1. Diseñar un pequeño programa que lea el fichero de datos y calcule la valoración media, el número de votos medios considerando la colección All-Capitulos) así como el número de personajes, en media, que tiene los datos en la colección Guiones. Se recomienda utilizar openCSV (o cualquier otra librería para el acceso correcto a los datos). Además los datos deben pasar por un proceso de limpieza y mecanismo para tratar las excepciones.

Definimos dos rutas de directorio carpetaCU y carpetaC que representan las carpetas que contienen los archivos CSV de dos colecciones de datos: "CapitulosUnidos" y "Capitulos."

Inicializamos las variables valoraciones, votos, personajesTotal, totalCU, y totalC para mantener un seguimiento de las valoraciones, votos, número total de personajes y el número total de archivos procesados.

Creamos objetos File para representar las carpetas y luego obtiene la lista de archivos CSV en cada carpeta que cumplan con la extensión ".csv."

Recorremos la lista de archivos CSV en la carpeta "CapitulosUnidos" (representada por csvUnidos), y para cada archivo, realizamos lo siguiente:

- Abre el archivo CSV y lee sus contenidos.
- Extrae la valoración (valor numérico) y el número de votos de cada registro en el archivo CSV.
- Acumula las valoraciones y votos en las variables valoraciones y votos.
- Incrementa el contador totalCU para realizar un seguimiento del número de archivos procesados.

Recorremos la lista de archivos CSV en la carpeta "Capitulos" (representada por csvCaps), y para cada archivo, realizamos lo siguiente:

- Abre el archivo CSV y lee sus contenidos.
- Extrae el número de personajes de cada registro en el archivo CSV.
- Acumula el número de personajes en la variable personajesTotal.
- Incrementa el contador totalC para realizar un seguimiento del número de archivos procesados.

Calculamos las medias de valoración y votos si se procesaron archivos en la carpeta "CapitulosUnidos." Luego, muestra estos valores en la consola.

Calculamos la media del número de personajes por capítulo si se procesaron archivos en la carpeta "Capitulos." Luego, muestra este valor en la consola.

El programa utiliza la librería openCSV para leer y analizar los archivos CSV y realiza un manejo básico de excepciones para manejar errores en la lectura de archivos

```java
import java.util.Iterator;


public class ej1{

    Run | Debug
    public static void main(String[] args){
        String carpetaCU = "/home/anne/Escritorio/RI/RI/P3/CapitulosUnidos";
        String carpetaC = "/home/anne/Escritorio/RI/RI/P3/Capitulos";

        double valoraciones=0;
        double votos=0;
        int personajesTotal=0, totalCU=0, totalC=0;

        File carpetaCapUnid  = new File(carpetaCU);
        File carpetaCaps  = new File(carpetaC);

        File[] csvUnidos= carpetaCapUnid.listFiles((dir,nombre)->nombre.endsWith(".csv"));
        File[] csvCaps= carpetaCaps.listFiles((dir,nombre)->nombre.endsWith(".csv"));


        if(csvUnidos!=null){
            for(int i=0; i< csvUnidos.length;i++){
                File archivo= csvUnidos[i];
                Reader reader = new FileReader(archivo);
                if(!reader.ready()) {
                    System.out.println("Error en el fichero de datos");
                }

                CSVReader csvReader= new CSVReader(reader);
                String[] nextRecord;
                String[] firstLine;

                firstLine=csvReader.readNext();

                while ((nextRecord = csvReader.readNext()) != null){
                    if(nextRecord.length>=4){
                        double valoracion = Double.parseDouble(nextRecord[3]);
                        int votacion=  Integer.parseInt(nextRecord[4]);

                        valoraciones+=valoracion;
                        votos+=votacion;
                    }
                }
                totalCU++;
            }
        }
        if(csvCaps!=null){
            for(int i=0; i< csvCaps.length;i++){
                File archivo= csvCaps[i];
                Reader reader = new FileReader(archivo);
                if(!reader.ready()) {
                    System.out.println("Error en el fichero de datos");
                }

                CSVReader csvReader= new CSVReader(reader);
                String[] nextRecord;
                String[] firstLine;

                firstLine=csvReader.readNext();

                while ((nextRecord = csvReader.readNext()) != null){
                    if(nextRecord.length>=4){
                        int personajes=Integer.parseInt(nextRecord[3]);
                        personajesTotal+=personajes;
                    }
                }
                totalC++;


            }
        }

        if(totalCU>0){
            double valoracionMedia= valoraciones/totalCU;
            double mediaVotos= votos/totalCU;

            System.out.println("Valoración media de todos los capítulos --> "+ valoracionMedia);
            System.out.println("Media del número de votaciones --> "+ mediaVotos);

        }
        if(totalC>0){
            double mediaPersonajes = personajesTotal/totalC;
            System.out.println("Media del número de personajes por capítulo --> "+ mediaPersonajes);

        }



    }


}
```

2. Identificar para cada campo de los distintos ficheros qué analizador utiliza- rías, y en caso de necesitar uno específico implementarlo. Recordad que el tipo de analizador dependerá de cómo vayáis a realizar las búsquedas por lo que es necesario plantearos antes esta cuestión.

En la carpeta capitulos podemos diferenciar los siguientes campos en los CSV

- episode_id : keyWordAnalyzer
- number : keyWordAnalyzer
- timestamp_in_ms : keyWordAnalyzer
- raw_character_text : CustomAnalyzer
- raw_location_text : simpleAnalyzer
- spoken_words : standarAnalyzer

En la carpeta capitulos unidos diferenciamos los siguientes campos

- episode_id : keyWordAnalyzer
- spoken_words : stopAnalyzer
- raw_character_text : CustomAnalyzer
- imdb_rating : keyWordAnalyzer
- imdb_votes : keyWordAnalyzer
- numer_in_season : keyWordAnalyzer
- original_air_date : keyWordAnalyzer
- season : keyWordAnalyzer
- title : stopAnalyzer
- us_viewers_in_millions : keyWordAnalyzer
- views : keyWordAnalyzer

CustomAnalyzer

El analizador creado es un analizador que pone las palabras a minuscula y posteriormente reemplaza los espacios en blanco por _ para facilitar el posterior procesado de los datos

```
Analyzer analyzer = CustomAnalyzer.builder()
        .withTokenizer("standard")
        .addTokenFilter("lowercase")
        .addTokenFilter("patternreplace",
            "pattern" , "\\s+", // Eliminar espacios vacios
            "replacement" , "_" // Reemplazar por _ para posterior procesado de datos
        )
        .build();
```

3. Diseñar una consulta, para ello debemos de proporcionar los términos de la misma (con cuatro o mas términos) así como qué es lo que debe tener un documento para ser considerado relevante. Para encontrar los documentos (diálogos/capítulos) relevantes en la colección se debe lanzar la consulta sobre dos índices (uno para cada colección).

La consulta que he seleccionado ha sido Homer Simpson Duff Beer
Los resultados han sido los siguientes