Problema JAVA (I/O.- Adaptación de datos a requerimientos de base de datos)

Tenemos que aportar datos nuevos que recibimos en dos ficheros de texto (***DB1.txt*** y ***DB2.txt***) a una base de datos de bibliografía. En estos ficheros tenemos una entrada por cada línea con los siguientes campos separados por el carácter punto y coma (este carácter no podrá aparecer en ningún otro momento):

DB1.txt

- Un número de referencia (sin orden preestablecido pero único para el conjunto de registros)
- una lista con uno o más autores con su filiación separados por comas. La filiación se presenta entre comillas.
- el título del trabajo
- el año de publicación
- la publicación en que aparece
- una lista de referencias "internas" (es decir, referencias a artículos del mismo conjunto de ficheros) consistentes en los números de referencia correspondientes separados por espacios.

DB2.txt

- Un número de referencia a un registro del fichero ***DB1.txt***
- una dirección web (url)

Ejemplos: (es una sola línea en cada caso)

```
DB1.txt
142; J. Gray "University of California Irvine, uci.edu, USA"; Queues are
databases; 2007; arxiv.org; 248 5 9 10 14 471 325 329 340 464 31
```

```
DB1.txt
142; http://arxiv.org/abs/cs/0701158
```

(para cada registro de ***DB1.txt*** puede haber 0 ó 1 URLs, en consecuencia ***DB2.txt*** puede tener menos entradas que ***DB1.txt***)

La tarea consiste en recoger toda la información que se nos aporta y generar los listados (en otros ficheros de salida) correspondientes a cada una de las tablas que serán necesarias para actualizar posteriormente la base de datos.

Las tablas a generar son:

- ***Filiaciones.txt***: dos columnas separadas por ",". La primera un número de orden dado por el programa y la segunda una institución sacada del campo 2 de ***DB1.txt***. Debe tenerse en cuenta que una línea de ***DB1*** puede tener más de una filiación, y por otro lado las filiaciones se repiten numerosas veces en ***DB1.txt*** pero no deben aparecer repeticiones en Filiaciones.txt.
- ***Autores.txt***: dos columnas separadas por ",". La primera un número de orden dado por el programa y la segunda un autor sacado del campo 2 de ***DB1.txt***. Debe tenerse en cuenta que una línea de ***DB1*** puede tener más de un autor, y por otro lado los autores se repiten numerosas veces en ***DB1.txt*** pero no deben aparecer repeticiones en ***Autores.txt***.
- ***Autor-filiacion.txt***: dos columnas separadas por "," La primera un número de orden asociado a un autor conforme a ***Autores.txt***, y la segunda con el número de orden asociado a Filiación que le corresponde codificado según ***Filiaciones.txt***. Obsérvese que en algún caso es posible que un mismo autor aparezca con distintas filiaciones en ***DB1.txt***; no resolveremos esto y permitiremos que se pierda información incluyendo sólo una de ellas[1].
- ***Info-articulo.txt***: 5 columnas separadas por ",". La primera un número de orden, la segunda el título, la tercera el año, la cuarta la publicación y la quinta columna la dirección web, que en caso de no estar disponible se consignará "null"

[1] Esto no es admisible en un caso real, pero así simplificamos el ejercicio (no por simplificar el programa java, sino por no complicar la estructura de la base de datos objetivo)

- ***Articulo-autores.txt***: dos columnas separadas por ","; la primera un número de orden asociado a un artículo conforme a ***Info-artículo.txt*** y la segunda un autor conforme a ***Autores.txt***. Debe tenerse en cuenta que un articulo puede tener varios autores, y en tal caso se generará una nueva línea en este fichero por cada autor.
- ***Articulo-referencias.txt***: dos columnas separadas por ","; la primera un número de orden asociado a un artículo conforme a ***Info-artículo.txt*** y la segunda una referencia conforme al sexto campo de ***DB1.txt***. Debe tenerse en cuenta que un articulo tendrá normalmente varias referencias, y en tal caso se generará una nueva línea en este fichero por cada referencia. También hay artículos sin referencias, por lo que ha de ponerse cuidado en que no aparezcan líneas con número de artículo y sin referencia asociada.

A tener en cuenta:

Los ficheros ***DB1.txt*** y ***DB2.txt*** se han generado extrayendo automáticamente la información de Google Scholar y pueden presentar las faltas de homogeneidad propias de dicha aplicación web (p. ej. Un mismo autor con variantes en su denominación que le harán aparecer como más de uno). Al examinar los resultados esto puede inducir en algún caso a pensar que el programa no ha funcionado correctamente, por lo que conviene "sospechar" también de los datos de origen. En todo caso lo que se pretende es que la salida sea consistente con la entrada[i].

Problema OPCIONAL JAVA (acceso a la base de datos)

Para quien acepte el "reto": el programa anterior puede modificarse de un modo muy simple para que en vez de generar las tablas en ficheros en disco, vayan a aparar directamente a la base de datos. Sólo hay que conectar como vimos el último día de clase (ver última hoja proyectada)

```
import java.sql.*
try {
Class.forName("com.mysql.jdbc.Driver").newInstance();
Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/DataBase","usuario", "clave");
//Continúa la aplicación
}
catch(SQLException ex) {
        System.err.println("Problema de conexión con la base de datos: "+ex.getMessage());
}
catch(ClassNotFoundException ex) {
        System.err.println("No se encuentra el driver JDBC: "+ex.getMessage());
}
catch(InstantiationException ex) {
        System.err.println("No puede instanciarse el driver JDBC: "+ex.getMessage());
}
catch(IllegalAccessException ex) {
        System.err.println("Intento de acceso ilegal al driver JDBC: "+ex.getMessage());
}

Y luego…   ResultSet rs = con.createStatement().executeQuery("select * from autores");
```

(Ojo, en este caso, para introducir batos en la base de datos, hay que usar "executeUpdate" en lugar de "executeQuery", ya que no se devuelve un resultado tabla -ResultSet-. COnsultar la documentación de "Statement")

[i] De nuevo tomamos el camino sencillo por tratarse de un ejercicio, pero podría ganarse en "robustez" haciendo cosas como considerar que los separadores pueden no ser únicos o seguir un patrón determinado (p.ej. es igual <J. Gray> que <J. Gray>), no permitir que la capitalización de caracteres influya (p.ej. es igual <J. Gray> que <j. gray>), admitir líneas vacías, etc.