**Programación por capas**

“La **programación por capas** es un modelo de desarrollo de software en el que el objetivo primordial es la separación (desacoplamiento) de las partes que lo componen”

<https://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas>

Para nuestro caso, cada capa va a estar representada por un paquete. Cada capa (paquete) tiene una o más clases que realiza una tarea determinada. El cambio en la implementación de una capa no tiene que afectar el resto de las capas.

**Paquete modelo**

Clases que representan objetos del mundo real: Ej.: Linea, Estacion, Tramo. Tienen atributos y métodos generales como getters y setters, equals, toString y códigos relacionados a la validación de los objetos creados.

**Paquete de datos**

Clases relacionadas con la gestión de los datos de la aplicación.

En nuestro caso la configuración y carga de datos que necesita la aplicación.

En nuestra implementación utilizamos archivos de texto. Debería ser fácil cambiar la fuente de donde obtenemos los datos (archivos binarios, base de datos, etc.) modificando solo las clases de este paquete sin tocar el resto del proyecto.

Tener en cuenta los problemas que se pueden presentar y lanzar las excepciones correspondientes. Ej.: no se encontró el archivo, el archivo no tiene el formato esperado, etc.

Dividir en métodos (o clases) la carga de datos: parámetros, líneas, estaciones, tramos, etc.

**Paquete lógica o de negocio**

Clases que contiene todo lo relacionado con el proceso de los datos.

En nuestro caso son las clases que reciben los datos, arman el grafo y calcula caminos según las opciones que le pasan. En esta implementación utilizamos los grafos de la biblioteca de TADs de Goodrich, pero se puede cambiar su implementación por otro paquete como por ejemplo JGraphT, hacer los cálculos usando otro tipo de estructuras, etc. Cualquier cambio en este paquete no debería afectar el resto de los paquetes.

**Paquete interfaz con el usuario**

Clases que realizan la iteración con el usuario de la aplicación. Por lo general la suelen llamar GUI (Graphical User Interface)

Todavía no vimos las clases que provee Java para desarrollar interfaces. La aplicación va a estar limitada a clases que manejen un menú de texto. Siguiendo la misma línea que los paquetes anteriores debería ser posible cambiar nuestra interfaz por una que contenga componentes gráficos o se acceda desde la Web, etc. solamente cambiando las clases que componen este paquete.

De todas maneras, aun siendo un menú de texto, facilitar la iteración con el usuario mostrando los códigos de las estaciones, líneas, etc., validar los datos ingresados, permitir cambiar de opciones o hacer otra consulta, etc.

**Paquete de test**

Clases que permiten realizar pruebas en las clases implementadas.

Por ejemplo implementar clases utilizando la biblioteca JUnit para probar los resultados de los métodos de la capa lógica.