**Listas Lnkeadas**

Las listas linkeadas en computación son una estructura de datos muy utilizada cuando se quiere mantener un contenedor de datos (u objetos), y recorrerlo en forma ordenada. Este tipo de datos es muy adecuado cuando queremos hacer inserciones y borrar elementos.

En otros tipos de contenedores como el array, esta tarea es dificultosa, ya que si queremos insertar un elemento en el medio del array, debemos “correr” todos los siguientes un lugar. De la misma forma si queremos borrar uno, debemos realizar un reacomodamiento de todos los elementos.

En el caso de las listas linkeadas, cada elemento se almacena independientemente en la memoria, y junto con él se almacena un puntero al próximo elemento.

Java provee la clase *LinkedList,* y otros contenedores específicos (nosotros hemos visto los principales) para proveer esta funcionalidad y otras más avanzadas, frente a otros tipos de contenedores como el *Vector* o *ArrayList*.

A continuación les mando un caso de uso, donde podemos observar una lista linkeada, pero sin utilizar la clase *LinkedList*. En este caso creamos programa que presenta un comportamiento similar.

**Caso: Viajante**

Se solicita un programa que calcule el costo de un viajante que debe circular por diferentes ciudades de acuerdo al siguiente gráfico.



El gráfico está representado como una **lista linkeada**. Cada nodo contiene un puntero al siguiente. Cada nodo tiene dos datos: nombre de la ciudad, y cuanto cuesta llegar a ella.

Se solicita un programa que dada la anterior lista, calcule el costo de llegar desde Mendoza a Buenos Aires.

**Resolución**:

Primero planteamos la estructura de datos para el programa. Lo principal es algún tipo de datos que permita guardar la ciudad.



Luego, deberíamos establecer algún mecanismo para que una ciudad pueda “apuntar” hacia la siguiente ciudad. Lo más lógico sería agregar un atributo más (un puntero). O una referencia a un objeto (también) del tipo Ciudad. Llamamos a este atributo ***proxima***.



Podemos completar el modelo con otros datos para enriquecerlo. Por ejemplo:



Decimos que una ***Ciudad*** “es un” tipo particular de un ***LugarGeográfico*** (otro tipo particular de lugar geográfico podría ser un país). También decimos que una ***Ciudad*** implementa el comportamiento de la interfaz ***Acciones***, En particular, que toda Ciudad debe implementar un método ***calculaSuperficie()*,** y luego “saber” como calcular su propia superficie.

También implementa (hereda) la interfaz *AccionesLista*. Quien escribe dicha interfaz “decide” el conjunto de métodos que debe tener toda clase que herede de ella. Se especifica **qué es lo que hace falta**, pero **no como se debe realizar** (por eso los métodos son implícitamente abstractos).

Por ahora el único método que vamos a colocar en la interfaz es ***proximo()***, ya que en toda lista encadenada debemos poder “pasar al próximo elemento”. La clase que herede de esta interfaz deberá escribir el código del método ***proximo()***.