## Normas de presentación

El trabajo práctico se realizará en forma grupal.

El grupo constará de 3 a 4 integrantes el cual quedará conformado la primera semana de clase.

Se realizará un proyecto en entorno eclipse, el cual contendrá los siguientes paquetes:

***Api***: contiene la definición de los TDA

***Api\_es***: contiene los métodos para poder cargar TDA desde teclado, archivo o con datos al azar. Además los métodos para mostrar los datos de los TDA por pantalla.

***Aplicaciones:*** contiene las distintas aplicaciones requeridas.

***Implementaciones***, con sub-paquetes: ***Estatica*** y ***Dinamica:*** contiene las implementaciones estática y dinámica de los distintos TDA’s.

***Metodos:*** contiene los distintos métodos usados por las distintas aplicaciones.

## Fechas de entrega

La fecha final de entrega de los puntos 1 y 2 corresponde a la 1era fecha de entrega TPO publicada en el cronograma.

La fecha de entrega final del punto 3 corresponde a la fecha final de entrega TPO publicada en el cronograma.

La entrega será subida a la opción ***Grupos*** de Webcampus, una vez aprobada la defensa oral.

El trabajo será desaprobado, si no se tiene la aprobación de la entrega en tiempo y forma.

## Documentación

Todos los archivos del proyecto deberá tener la siguiente documentación como encabezado:

/\*\*

@Autores:

@Grupo:

\*\*/

**Especificación de TDA (Incluir en paquete api del proyecto)**

Especificar los siguientes TDA, estableciendo tarea y precondición de cada operación asociada.

***Datos adicionales en el encabezado del archivo***

@TDA:

@Definición:

***Formato de la información en el encabezado de cada método***

/\*\*

@Tarea:

@Parámetros:

@Devuelve:

@Precondición:

@Postcondición:

\*\*/

**Implementación de TDA (Incluir en paquete: implementación, sub-paquetes: arreglos y/o listas según corresponda)**

Implementar los TDA especificados en el punto anterior. Las implementaciones estáticas se colocarán en un paquete del proyecto y dinámicas en otro paquete.

***Datos adicionales en el encabezado del archivo***

@Costo de cada uno de los métodos:

***Métodos externos***

Los métodos utilizados por la aplicación se deberá especificar: tarea, parámetros, que devuelve, precondición y postcondición. Con el siguiente encabezado:

***Formato de la información en el encabezado de cada método***

/\*\*

@Tarea:

@Parámetros:

@Devuelve:

@Precondición:

@Postcondición:

@Costo:

\*\*/

## Enunciado

Una organización dedicada a turismo desea desarrollar una aplicación para obtener información sobre recorridos de las distintas líneas de subte y trenes del área metropolitana.

Para ello dispone de archivos csv para cada una de las líneas. La información en dicho archivo son: sentido de la estación, estación, y línea/estación de transferencia de dicha estación. La cantidad de archivos e información puede variar.

* Subte A.csv
* Subte B.csv
* Subte C.csv
* Subte D.csv
* Subte E.csv
* Subte H.csv
* Premetro.csv
* FCGSM
* FCDFS
* FCGBM - R.Tigre
* FCGBM - R.JLSuarez
* FCGBM - R.Mitre
* FCGU

Ejemplo de formato de los archivos:

A;Plaza de Mayo;;

A;Perú;D;Catedral

A;Perú;E;Bolivar

A;Piedras;;

A;Lima;C;Avenida de Mayo

A;Sáenz Peña;;

A;Congreso;;

V;Pasco;;

I;Alberti;;

A;Plaza Miserere;FCDFS;Once

A;Plaza Miserere;H;Once

Campos:

1ero: sentido en que paran las formaciones (a: ambos; I: sólo de ida; V: sólo de vuelta)

2do: nombre de la estación

3ro: línea de subte/tren de transferencia

4to: nombre de la estación de transferencia

Existen tantas líneas para una estación como líneas a la cual se puede transferir.

Se pide:

1. Cargar las estaciones de cada línea en un diccionario múltiple. Donde la clave corresponde a la línea y los valores a los nombres de estaciones para esa línea.
2. Informar:
   1. Cantidad de estaciones para cada una de las líneas
   2. Porcentaje de estaciones que tienen estaciones de transferencia por cada línea
   3. Lista de estaciones de igual nombre, indicando a que línea de subte o tren pertenece
   4. Lista de estaciones de transferencia en orden alfabético (nombre, línea a la cual pertenece) indicando cantidad de líneas posibles se puede combinar.
3. Como mejora de la aplicación se requiere:
   1. Generar un grafo de estaciones – línea. El peso de la arista entre dos estaciones corresponderá la distancia en km entre las dos estaciones.
   2. Cuál es la menor/mayor distancia entre dos estaciones, indicando distancia y entre que estaciones ocurre.
   3. Dada dos estaciones indicar si hay un camino entre ellas. Si lo hay indicar las estaciones que debe recorrer para llegar de una a la otra y distancia total a recorrer

### Nota:

Para realizar la aplicación, tomar las interfaces e implementaciones usadas en clase y modificar su tipo de dato a String si fuese necesario.