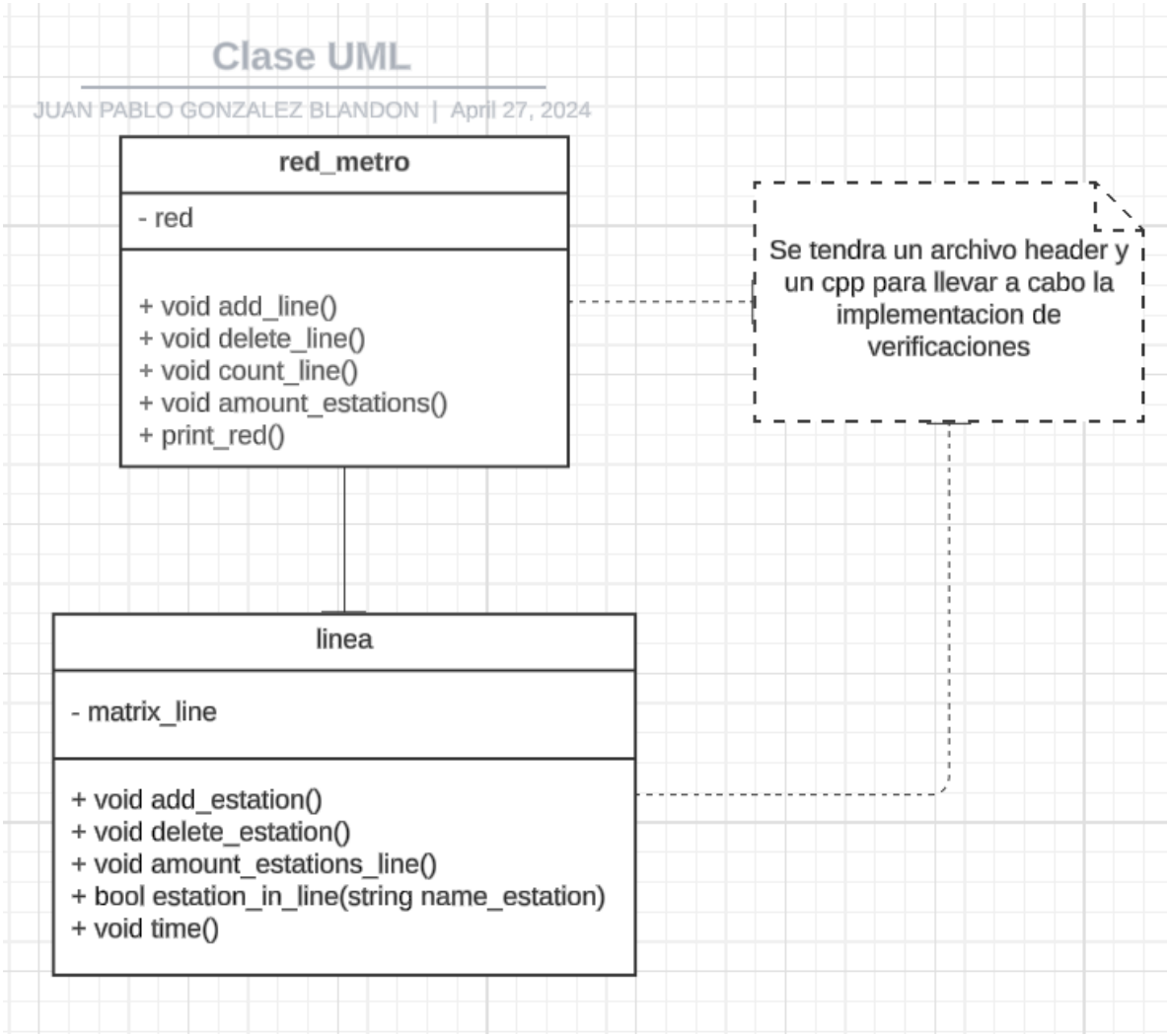
Este será el atributo principal de la clase red encargada del modelamiento de toda de la red.

Ahora bien, para la clase línea se tendrá una estructura bidimensional extraída de la estructura tridimensional la cual será encargada del modelamiento de una línea en especifico y realizar los cambios y operaciones que quiera sobre esta, su estructura será la siguiente:



Este será el atributo principal de la clase línea en la cual tendremos almacenado los tiempos de llegada de una estación a otra, ya que no tenemos la posibilidad de usar mapas, las posiciones de estas matrices serán correspondientes a los tiempos de cada una de las líneas, es decir, si en la primera matriz en la primera posición esta la estación 1 de la línea 1, en la otra estarán los tiempos de llegada de la estación 1 a la siguiente y para conocer el tiempo a la previa simplemente será devolvernos una posición en la matriz.

Diagrama de clases



Algoritmos implementados

- add_line() – add_estation()

  En este método se implementará un algoritmo el cual se encargará de preguntar por consola el nombre la estación a agregar y si es de transferencia o no (el nombre de la línea

será asignado por nosotros) se verificará que la estación a agregar no este incluida en las demás líneas. Se accederá al arreglo tridimensional “red” o bidimensional “matrix_line” según sea el caso y a través del triple-doble puntero podremos manipularla agregando de esta forma la línea al final del arreglo. Para agregar la línea se le preguntara al usuario las estaciones y los tiempos de llegada entre ellas. (Se esta pensando en la posibilidad de agregar una opción para generar los tiempos de forma aleatoria).

- delete_line() – delete_estation()

  En este método se implementará un algoritmo el cual se encargará de preguntar por consola el nombre la estación a eliminar y si es de transferencia o no (el nombre de la línea será asignado por nosotros) se verificará que la estación a eliminar exista. Se accederá al arreglo tridimensional “red” o bidimensional “matrix_line” según sea el caso y a través del triple-doble puntero podremos manipularla eliminando de esta forma la línea o estación deseada.

- amount_estations() - count_line() - amount_estations_line()

  De forma general lo que se realizara en estos tres métodos es ingresar al arreglo correspondiente y contar las posiciones de la capa del arreglo correspondiente e imprimir por pantalla la cantidad.

- print_red()

  Para una mejor interacción con el usuario durante la ejecución del programa siempre estará a su disposición poder visualizar la red, al igual que los anteriores algoritmos se accederá al arreglo tridimensional y se imprimirán las posiciones.

- estation_in_line

  Se preguntar al usuario la estación que quiere verificar y la línea, posteriormente se acceder al arreglo bidimensional para verificar que la estación se encuentre en este mismo este proceso se llevar a cabo recorriendo el arreglo, retornara un booleano correspondiente al estado de la búsqueda.

- time()

  Se investigará como se puede tomar la hora actual, se calculara el tiempo de llegada sumando los tiempos que tenemos registrados en el arreglo y por último se le sumara a la hora actual para imprimir por pantalla, hora actual, hora de llegada y tiempo que tardo en llegar.

Problemas de desarrollo que afronto – consideraciones a tener en cuenta en la solución

El primer problema evidenciado es el gestionamiento de los arreglos en el caso en el que se desee, eliminar o agregar un elemento que no se encuentre en los extremos del arreglo, se ha pensando en ciertas soluciones, pero debemos implementarlas y estarán sujetas a mejoras.

Hubo una confusión la hora de definir las clases, pero con un estudio de la problemática un razonamiento sobre esta se pudo solventar de manera satisfactorio.

Se tuvieron ciertos problemas a la hora de definir los constructores.

Se presento un error con el manejo de las dimensiones de los arreglos ya que una línea no necesariamente tiene la misma cantidad de estaciones que las demás, sin embargo, se llegó a una solución a este problema.

Avance

El avance de la solución se puede evidenciar en el repositorio con los commits diarios realizados.