****

**Universidad De Antioquia**

**Facultad de ingeniería**

**Materia:**

**Informática II**

**Trabajo presentado por:**

**Carlos Manuel Gutiérrez Peñafiel**

**Fabio Andrés García Pérez**

**Presentado a:**

**Augusto Enrique Salazar Jimenez**

**Medellín, Colombia**

**Análisis del problema:**

Para el ejercicio planteado propone hacer una interfaz en la que un usuario pueda crear una red metro con distintas opciones como en el siguiente menú:

**A.** Agregar una estación a una línea.

**B.** Eliminar una estación de una línea.

**C.** Saber cuántas líneas tiene una red Metro.

**D.** Saber cuántas estaciones tiene una línea dada.

**E.** Saber si una estación dada pertenece a una línea específica.

**F.** Agregar una línea a la red Metro.

**G.** Eliminar una línea de la red Metro.

**H.** Saber cuántas estaciones tiene una red Metro.

para la solución pensada, se planea en una primera instancia cuando no hay ninguna línea ni estaciones en nuestra red metro hacer lo siguiente:

1. pedir al usuario que ingrese el nombre para la primera línea, está esta línea será considerada como un objeto con los siguientes atributos:

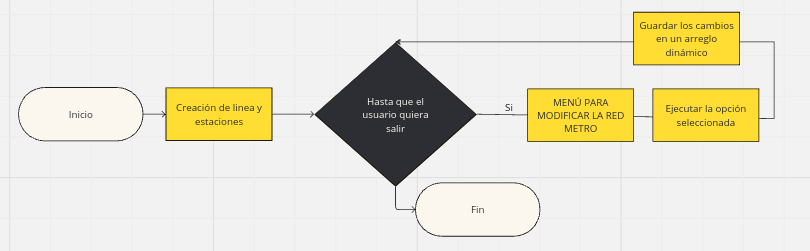
* Nombre de la línea.
* Estaciones.

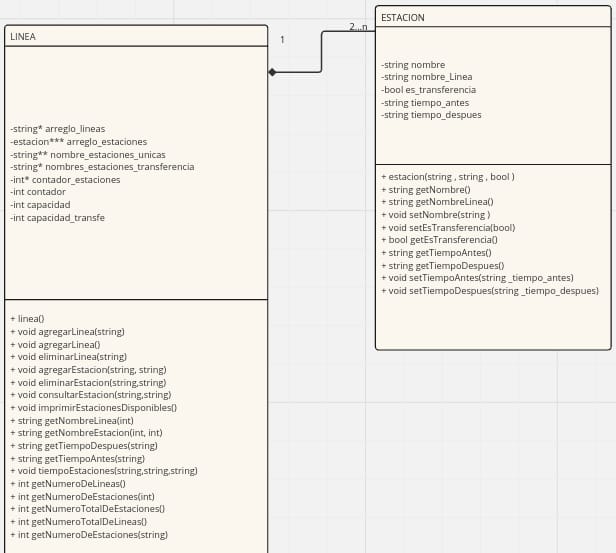
1. Se pedirá al usuario que ingrese las estaciones para que la línea se pueda crear correctamente. cada estación será un objeto con los siguientes atributos:

* Nombre de la línea a la que pertenece.
* Nombre de la estación.
* Tiempo de llegada hasta la estación anterior.
* Tiempo de llegada hasta la estación siguiente.
* Nombre de la estación anterior.
* Nombre de la siguiente estación.
* Entre qué estaciones desea crear la nueva estación (Solo si ya ha creado un mínimo de 2 estaciones en la línea principal).



Las estaciones se manipularán a través de un arreglo dinámico de objetos. Una vez que se tenga mínimo una línea con 2 estaciones, se entrará en un ciclo mostrando las opciones que tiene el usuario para manipular la red metro, este ciclo terminará hasta que el usuario ya no desee seguir modificando la red metro.



**Diagrama de clases:**

**Algoritmos Implementados:**

En la clase Estación se usaron los siguientes métodos:

|  |
| --- |
| 1.**String getNombre ():** Método que retorna el nombre de la estación en una variable tipo string.  2.**String getNombreLinea ():** método que retorna en una variable tipo string el nombre de la línea a la que pertenece la estación.  3.**Void setNombre (string):** método set para asignar el nombre de la estación.  4.**Void setEsTransferencia(bool):** método que tiene como parámetro un bool y modifica el valor del atributo de la estación “es\_trasferencia”, asignándole true o false dependiendo del caso.  5**.Bool getEstransferencia ():** Método get para determinar si una estación es de transferencia o no.  6.**String getTiempoAntes ():** método get que retorna un string con el tiempo que se demora en ir de una estación a otra que esta antes.  7.**String getTiempoDespues ():** método get que retornar un string con el tiempo que se demora en ir de una estación a otra que esta después.  8.**Void setTiempoAntes (string):** método set que tiene como parámetro un string que sería el nuevo tiempo que se tarda en ir de una estación a otra que se encuentra antes.  9.**Void setTiempoDespues (string):** método set con nuevo tiempo como parámetro. Modifica el tiempo que se demora en ir de una estación a la siguiente. |

En la clase Línea se tienen los siguientes métodos:

|  |
| --- |
| 1.**Void agregarLinea (String):** método que tiene como parámetro el nombre de la línea que se desea crear, verificando que no se encuentre otra línea con el mismo nombre ya creada anteriormente.  2.**Void agregarLinea ():** método usado para crear mínimo una línea y dos estaciones.  3.**Void eliminar Línea(string):** método que tiene como parámetro el nombre de la línea que se desea eliminar, verificando que no tenga una estación de transferencia.  4.**Void agregarEstacion (string, string):** método que tiene como parámetro el nombre de la estación y el nombre de la línea a la que se va a agregar, verificando que no exista otra estación con el mismo nombre.  5.**Void eliminarEstacion (string, string):** método que tiene como parámetros el nombre de la estación que se desea eliminar y en qué línea se encuentra, verificando que la estación exista.  6.**Bool consultarEstacion (string, string**): método que tiene como parámetros el nombre de una estación y el nombre de una línea para verificar si la estación se encuentra en la línea, retornando 1 si se encuentra en la línea y 0 si no se encuentra.  7.**Void imprimirEstacionesDisponibles** (): método que imprime por pantalla todas las líneas creadas con sus respectivas estaciones.  8.**String getNombreLinea (int):** Método get que tiene como parámetro un entero que sirve para retornar el nombre de una línea de un arreglo en la posición del entero que esta como parámetro.  9.**String getNombreEstacion (int, int):** método get que tiene como parámetros dos enteros para acceder a los atributos de una estación a través del triple puntero.  10.**Void tiempoEstaciones (string, string, string):** método que tiene como parámetros el nombre de la línea, nombre de la estación inicial y final, e imprime por pantalla el tiempo que demora en llegar a la estación de destino.  11.**Int getNumeroDeLineas ():** método get que retorna contador con el valor del número de líneas creadas.  12.I**nt getNumeroDeEstaciones(int**): método get que tiene como parámetro un entero para retornar el número de estaciones ubicadas en un arreglo en la posición del parámetro.  13.**Int getNumeroTotalDeEstaciones ():** método get que retornar el número total de estaciones creadas en la red metro.  14.**Int getNumeroTotalDeLineas ():** método get que retorna la cantidad de líneas creadas en la red metro.  15.**Int getNumeroDeEstaciones(string):** método get que tiene como parámetro el nombre de una línea y retorna la cantidad de estaciones creadas en esa línea. |

**Problemas durante el desarrollo del programa:**

* Verificación para estación ya creada: se tenía previsto crear un arreglo que contuviera los nombres de todas las estaciones de metro para cuando crearan una nueva estación verificar que no existiera previamente. Sin embargo, en el arreglo se estaba guardando el nombre de la estación de transferencia junto con un “- “y el nombre de la línea, por ende, podían crear una nueva estación con el nombre de la estación de transferencia. Este problema se solucionó guardando los nombres de las estaciones sin la línea, aunque fueran de transferencia.
* Actualizar tiempo luego de eliminar estaciones: cuando se eliminaba alguna estación ya sea que este entre dos estaciones o al final de la línea, se tenia que actualizar los tiempos de las estaciones que estuvieran antes y después. Sin embargo, nos daba error al tratar de cambiar los tiempos, esto se debía a que no estábamos actualizando el arreglo con nuestros objetos y se trataba se cambiar el tiempo en una posición donde no había nada porque se acababa de eliminar la estación en ese lugar.

Para el caso de eliminar una estación entre dos estaciones se hicieron los siguientes pasos.

* Primero se debía localizar la estación que se iba a eliminar.

|  |  |  |
| --- | --- | --- |
| estacionA | EstacionB | EstacionC |

* Se elimina la estación.

|  |  |  |
| --- | --- | --- |
| estacionA |  | EstacionC |

* Ahora se acomodará el arreglo y preguntará el nuevo tiempo de una estación a otra.

|  |  |
| --- | --- |
| estacionA tiempo= 5 | Tiempo = 5 estacionC |

Para los casos de eliminar una estación en los extremos, se debe actualizar la dimensión del arreglo y cambiar el tiempo antes si se elimina la primera estación o cambiar el tiempo a la estación después si se elimina la última estación

**Evolución de la solución:**

Se decidió cambiar la forma en la que se almacenaban los datos, cambiando los atributos de la clase línea y agregándole diferentes punteros para crear arreglos dinámicos en los cuales se almacenaran los objetos de clase estación, nombres de líneas, nombres de estaciones de transferencia.

En la clase linea se tiene los siguientes atributos para almacenar los datos:

-String\* arreglo líneas: es un puntero a un arreglo de strings en donde estarán los nombres de todas las líneas.

-Estacion\*\*\* arreglo estaciones: es un puntero a un arreglo de punteros a punteros de objetos estaciones.

-String\* nombres\_estaciones\_transferencia: es un puntero a arreglo de strings con los nombres de las estaciones de transferencia.

-Int\* contador\_estaciones: es un puntero a un arreglo de enteros, con la cantidad de estaciones que hay por línea.

En la clase estacion se cambiaron los atributos para guardar los datos de cada estacion de la siguiente manera:

-String nombre

-String nombre\_linea

-Bool es\_transferencia

-String tiempo\_antes

-String tiempo\_despues