PRÁCTICA 03

Utilización de Estructuras de Datos Asociativas en Java

Objetivos

- Estudiar los conjuntos y los mapas para la resolución de diferentes problemas (desde los puntos de vista del almacenamiento de datos y de consulta y manipulación).
- Aprender a utilizar la composición de estructuras de datos asociativas en la resolución de problemas.

Requisitos

Para superar esta práctica se debe realizar lo siguiente:

- Dominar las estructuras de datos asociativas `TreeSet<T>`, `TreeMap<K,V>`, `HashSet<T>` y `HashMap<K,V>`, y la combinación de ellas para la resolución de problemas.
- Conocer cómo realizar consultas sobre estructuras de datos asociativas, así como la mejor forma de plantearlas en problemas sencillos.

Enunciado

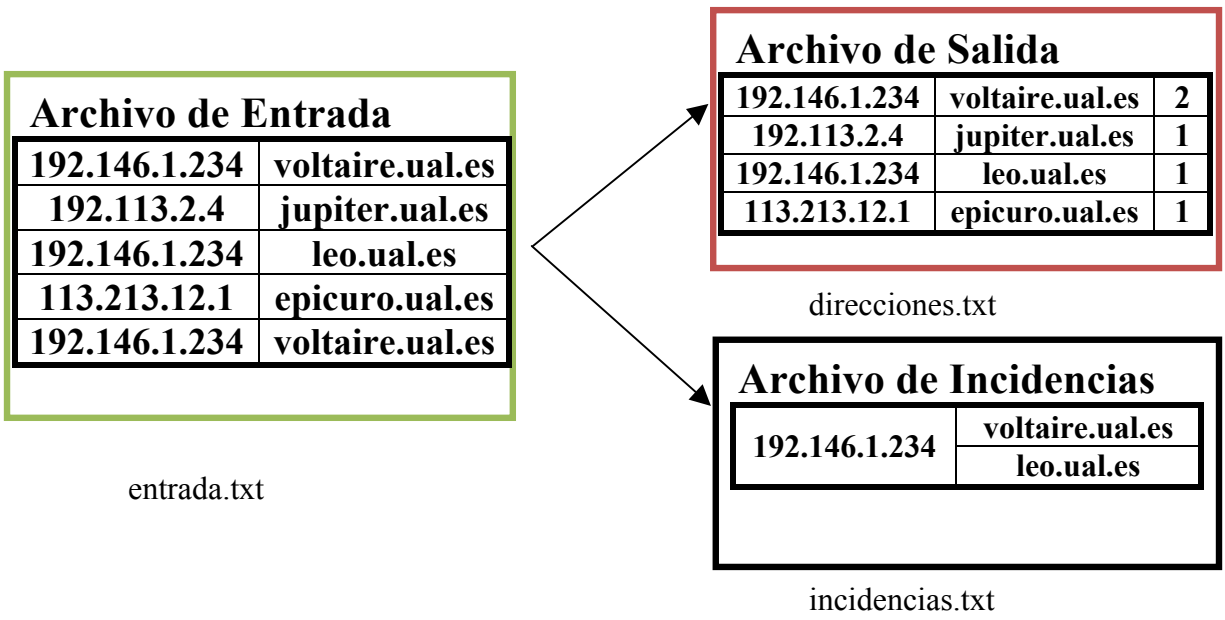
1. **Ejercicio con mapas**. Control de una puerta de acceso a una red.

Tal y como se ha estudiado en la práctica 02, las direcciones IP identifican de forma unívoca todos y cada uno de los dispositivos (ordenadores, impresoras, etc.) conectados en Internet. Cada IP tiene asociado un nombre de máquina; por ejemplo, *filabres.ual.es* tiene asociada la dirección IP *150.214.156.2*. Estas direcciones están formadas por cuatro campos que representan partes específicas de Internet, *red.subred.subred.máquina*, que el ordenador trata como una única dirección IP. Esta dirección consta de 32 bits, aunque normalmente se representa en notación decimal, separando los bits en cuatro grupos de 8 bits cada uno, expresando cada campo como un entero decimal (0..255) y separando los campos por un punto. En el ejemplo anterior: *filabres.ual.es* ⇔ *150.214.156.2*.

Supongamos que se establecen conexiones desde una red a través de una puerta de acceso a otra red, y cada vez que se establece una conexión, la dirección IP del ordenador del usuario se almacena en un archivo junto con el nombre de la máquina. Estas direcciones IP y sus máquinas se pueden recuperar periódicamente para controlar quiénes han utilizado la puerta de acceso y cuántas veces lo han hecho.

Como **EJERCICIO** se pide lo siguiente: implementad un programa en Java que lea desde un archivo estas direcciones IP y los nombres de las máquinas asociadas, almacenando en un **TreeMap(K,V)** de **TreeMap<K,V>'s** de la JCF de Java. Cada una de las entradas del primer **TreeMap(K,V)** guarda la dirección IP (clave), mientras que en el otro **TreeMap<K,V>** (valor), se almacenará el nombre de máquina (clave), así como el número de veces que dicho par (dirección IP, nombre máquina) aparece en el archivo de entrada (valor).

- Según se lee cada dirección IP y nombre de máquina del archivo, el programa debe comprobar si dicha dirección está ya en el mapa. Si lo está, debe comprobar que la máquina está como clave en el segundo mapa, en cuyo caso su contador se incrementa en 1; en caso contrario, se inserta en el mapa. Si no está la dirección, hay que crear el segundo mapa y añadir el par máquina y contador a 1.
- Una vez que todas las direcciones IP y nombres de máquina han sido leídas del archivo, se generará como resultado un archivo de salida donde se indiquen los pares (dirección IP, nombre máquina) y sus contadores.
- Finalmente, el programa deberá comprobar, para cada dirección IP, si ésta ha tenido asociado más de un nombre de máquina, en cuyo caso se generará también un archivo de salida indicando dicha incidencia.



Adicionalmente, resulta interesante que indiquéis (**practica03_ejercicio01.pdf)** cómo tendría que plantearse la resolución del problema utilizando como estructura de datos en su implementación un **TreeMap<String, TreeSet<MaquinaContador>>**, en lugar de un TreeMap<K,V> de TreeMap<K,V> como el se ha utilizado como solución (**TreeMap<String, TreeMap<String, Integer>>)**.

Además, se debe responder (con la implementación del correspondiente método) a las siguientes consultas sencillas (asociadas con los test propuestos):

- Devolver todas las máquinas con contador mayor que 1 y mayor que 2.
- ¿Cuántas máquinas tienen un valor de contador igual a 2, e igual a 3?.
- ¿Cuál es el valor del contador del par (dirección, máquina) = (192.146.1.233, pascal.ual.es) y (dirección, máquina) = (192.146.1.234, voltaire.ual.es)?.
- Devolver las incidencias que ha registrado la dirección 192.146.1.233 y las que ha registrado la dirección 192.146.1.234 a nivel de máquinas?.
- ¿Cuántas incidencias han registrado las direcciones 113.213.12.1 y 192.146.1.234?.

2. Ejercicio con colecciones asociativas un poco más complejas. Ciudades en las que determinadas empresas de software desarrollan sus proyectos.

Como ya hemos estudiado en las prácticas 01 y 02, disponemos de información para gestionarla en una estructura de datos. Dicha información está relacionada con las empresas de software que desarrollan proyectos en determinados lugares donde tienen las sedes de dichos proyectos. Para este ejercicio tendremos un el archivo de entrada "**masNuevasEmpresasProyectosCiudades.txt**".

Como **EJERCICIO** se pide lo siguiente: implementad un programa en Java que lea desde un archivo de entrada la anterior lista de **Empresa_Software Proyecto_Software Ciudad_Donde_Se_Desarrolla**, almacenando los datos en estructura de datos tipo Tabla. Según se lee cada línea del archivo (**Empresa_Software Proyecto_Software Ciudad_Donde_Se_Desarrolla**), el programa debe comprobar si dicha tripleta está ya en el contenedor. Si lo está, no hacer nada pues será una línea repetida; en caso contrario, se inserta en la estructura de datos. Una vez que todas las líneas (tripletas) han sido leídas del archivo de entrada y almacenadas en memoria en estructura de datos asociativas **TreeMap<K,V>** y **TreeSet<T>**, se generará como resultado un listado que debe coincidir con el que se proporciona en el test.

Como sugerencia para la realización del ejercicio, podemos plantear una estructura de datos basada en **TreeMap<K,V>** y **TreeSet<T>**. De forma genérica podría plantearse la siguiente estructura:

TreeMap(Empresas, **TreeMap**(Proyectos, **TreeSet**(Ciudades)))

Además, una vez organizados los datos en la ED indicada anteriormente, se debe responder (con la implementación del método correspondiente) a las siguientes consultas:

- Devolver todas las empresas, todos los proyectos y todas las ciudades en tres funciones independientes.
- Devolver las empresas que tienen su sede en la ciudad Miami.
- Devolver los *proyectos* con sede en Washington.
- ¿En cuántas ciudades diferentes se desarrollan proyectos de Google?.
- Devolver cuál es el proyecto con mayor número de sedes (ciudades).
- Devolver cuál es la empresa con mayor número de proyectos.
- Devolver cuál es la ciudad con mayor número de proyectos

Para comprobar la correcta realización de este ejercicio se proporcionará el **test** que se deberá pasar correctamente.