****

**Planteo del Problema**

Desarrollar un programa en el cual se simula el funcionamiento del sistema metro y las partes esenciales de su funcionamiento como lo son la red metro, las líneas y las estaciones que componen a esta, adicionalmente se debe tener en cuenta los tipos de estaciones ya que existen estaciones de tránsito regular y estaciones que funcionan también como estaciones de transferencia entre líneas, estas deben de conectar 2 diferentes líneas, se debe desarrollar un programa que calcule el tiempo de tránsito entre dos estaciones X & Y. Este programa estará compuesto por una Clase Padre llamada Red Metro, la clase líneas es derivada de está y la clase estación es derivada de la clase línea, también debemos crear un método que nos permita agregar y modificar estaciones.

**Funcionamiento de la “Red Metro”**

Esta es la clase principal que debe estar compuesta por la subclase Líneas, esta debe llevar los siguientes atributos; nombre de la red, número de líneas y obviamente las líneas. Los métodos de Eliminar y Agregar Líneas.

**Funcionamiento de las “Líneas”**

Cada línea debe de estar compuesta por 2 o más estaciones, puesto que de lo contrario no serían líneas, en las líneas podemos movernos de manera bidireccional y es sabido que el tiempo que nos demoramos yendo de norte a sur, debe ser el mismo que de sur a norte, puesto que no tenemos interferencia alguna en la vía, la clase línea está compuesta por atributos los cuales son; un identificador tipo string, y también por un número de estaciones que debe estar definida en esta clase. Además esta debe estar compuesta por métodos de eliminar y agregar estaciones.

**Funcionamiento de “Estación”**

Esta es una subclase de la clase líneas, que está compuesta por los siguientes atributos; nombre de la estación, tipo de estación (Transito,transferencia), tenga en cuenta que si es una estación de transferencia, esta debe de llevar el identificador de una de las líneas de la red, necesitamos un puntero que apunta a la estación siguiente y la estación anterior.

**Definición de tipo de acceso a los atributos y métodos de las clases**

**Clase RedMetro**(Clase padre)

**Atributos:**

* **nombreRed**(string) **privado**: El nombre de la red debe ser privado para encapsular y evitar su modificación directa desde fuera de la clase.
* **numeroLineas**(int) **privado**: El número de líneas también debe ser privado para controlar su acceso y modificación.
* **lineas**(puntero a línea) **privado**: El arreglo de líneas debe ser privado para controlar su modificación y acceso desde fuera de la clase.

**Métodos:**

* **agregarLinea**(Linea linea) **público**: Se hace público para permitir agregar nuevas líneas a la red desde fuera de la clase.
* **eliminarLinea**(string idLinea) **público**: También es público para permitir eliminar líneas de la red desde fuera de la clase.

**Clase Linea**(Clase derivada de RedMetro)

**Atributos:**

* **idLinea**(string) **privado**: El identificador de la línea debe ser privado para encapsular y controlar su acceso y modificación.
* **numeroEstaciones**(int) **privado**: El número de estaciones también debe ser privado para controlar su acceso y modificación.
* **estaciones**(puntero a estación) **privado:** El arreglo de estaciones debe ser privado para controlar su acceso y modificación desde fuera de la clase.

**Métodos:**

* **agregarEstacion**(Estacion estacion) **público**: Se hace público para permitir agregar nuevas estaciones a la línea desde fuera de la clase.
* **eliminarEstacion**(string nombreEstacion) **público:** También es público para permitir eliminar estaciones de la línea desde fuera de la clase.

**Clase Estacion**(Clase derivada de línea)

**Atributos:**

* **nombreEstacion** (string) **privado:** El nombre de la estación debe ser privado para encapsular y controlar su acceso y modificación.
* **tipoEstacion** (string) **privado:** El tipo de estación debe ser privado para encapsular y controlar su acceso y modificación.
* **idLineaTransferencia** (string) **privado:** El identificador de la línea de transferencia debe ser privado para encapsular y controlar su acceso y modificación.
* **siguiente** (puntero a estación) **protegido:** El puntero a la estación siguiente se declara como protegido para permitir su acceso desde las clases derivadas (en este caso, no se necesitan clases derivadas, pero si se necesitaran en el futuro, este puntero estaría disponible para ellas).
* **anterior** (puntero a estación) **protegido:** Similar al anterior, se declara como protegido para permitir su acceso desde las clases derivadas.

Al definir los atributos como `privados`, garantizamos el principio de encapsulamiento, lo que nos permite controlar el acceso a los datos y métodos de la clase. Al definir los métodos como `públicos`, proporcionamos una interfaz pública para que los usuarios de la clase puedan interactuar con ella de manera segura y controlada.