**Problema**

En nuestro sistema de metro, cada estación tiene un nombre único y un valor de tiempo que indica la duración en segundos para llegar a la siguiente o anterior estación en la misma línea.

También hay estaciones de trasferencia, que son puntos de intersección entre dos líneas, esta estación de trasferencia no podrá ser eliminada sin tiene conexiones con dos estaciones entre ellas.

Tenemos tres clases de objetos: estación, línea y red, con la siguiente jerarquía:

Una línea pertenece a una única red.

Una estación aparece solo una vez en una línea.

Una línea está presente solo una vez en una red.

Si una red tiene múltiples líneas, estas no pueden estar desconectadas.

Funcionalidades Requeridas

El programa debe incluir constructores, destructores, getters y setters, y ser capaz de:

A. Agregar una estación a una línea, ya sea en los extremos o en posiciones intermedias.

B. Eliminar una estación de una línea, con la excepción de las estaciones de trasbordo.

C. Determinar cuántas líneas tiene una red de metro.

D. Contar el número de estaciones en una línea específica.

E. Verificar si una estación dada pertenece a una línea específica.

F. Agregar una línea a la red de metro.

G. Eliminar una línea de la red de metro, solo si no tiene estaciones de trasbordo.

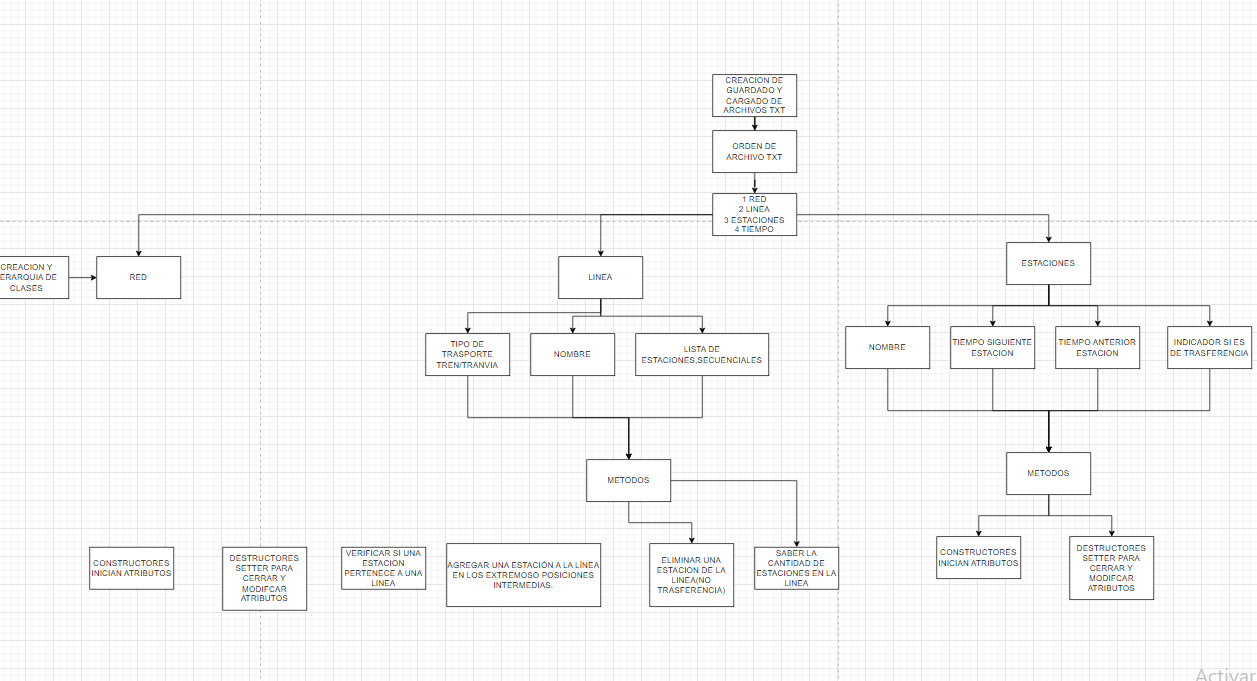
H. Calcular cuántas estaciones tiene una red de metro, con precaución respecto a las estaciones de trasbordo.

Además, se requiere un subprograma que simule el funcionamiento de la red, denominado "Cálculo del tiempo de llegada". Este programa toma la hora actual como el tiempo de salida de un tren desde una estación de la red y predice el tiempo que tardará ese tren en llegar a otra estación dentro de la misma línea. Esta simulación solo funciona para estaciones dentro de la misma línea.

**Diseño de Clases y Jerarquía:**

**ESTRUCTURA**

**Bitácora**: empezamos planeando en un diagrama de flujo la estructura de las clases



**PLANTEAMIENTO FINAL:**

**Clase Estación:**

Cada estación puede tener los siguientes atributos:

Nombre

Tiempo hacia la estación siguiente

Tiempo hacia la estación anterior

Indicador de si es una estación de transferencia

Métodos:

Constructores para inicializar los atributos.

Getters y setters para acceder y modificar los atributos.

**Clase Línea:**

Cada línea puede tener los siguientes atributos:

Nombre

Tipo de transporte (tren o tranvía)

Lista de estaciones en orden secuencial

Métodos:

Agregar una estación a la línea en los extremos o posiciones intermedias.

Eliminar una estación de la línea (excepto estaciones de transferencia).

Obtener la cantidad de estaciones en la línea.

Verificar si una estación dada pertenece a la línea.

Constructor para inicializar los atributos.

Destructor para limpiar la memoria si es necesario.

**Clase RedMetro:**

La red de metro puede tener los siguientes atributos:

Lista de líneas que forman parte de la red.

Métodos:

Agregar una línea a la red de metro.

Eliminar una línea de la red de metro (si no tiene estaciones de transferencia).

Obtener la cantidad de líneas en la red de metro.

Obtener la cantidad total de estaciones en la red de metro (considerando estaciones de transferencia).

Constructor para inicializar la red de metro.

Destructor para limpiar la memoria si es necesario.

Subprograma de Simulación ("Cálculo del Tiempo de Llegada"):

Este subprograma puede ser una función o método adicional que tome la hora actual como entrada, junto con la estación de salida y la estación de llegada.

Utiliza los tiempos de viaje entre las estaciones para calcular el tiempo estimado de llegada.

Esta función solo realiza cálculos entre estaciones de la misma línea.

ACTUALIZACION 27/04/2024

Se empezó a desarrollar la clase estación, teniendo en mente que no se pueden usar contenedores, esto ha generado una gran problemática al tener varios intentos fallidos de cargado de información, lo cual no esta generado un reto para poder implementar el guardado de información, ya que sin esto es complicado avanzar en la otra implementación del código.

**Problemas de implementación:**

1. Se tiene un primer problema al crear la función de implementar nuevas funciones estaciones a una línea; prototipo:   
    *string nombreLinea, nombreEstacion;*

*int tiempoSiguiente, tiempoAnterior;*

*bool esTransferencia;*

*cout << "Ingrese el nombre de la linea: ";*

*getline(cin, nombreLinea);*

*cout << "Ingrese el nombre de la estacion: ";*

*getline(cin, nombreEstacion);*

*cout << "Ingrese el tiempo a la estacion siguiente (segundos): ";*

*cin >> tiempoSiguiente;*

*cout << "Ingrese el tiempo a la estacion anterior (segundos): ";*

*cin >> tiempoAnterior;*

*cout << "¿Es estacion de transferencia? (0: No, 1: Si): ";*

*cin >> esTransferencia;*

*Linea\* linea = new Linea(nombreLinea, "tren");*

*Estacion\* estacion = new Estacion(nombreEstacion, tiempoSiguiente, tiempoAnterior, esTransferencia);*

*linea->agregarEstacion(estacion);*

*red.agregarLinea(linea);*

*cout << "Estacion agregada correctamente.\n";*

*break;*

*}*

1. Hay un segundo problema al crear la función 7 *(Eliminar una línea de la red Metro (sólo puede eliminarse si no posee estaciones de transferencia)),* el problema es que no se implementó que solo se puedan eliminar líneas que no tengan transferencias:   
   *void RedMetro::eliminarLinea(string nombreLinea) {*

*auto it = lineas.begin();*

*while (it != lineas.end()) {*

*if ((\*it)->getNombre() == nombreLinea) {*

*delete \*it; // Liberar la memoria de la línea eliminada*

*it = lineas.erase(it); // Actualizar el iterador después de la eliminación*

*cout << "Linea eliminada de la red Metro.\n";*

*return;*

*} else {*

*++it;*

*}*

*}*

*cout << "La linea no existe en la red Metro.\n";*

*}*