

# Práctica Storage Technologies

Sept 1, 2024

## Índice

### 1 – Instalación de programario

#### **1.1 – Instalación de Hadoop**

#### **1.2 – Instalación de Spark**

#### **1.3 – Instalación de la máquina virtual**

### 2 – Primera pregunta

#### **2.1 – Partitioner de Hadoop**

#### **2.2 – ToolRunner de Hadoop**

### 3 – Segunda pregunta

### 4 – Tercera pregunta

### 5 – Cuarta pregunta

#### **5.1 – Redactado inicial del código**

#### **5.2 – Redactado final del código**

### 6 – Quinta pregunta

#### **6.1 – Creación de .class .jar por línea de comando y disposición de los archivos**

#### **6.2 – Modificación en el orden en las validaciones del Driver**

**6.3 – 1ª ejecución con falta de un input**

**6.4 – 2ª ejecución con falta de archivo de input**

**6.5 – 3ª ejecución procesando datos con formato no esperado**

**6.6 – 4ª ejecución sin previo borrado del directorio de salida**

**6.7 – 5ª ejecución con campo género en formato json**

**6.8 – 6ª ejecución con columnas asignadas erróneamente y bucle for innecesario**

**6.9 – 7ª ejecución. Ejecución correcta.**

**6.10 – Visualización y agrupación de los resultados**

**6.11 – Edge cases**

## **7– Fuentes de información y consulta**



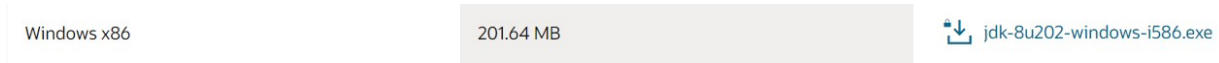


# 1 – Instalación de programario

## 1.1 – Instalación de Hadoop

Se sigue este videotutorial para instalar hadoop <https://www.youtube.com/watch?v=oTZPxyE6QoY>

Primero se instala Java de buscar en Google “java jfk 8” ir a <https://www.oracle.com/es/java/technologies/javase/javase8-archive-downloads.html>, seleccionar la siguiente opción de instalación, crear una cuenta y descargar.



En la descarga es importante que crear una carpeta “Java” en la raíz C/ en donde situemos la carpeta descargada y descomprimida.

En segundo lugar se instala Hadoop

En la carpeta “etc”, concretamente en la ruta C:\hadoop\hadoop-3.3.5\etc\hadoop, en el archivo “hadoop-env.cmd” se modifica la línea 25 para que ponga “set JAVA\_HOME=C:\Java\jdk1.8.0\_202” en vez de “set JAVA\_HOME=%JAVA\_HOME%”. De esta forma la versión de Hadoop está correctamente enlazada con Java como vemos en terminal. Primera ejecución de “hadoop -version” sin el cambio de dicho archivo. Segunda ejecución con el cambio en el archivo.

```
C:\>cd hadoop

C:\hadoop>cd hadoop-3.3.5

C:\hadoop\hadoop-3.3.5>cd bin

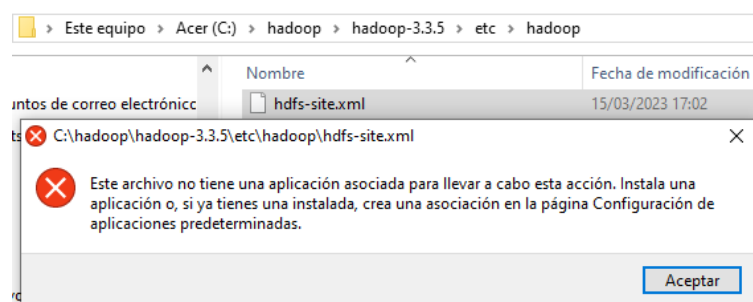
C:\hadoop\hadoop-3.3.5\bin>hadoop -version
java version "17.0.11" 2024-04-16 LTS
Java(TM) SE Runtime Environment (build 17.0.11+7-LTS-207)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.11+7-LTS-207, mixed mode, sharing)

C:\hadoop\hadoop-3.3.5\bin>hadoop -version
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) Client VM (build 25.202-b08, mixed mode)

C:\hadoop\hadoop-3.3.5\bin>
```

Seguidamente se debería acceder a “hdfs-site.xml” dentro de esta misma carpeta y modificar las configuraciones. No obstante con la opción de “Editar”

no deja modificar ese archivo



de la forma que sí lo hace en el minuto 11:00 del vídeo.

## 1.2 – Instalación de Spark

Al no poder terminar la instalación Hadoop, se sigue la recomendación de Laura Mestres Mercader de realizar la instalación de Spark de prácticas anteriores (práctica 9) ya que con esa instalación ya se debería tener una distribución Hadoop. Los pasos para tal descarga e instalación del Spark y levantamiento de spark-shell en local son los siguientes. Fuente totalmente sacada de práctica 9 con autores Albert Ripoll y José David Angulo.

### 1.2.1 – Descargas

<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>

Se descarga el “x64 Compressed Archive” seleccionando opciones JDK17 y Windows.

JDK 22   JDK 21   **JDK 17**   GraalVM for JDK 22   GraalVM for JDK 21   GraalVM for JDK 17

#### JDK Development Kit 17.0.11 downloads

JDK 17 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 17 will receive updates under the NFTC, until September 2024. Subsequent JDK 17 updates will be licensed under the [Java SE OTN License \(OTN\)](#) and production [limited free grants](#) of the OTN license will [require a fee](#).

Linux   macOS   **Windows**

Product/file description	File size	Download
x64 Compressed Archive	172.83 MB	<a href="https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip">https://download.oracle.com/java/17/latest/jdk-17_windows-x64_bin.zip</a> ( sha256)

<https://spark.apache.org/downloads.html>

Se descarga spark con enlace de opción 3

### Download Apache Spark™

1. Choose a Spark release: **3.5.1 (Feb 23 2024)** ▼
2. Choose a package type: **Pre-built for Apache Hadoop 3.3 and later** ▼
3. Download Spark: [spark-3.5.1-bin-hadoop3.tgz](#)
4. Verify this release using the 3.5.1 [signatures](#), [checksums](#) and [project release KEYS](#) by following these [procedures](#).



Se descarga winutils con el icono de la derecha

[winutils](#) / [hadoop-3.3.0](#) / [bin](#) / [winutils.exe](#)

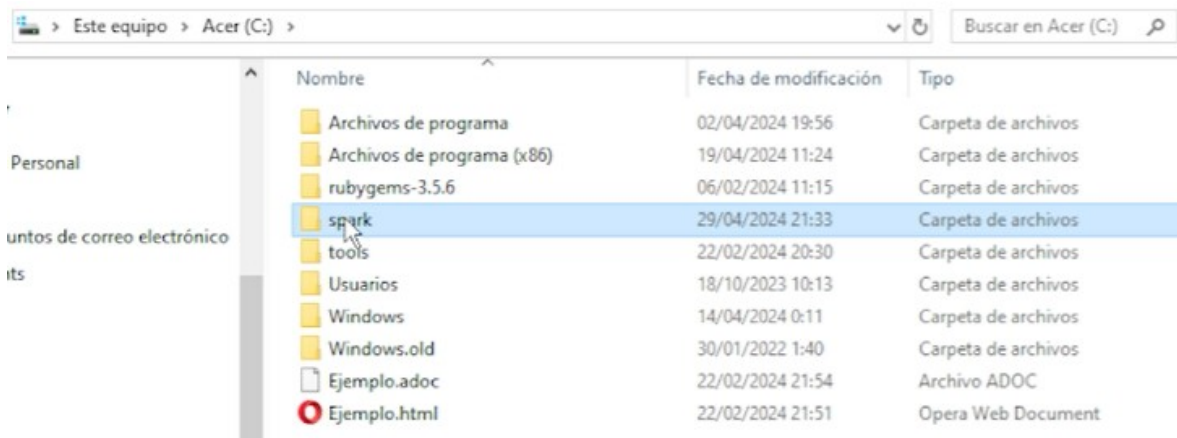
FahaoTang Added hadoop 3.3.0 winutils binary. b51cd5b · 4 years ago

Code   Blame   110 KB

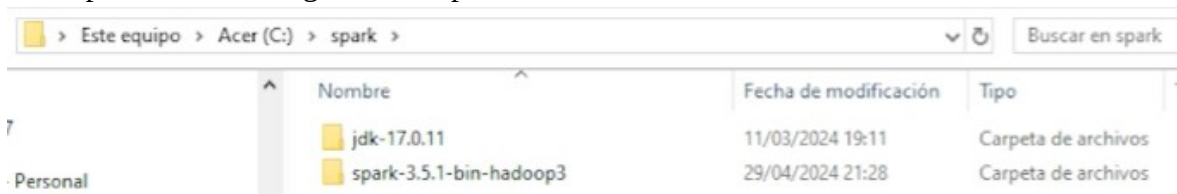
[View raw](#)

### 1.2.2 – Colocación de las descargas

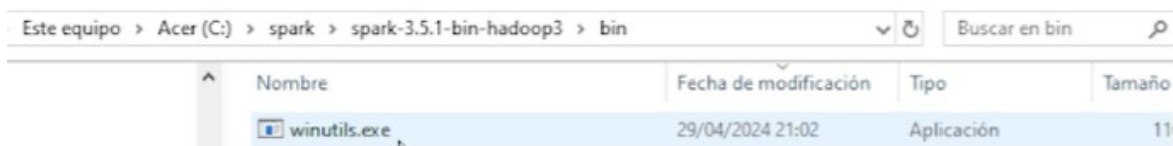
En C: Acer se crea una carpeta llamada “spark”



Dentro de esta carpeta “spark” hemos situado las dos carpetas “jdk-17.0.11” y “spark-3.5.1-bin-hadoop3” de las descargas descomprimidas.

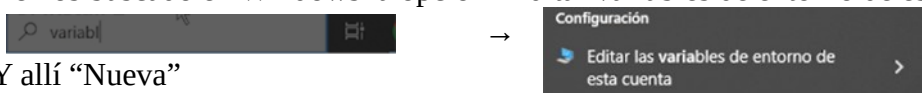


Dentro de “spark-3.5.1-bin-hadoop3” y allí dentro de la carpeta “bin” hemos situado el ejecutable “winutils.exe”.

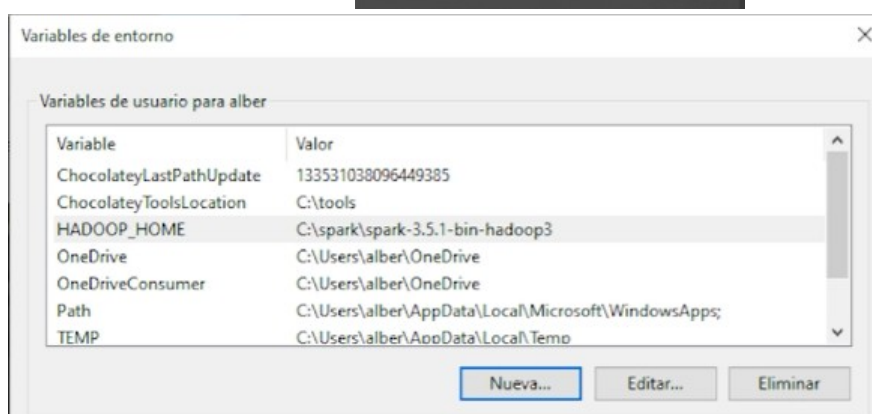


### 1.2.3 – Editar variables de entorno de esta cuenta

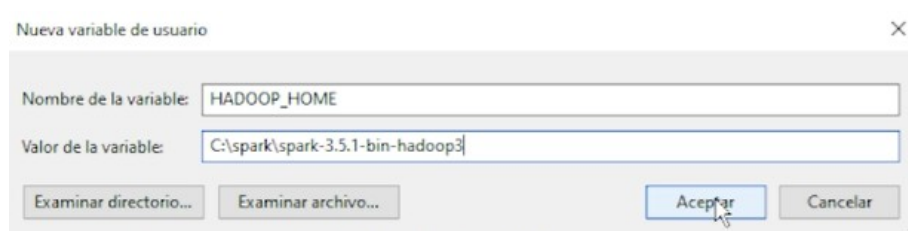
Hemos buscado en Windows la opción “Editar variables de entorno de esta cuenta”

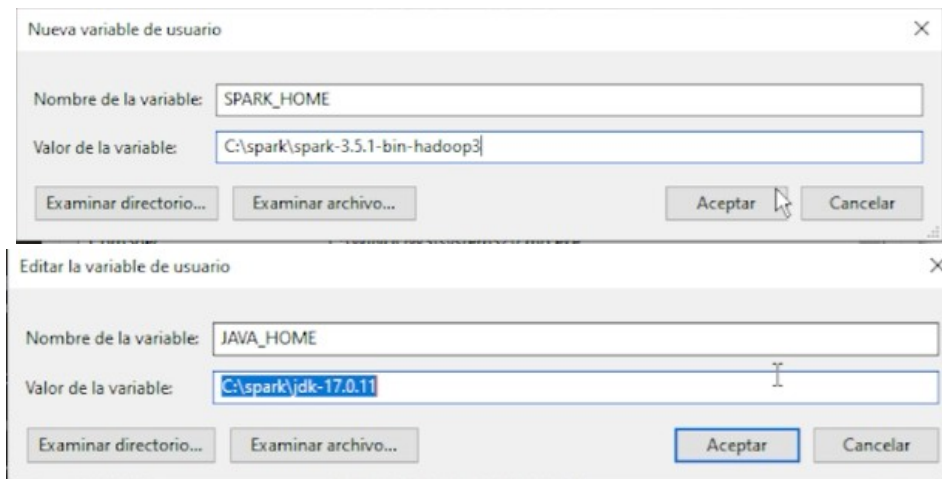


Y allí “Nueva”



Para añadir las distintas variables:

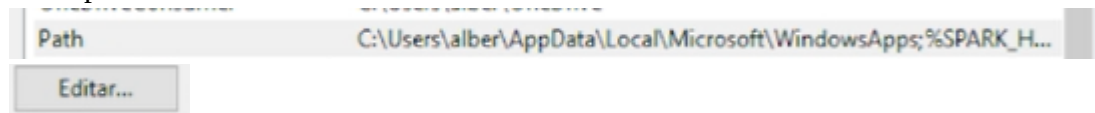




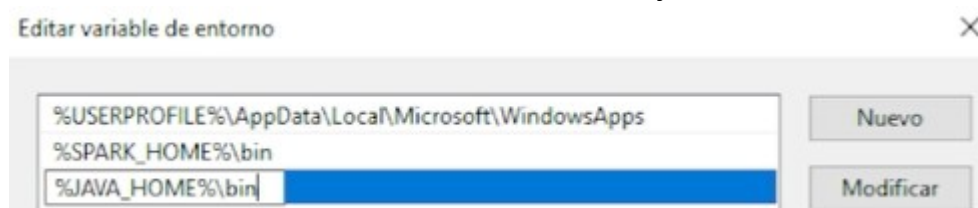
Tal que así:

SPARK_HOME	C:\spark\spark-3.5.1-bin-hadoop3
HADOOP_HOME	C:\spark\spark-3.5.1-bin-hadoop3
JAVA_HOME	C:\spark\jdk-17.0.11

En el path le damos a Editar

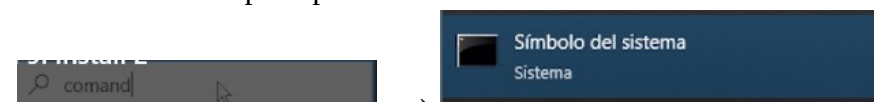


Y en “Nuevo” definimos %SPARK\_HOME%\bin y %JAVA\_HOME%\bin



## 1.2.4 – Comprobación de correcta instalación

Vamos al comand prompt



Nos situamos en la carpeta

```
C:\Users\alber>dir C:\spark
El volumen de la unidad C es Acer
El número de serie del volumen es: B82F-6625

Directorio de C:\spark

29/04/2024  21:33    <DIR>        .
29/04/2024  21:33    <DIR>        ..
11/03/2024  20:11    <DIR>        jdk-17.0.11
29/04/2024  21:28    <DIR>        spark-3.5.1-bin-hadoop3
                0 archivos            0 bytes
                4 dirs  659.941.773.312 bytes libres
```

Vamos a bin

```
C:\Users\alber>cd C:\spark\spark-3.5.1-bin-hadoop3\bin
```

Abrimos spark-shell

```
C:\spark\spark-3.5.1-bin-hadoop3\bin>spark-shell
```

Comprovamos



```
C:\spark\spark-3.5.1-bin-hadoop3>echo %JAVA_HOME%  
C:\spark\jdk-17.0.11
```

Ejecutamos spark-shell

```
C:\spark\spark-3.5.1-bin-hadoop3\bin>spark-shell
```

## 1.3 – Instalación de la máquina virtual

De la práctica 2 de HDSF de la asignatura se facilita el siguiente enlace

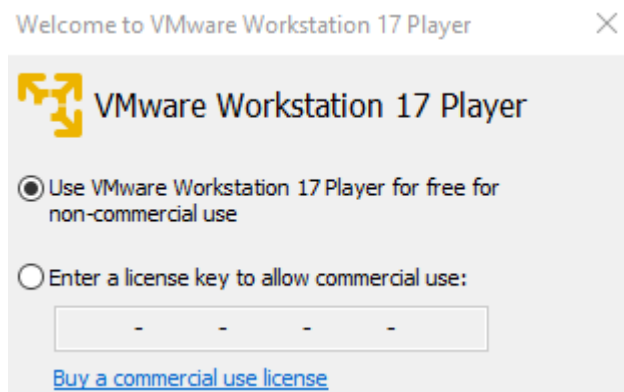
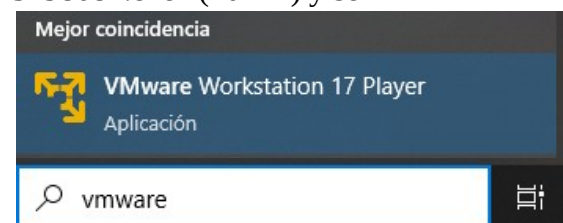
[https://lasalleuniversities-my.sharepoint.com/:u:/r/personal/laura\\_mestres\\_salle\\_url\\_edu/Documents/Storage%20technologies%202023/Developer\\_Hadoop.rar?download=1](https://lasalleuniversities-my.sharepoint.com/:u:/r/personal/laura_mestres_salle_url_edu/Documents/Storage%20technologies%202023/Developer_Hadoop.rar?download=1)

Desde este enlace se puede descargar ZIP “Developer\_Hadoop.rar” (1h30) y descomprimirlo (30 min).

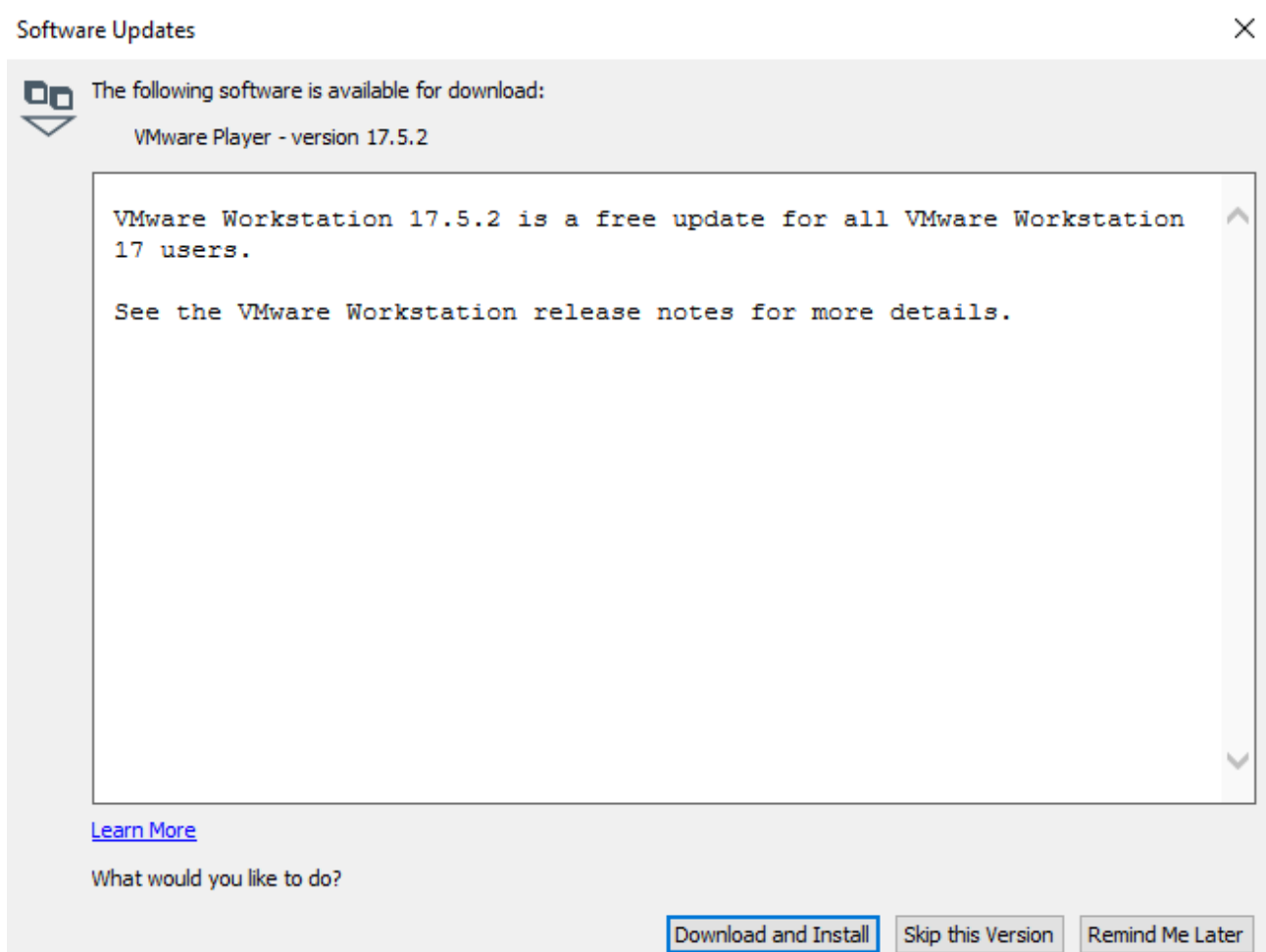
Seguidamente se descarga el archivo “Vmware-player-full-17.5.1-23298084.exe” (40min) y se ejecuta para instalar la máquina virtual.

De esta forma ya se tiene la máquina virtual instalada:

Al abrir la máquina virtual tenemos que seleccionar la opción de usar la máquina gratuitamente por un uso no comercial:



Seguidamente nos informa de actualizaciones disponibles a lo que respondemos con “Skip this Version”. Saltar esa versión. Ya que en principio ya hemos instalado la version que se necesitaba.



## 2 – Primera pregunta

“Explica qué es y para qué sirve tanto el Partitioner como el ToolRunner de Hadoop.”

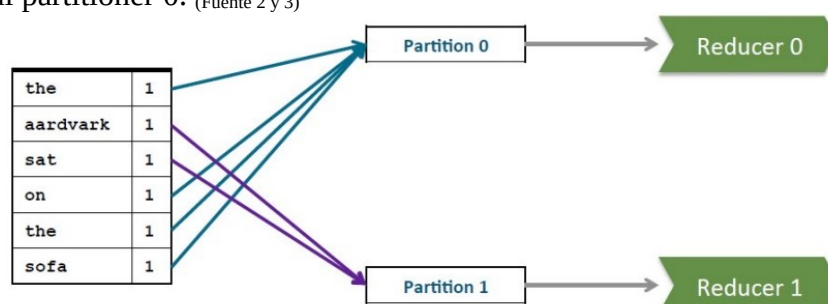
### 2.1 – Partitioner de Hadoop

“Explica qué es y para qué sirve el Partitioner de Hadoop.”

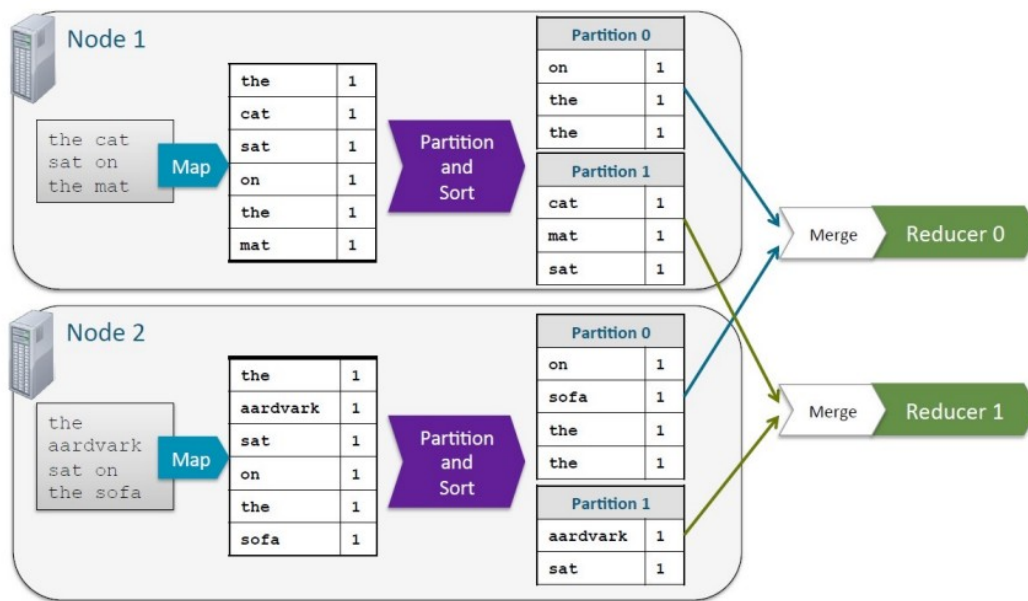
El Partitioner en Hadoop es un componente o tipo de tarea de MapReduce <sup>(Fuente 2)</sup> que controla la distribución de los datos entre los distintos reducers durante la fase de shuffle and sort. Su función principal es determinar a cuál de los reducers se debe enviar cada clave generada por los mappers.

(Fuente 4)

Lo que determina el Partitioner es en qué reducer va cada key. O sea, distribuye las key en los reducers. Es importante, para el correcto funcionamiento, que las mismas key vayan al mismo reducer para que lo cuente bien. Por ejemplo en una tarea de conteo de palabras, es crucial que todos los “the” vayan al partitioner 0. <sup>(Fuente 2 y 3)</sup>



O en este ejemplo podemos ver claramente que la partición 0 contiene las palabras “the” y “on” mientras que la partición 1 contiene la palabra “sat”. Por esa razón la partición 0 se junta (*merges*) en el reducer 0 y la partición 1 se junta (*merges*) en el reducer 1. (Fuente 3)



En palabras propias y simples, lo que hace el componente partitioner es particionar la salida del mapeo para distribuir los datos en distintos reducers. Este proceso tiene las siguientes utilidades:

- Distribuir los datos: Asegura que los datos estén distribuidos uniformemente entre los reducers, evitando sobrecargas en algunos reducers mientras otros están inactivos.
- Controlar la lógica de la partición: Permite definir reglas personalizadas en la partición de los datos mediante la implementación de la función `getPartition()`.
- Optimizar el rendimiento: Una buena estrategia de particionamiento puede mejorar significativamente el rendimiento de un trabajo MapReduce, evitando cuellos de botella en ciertos reducers, hecho que es crucial cuando se trabaja con grandes volúmenes de datos.

(Fuente 4)

## 2.2 – ToolRunner de Hadoop

“Explica qué es y para qué sirve el ToolRunner de Hadoop.”

ToolRunner es un componente no requerido pero si una buena práctica de uso en las clases driver.

(Fuente 3) Esta utilidad proporcionada por Hadoop facilita la ejecución de clases al brindar una interfaz para las herramientas, argumentos y aplicaciones de Hadoop que se necesitan para un *job*. (Fuente 4)

ToolRunner es útil ya que:

- Facilita la ejecución: Permite especificar opciones de configuración en la línea de comandos brindando una manera simplificada de obtener y manejar Hadoop. (Fuente 2, 3 y 4)
- Maneja la configuración: Sirve para manejar su *configuration* sobrescribiendo parámetros por defecto en configuración sin sobrescribir los parámetros seteados con código en el Driver. También permite especificar las propiedades de Hadoop usando *flag -D* o especificar configuraciones de ficheros XML con el parámetro *-conf*. (Fuente 3)
- Gestiona los argumentos: Permite el análisis y la gestión de los argumentos de la línea de comandos que se pasan al programa MapReduce. (Fuente 4)

En palabras propias y simples, ToolRunner actúa como un asistente que ayuda a recoger los ingredientes necesarios y a guiar en las distintas etapas para realizar un trabajo de MapReduce. Facilita que el trabajo sea más fácil y rápido.

### 3 – Segunda pregunta

---

“Busca en Internet un dataset interesante que te permita demostrar el funcionamiento del Partitioner y el ToolRunner. Describe el dataset y justifica por qué es apropiado para esta práctica.”

Para este ejercicio, usaremos un dataset de películas disponibles en Kaggle - The Movies Dataset.

<https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset?resource=download>

Este dataset contiene información sobre películas, incluyendo su título, género, calificación e ingresos. Columnas del dataset:

- `movieId`: Identificador único de la película.
- `title`: Título de la película.
- `genres`: Género de la película.
- `rating`: Calificación promedio de la película.
- `revenue`: Ingresos generados de la película.

Es un dataset adecuado para demostrar el funcionamiento del Partitioner y el ToolRunner porque tiene múltiples campos que pueden ser utilizados para particionar los datos. En el caso de esta práctica se particionarán las películas por género. Es decir, se distribuirán los datos en diferentes reducers según su género. A su vez, el dataset permitirá el uso del ToolRunner demostrando cómo se usan parámetros por línea de comandos. En el caso de esta práctica se filtrarán las películas por umbral de ingresos.

### 4 – Tercera pregunta

---

“Propón un ejercicio no trivial en el que usar un Partitioner y el ToolRunner (para pasar algún parámetro por línea de comandos) sobre el dataset que has elegido en la pregunta anterior. Asegúrate de que es un ejercicio novedoso y no está resuelto en Internet.”

El ejercicio que se propone realiza lo siguiente:

1. Particiona las películas según su género.
2. Usa ToolRunner para aceptar un parámetro de línea de comandos y determina un umbral de ingresos.
3. Filtra las películas por un umbral de ingresos y calcula la calificación promedio en cada género.

Para realizar este ejercicio:

1) Doble click en la máquina virtual, se selecciona uso no comercial y se saltan las actualizaciones.

2) Se abre una nueva máquina virtual.



#### Open a Virtual Machine

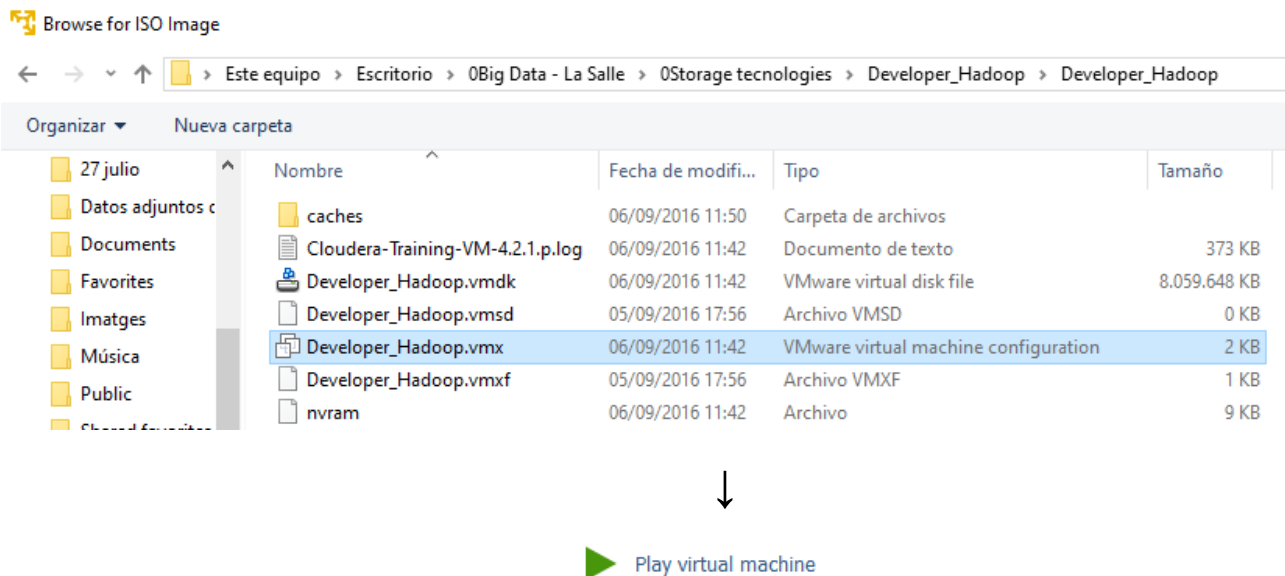
Open an existing virtual machine, which will then be added to the top of your library.



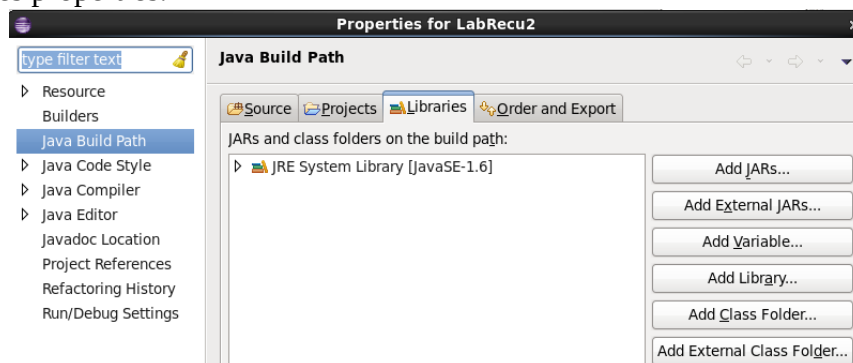
● Installer disc image file (iso):

Browse...

⇒ Select the installer disc image to continue.

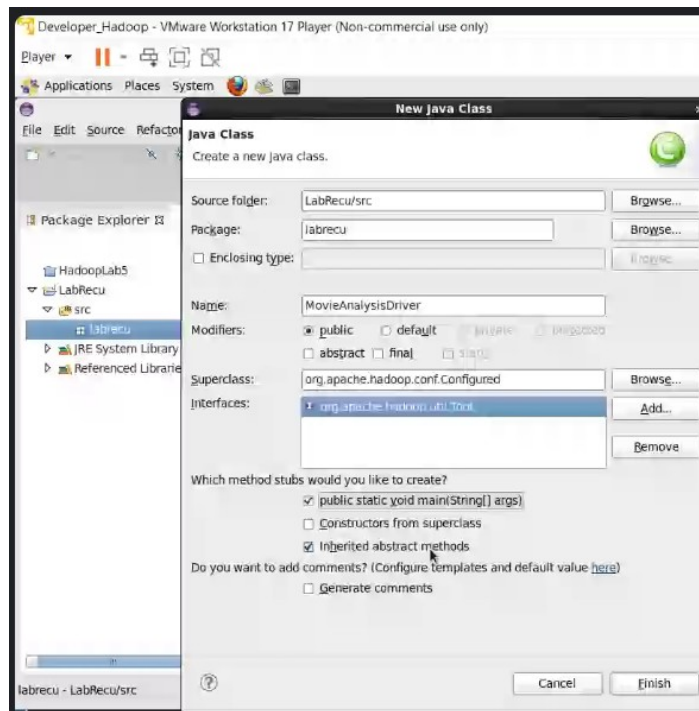


3) Una vez abierta la máquina virtual, doble click en Eclipse. En la izquierda de *package explorer* doble click en crear proyecto y en *New Java Project Name* se introduce “LabRecu” como nombre del proyecto. Por defecto se usa java 7. Se clicka en *Next*, *Next*. Se hace click derecho en el proyecto, después *properties*.



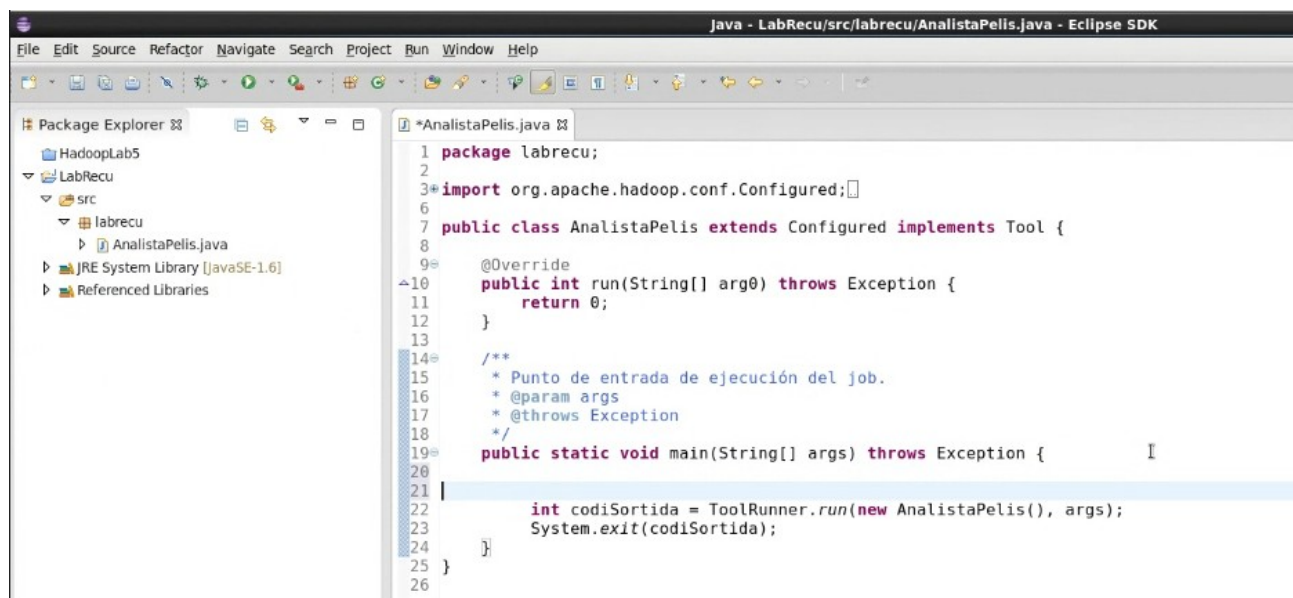
Entonces en las pestañas *Source*, *Project*, *Libraries* se va a la pestaña *Libraries* en donde se clicka en el segundo botón llamado “Add externals jars”. Allí se va a `/User/Lib/Hadoop/Client-0.20` ya que los Jars son los lugares donde se compilan los Drivers de Hadoop. En el listado se selecciona todo gracias a los botones `Shit + ↓` y click en el botón `OK` dos veces. Entonces la lista se llena de librerías Jars que son como unos Zip. Aquí hay implicados distintos conceptos: El *ClassLoader* que es el que sabe lo que hay dentro de cada Jar. El *Classpath* que es el que hace referencia a las librerías que se tienen que usar. *Package* que es un espacio de nombres usado para agrupar código.

(Fuente 5 y 6)

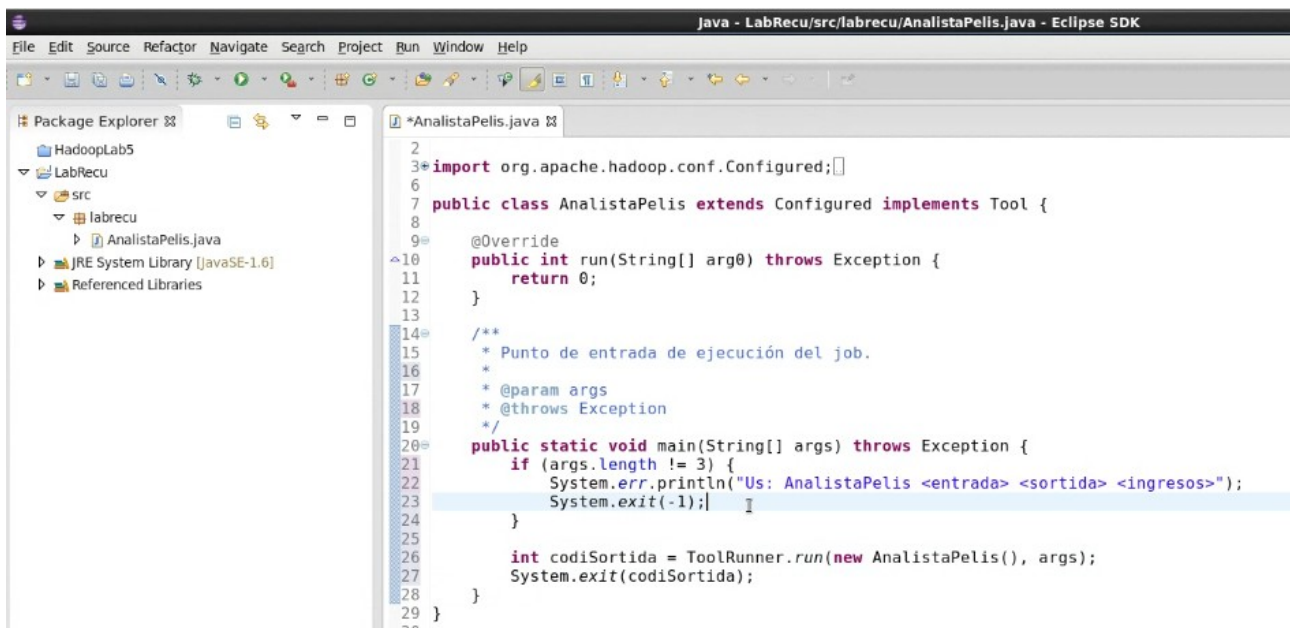


4) Una vez finalizado el llenado de librerías Jars explicadas en el punto 3, se le da al botón Finish. Entonces, se crean los package. SRC → NEW → PACKAGE. En nombre se coloca el nombre del paquete en este caso se introduce “LabRecu”. (Fuente 5)

5) Una vez creados los paquetes, se crea el driver o en otras palabras el código encargado de iniciar la ejecución del job. Sería redactar el código del paso 1 de la cuarta pregunta y adecuarlo todo junto.



El ToolRunner ejecuta una instancia. El `codiSortida` lo que hace es que si eso termina en un valor diferente de zero es porque hay un error, entonces `codiSortida` captura el mensaje de entero que se le ponga en el método “run”. El método “run” es la convención de que se va a ejecutar el método “run” que está definido en la clase `AnalistaPelis`. Si devuelve 0 está todo bien, sino es que hubo algún fallo. Al decir “`System.exit`” se está diciendo que se termine la aplicación con el siguiente código de error. (Fuente 5)



```
2
3 import org.apache.hadoop.conf.Configured;
6
7 public class AnalistaPelis extends Configured implements Tool {
8
9     @Override
10    public int run(String[] args) throws Exception {
11        return 0;
12    }
13
14    /**
15     * Punto de entrada de ejecución del job.
16     *
17     * @param args
18     * @throws Exception
19     */
20    public static void main(String[] args) throws Exception {
21        if (args.length != 3) {
22            System.err.println("Us: AnalistaPelis <entrada> <sortida> <ingresos>");
23            System.exit(-1);
24        }
25
26        int codiSortida = ToolRunner.run(new AnalistaPelis(), args);
27        System.exit(codiSortida);
28    }
29 }
```

En Java se pueden tener distintas clases definidas en un archivo pero el archivo se tiene que llamar como la primera clase definida. En este caso `AnalistaPelis`. En este código se incluyen las líneas 21 y 22 que verifica si la longitud de los argumentos de entrada es diferente de 3. En el caso que los argumentos de entrada no sean 3 (o sea que no se pasen los 3 valores necesarios a la clase), salta un error que escribe como se tiene que usar la clase `AnalistaPelis` ya que nos recuerda que el uso de esa clase es una entrada, una salida y un umbral de beneficios. Finalmente, al decir “`System.exit`” se está diciendo que se termine la aplicación con el código de error “-1”. El código funciona y está enlazado correctamente porque la variable “`args`” está definida en un sitio, llamada en otro sitio y usada finalmente con el mismo nombre. Se sitúa el código dentro del `run`. (Fuente 5)

Se necesita crear las clases `GenrePartitioner`, `MovieReducer`, `MovieMapper` con la estructura `public class GenrePartitioner {}`, `public class MovieReducer {}` y `public class MovieMapper {}`. Se extiende la clase `MovieMapper`: `public class MovieMapper extends Mapper<LongWritable, Text, Text, MovieWritable> {}` y se define su implementación en los campos título, calificación e ingreso que por una parte es la escribir esos campos (`write`) y por otra parte es leerlos (`readFields`): `public void write(DataOutput out) throws IOException {}`, `public void readFields(DataInput in) throws IOException {}`

Para generar el .zip de salida para entregar se selecciona en la esquina superior derecha `Places>HomeFolder`. Allí se va a la dirección `/home/training/workspace/` en donde hay guardado el archivo `LabRecu`. Allí se hace click derecho `>compress...>create` para que se cree el `LabRecu.zip`.

(Fuente 4 y 6)

## 5 – Cuarta pregunta

“Escribe el código java que resuelva el ejercicio propuesto. Explica el funcionamiento de cada clase así como las precondiciones y postcondiciones de todos los métodos.”

### 5.1 – Redactado inicial del código

#### Paso 1: Definición del Partitioner (ParticionadorGenere)



La clase `ParticionadorGenere` extiende `Partitioner<Text, PeliculaWritable>` e implementa el método `obtenerParticio`. Es decir aumenta la funcionalidad de la clase. El método toma de una película su género como clave y calcula el particionador basado en el hash del género. Lo que resulta del `Partitioner` es que las películas se distribuyen entre los reducers por su género, asegurándose que las películas de un género específico se procesen todas en el mismo reducer.

```
package LabRecu;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Partitioner;

public class ParticionadorGenere extends Partitioner<Text, PeliculaWritable> {
    @Override
    public int getPartition(Text clau, PeliculaWritable valor, int numParticions) {
        return (clau.toString().hashCode() & Integer.MAX_VALUE) % numParticions;
    }
}
```

(Fuente 4 y 6)

## **Paso 2: Definición del ToolRunner (AnalistaPelis)**

La clase `AnalistaPelis` extiende `Configured` e implementa `Tool`. Implementa un método `run` que configura y lanza el trabajo MapReduce. Usa `ToolRunner` para manejar la entrada de parámetros desde la línea de comandos. El parámetro de umbral de ingresos se pasa al trabajo de MapReduce a través de la configuración de Hadoop. La configuración del trabajo incluye la especificación de las clases de Mapper, Reducer y Partitioner, así como los formatos de entrada y salida.

```
package LabRecu;

import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class AnalistaPelis extends Configured implements Tool {
    @Override
    public int run(String[] args) throws Exception {
        if (args.length != 3) {
            System.err.println("Ús: AnalistaPelis <input path> <output path> <revenue threshold>");
            System.exit(-1);
        }

        Job job = Job.getInstance(getConf(), "Anàlisi de Pel·lícules");
        job.setJarByClass(AnalistaPelis.class);

        job.setMapperClass(MapejadorPelicules.class);
        job.setReducerClass(ReducidorPelicules.class);
        job.setPartitionerClass(ParticionadorGenere.class);
        job.setNumReduceTasks(3);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(PeliculaWritable.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.getConfiguration().set("revenue.threshold", args[2]);

        return job.waitForCompletion(true) ? 0 : 1;
    }
}
```



```

        public static void main(String[] args) throws Exception {
            int exitCode = ToolRunner.run(new AnalistaPelis(), args);
            System.exit(exitCode);
        }
    }
}

```

(Fuente 4 y 6)

### **Paso 3: Definición de PelículaWritable**

La clase PelículaWritable implementa la interfaz Writable y encapsula los campos `titol`, `valoracio` y `ingressos` de una película. Se define cómo se deben leer y escribir estos campos para permitir que Hadoop los transmita entre el Mapper y el Reducer. Esto es esencial para mantener los datos estructurados durante el proceso de MapReduce.

```

package LabRecu;

import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class PeliculaWritable implements Writable {
    private Text titol;
    private DoubleWritable valoracio;
    private DoubleWritable ingresos;

    public PeliculaWritable() {
        this.titol = new Text();
        this.valoracio = new DoubleWritable();
        this.ingressos = new DoubleWritable();
    }

    public PeliculaWritable(String titol, double valoracio, double ingresos) {
        this.titol = new Text(titol);
        this.valoracio = new DoubleWritable(valoracio);
        this.ingressos = new DoubleWritable(ingressos);
    }

    @Override
    public void write(DataOutput out) throws IOException {
        titol.write(out);
        valoracio.write(out);
        ingresos.write(out);
    }

    @Override
    public void readFields(DataInput in) throws IOException {
        titol.readFields(in);
        valoracio.readFields(in);
        ingresos.readFields(in);
    }

    public Text obtenerTitol() {
        return titol;
    }

    public DoubleWritable obtenerValoracio() {
        return valoracio;
    }

    public DoubleWritable obtenerIngressos() {
        return ingresos;
    }
}

```

(Fuente 4 y 6)

#### **Paso 4: Definición del Mapper y del Reducer (MapejadorReduidor)**

La clase `MapejadorPelicules` extiende `Mapper<LongWritable, Text, Text, PeliculaWritable>`. En el método `setup`, obtiene el umbral de ingresos de la configuración del trabajo. En el método `map`, procesa cada línea de entrada, filtra las películas que superan el umbral de ingresos y emite un par (género, `MovieWritable`) para cada género asociado a una película. Esto permite que las películas se agrupen por género para el procesamiento posterior en el reducer.

La clase `ReduidorPelicules` extiende `Reducer<Text, PeliculaWritable, Text, Text>`. En el método `reduce`, recibe una lista de `PeliculaWritable` para cada género, calcula la calificación promedio y emite el género y la calificación promedio como resultado. Esto resume los datos filtrados y particionados por el Mapper y Partitioner.

```
package LabRecu;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;
import org.apache.hadoop.io.LongWritable;
//import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
//import java.io.IOException;

class MapejadorPelicules extends Mapper<LongWritable, Text, Text, PeliculaWritable> {
    private double llindarIngressos;

    @Override
    protected void setup(Context context) {
        String llindarString = context.getConfiguration().get("revenue.threshold",
"0.0");
        llindarIngressos = Double.parseDouble(llindarString);
    }

    @Override
    protected void map(LongWritable clau, Text valor, Context context) throws
IOException, InterruptedException {
        String[] camps = valor.toString().split(";");
        String titol = camps[1];
        String generes = camps[2];
        double valoracio = Double.parseDouble(camps[3]);
        double ingressos = Double.parseDouble(camps[4]);

        if (ingressos > llindarIngressos) {
            for (String genere : generes.split("\\|")) {
                context.write(new Text(genere), new PeliculaWritable(titol, valoracio,
ingressos));
            }
        }
    }
}

class ReduidorPelicules extends Reducer<Text, PeliculaWritable, Text, Text> {
    @Override
    protected void reduce(Text clau, Iterable<PeliculaWritable> valors, Context context)
throws IOException, InterruptedException {
        int comptador = 0;
        double valoracioTotal = 0.0;

        for (PeliculaWritable valor : valors) {
            valoracioTotal += valor.obtenirValoracio().get();
            comptador++;
        }

        double valoracioMitjana = valoracioTotal / comptador;
        context.write(clau, new Text("Valoració mitjana: " + valoracioMitjana));
    }
}
```

## 5.2 – Redactado final del código

```
AnalistaPelis.java PeliculaWritable.java ParticionadorGenere.java *MapejadorReduidor.java

1 package labrecu;
2
3 import java.io.DataInput;
20
21 public class AnalistaPelis extends Configured implements Tool {
22
23     public class MovieWritable implements Writable {
24
25         @Override
26         public void readFields(DataInput input) throws IOException {
27         }
28
29         @Override
30         public void write(DataOutput output) throws IOException {
31         }
32     }
33
34     public class GenrePartitioner extends Partitioner<Text, MovieWritable> {
35
36         @Override
37         public int getPartition(Text text, MovieWritable value, int numpartitions) {
38             return 0;
39         }
40     }
41
42     public class MovieReducer extends Reducer<Text, MovieWritable, Text, Text> {
43     }
44
45     public class MovieMapper extends Mapper<LongWritable, Text, Text, MovieWritable> {
46
47         @Override
48         public int run(String[] args) throws Exception {
49             Job job = Job.getInstance(getConf(), "Movie Analysis");
50             job.setJarByClass(AnalistaPelis.class);
51             job.setMapperClass(MovieMapper.class);
52             job.setReducerClass(MovieReducer.class);
53             job.setPartitionerClass(GenrePartitioner.class);
54             job.setNumReduceTasks(3);
55             job.setOutputKeyClass(Text.class);
56             job.setOutputValueClass(MovieWritable.class);
57             FileInputFormat.addInputPath(job, new Path(args[0]));
58             FileOutputFormat.setOutputPath(job, new Path(args[1]));
59             job.getConfiguration().setFloat("revenue.threshold", Float.parseFloat(args[2]));
60             return job.waitForCompletion(true) ? 0 : 1;
61         }
62
63         /**
64          * Punto de entrada de ejecución del job.
65          *
66          * @param args
67          * @throws Exception
68          */
69         public static void main(String[] args) throws Exception {
70             if (args.length != 3) {
71                 System.err.println("Us: AnalistaPelis <entrada> <sortida> <ingresos>");
72                 System.exit(-1);
73             }
74             int codiSortida = ToolRunner.run(new AnalistaPelis(), args);
75             System.exit(codiSortida);
76         }
77     }
78 }
79 }
```

```
1 package LabRecu;
2
3 import org.apache.hadoop.io.DoubleWritable;
10
11 public class PeliculaWritable implements Writable {
12     private Text titol;
13     private DoubleWritable valoracio;
14     private DoubleWritable ingressos;
15
16     public PeliculaWritable() {
17         this.titol = new Text();
18         this.valoracio = new DoubleWritable();
19         this.ingressos = new DoubleWritable();
20     }
21
22     public PeliculaWritable(String titol, double valoracio, double ingressos) {
23         this.titol = new Text(titol);
24         this.valoracio = new DoubleWritable(valoracio);
25         this.ingressos = new DoubleWritable(ingressos);
26     }
27
28     @Override
29     public void write(DataOutput out) throws IOException {
30         titol.write(out);
31         valoracio.write(out);
32         ingressos.write(out);
33     }
34
35     @Override
36     public void readFields(DataInput in) throws IOException {
37         titol.readFields(in);
38         valoracio.readFields(in);
39         ingressos.readFields(in);
40     }
41
42     public Text obtenirTitol() {
43         return titol;
44     }
45
46     public DoubleWritable obtenirValoracio() {
47         return valoracio;
48     }
49
50     public DoubleWritable obtenirIngressos() {
51         return ingressos;
52     }
53 }
```

```

AnalistaPelis.java PeliculaWritable.java *ParticionadorGenere.java *MapejadorReduidor.java
1 package LabRecu;
2
3 import org.apache.hadoop.io.Text;
4 import org.apache.hadoop.mapreduce.Partitioner;
5
6 public class ParticionadorGenere extends Partitioner<Text, PeliculaWritable> {
7     @Override
8     public int getPartition(Text clau, PeliculaWritable valor, int numParticions) {
9         return (clau.toString().hashCode() & Integer.MAX_VALUE) % numParticions;
10    }
11 }
12
13 //clau.toString() són els gèneres:
14 //Action» 1
15 //Animation» 2
16 //Comedy» 3
17 //Fantasy» 4
18 //Foreign» 5
19 //Music» 6
20 //Mystery» 7
21 //Romance» 8
22
23 //clau.toString().hashCode() calcula un número de cada gènere és un número determinista en base als caràcters
24 // que hi ha a cada gènere
25 //Action» 1 (sempre el mateix número a cadascú, no té perquè ser 1 indexat)
26 //Animation» 2 (sempre el mateix número a cadascú, no té perquè ser 1 indexat)
27 //Comedy» 3 (sempre el mateix número a cadascú, no té perquè ser 1 indexat)
28 //Fantasy» 4 (sempre el mateix número a cadascú, no té perquè ser 1 indexat)
29 //Foreign» 5 (sempre el mateix número a cadascú, no té perquè ser 1 indexat)
30 //Music» 6 (sempre el mateix número a cadascú, no té perquè ser 1 indexat)
31 //Mystery» 7 (sempre el mateix número a cadascú, no té perquè ser 1 indexat)
32 //Romance» 8 (sempre el mateix número a cadascú, no té perquè ser 1 indexat)
33
34 //clau.toString().hashCode() & Integer.MAX_VALUE -> és un and binari que calcula en binari el número fins com a màxim  $2^{31}$ 
35 //1 -> 0001
36 //2 -> 0010
37 //3 -> 0011
38 //4 -> 0100
39
40 //x % numParticions -> Després es divideix la operació binària (x) entre el número de particions
41 //Aquest valor en aquest cas retorna número entre 0 i numParticions-1. 0 sigui 0,1,2.

```

```
AnalistaPelis.java PeliculaWritable.java *ParticionadorGenere.java *MapejadorReduidor.java
1 package LabRecu;
2
3 import org.apache.hadoop.io.Text;
4
5 class MapejadorPelis extends Mapper<LongWritable, Text, Text, PeliculaWritable> {
6     private double llindarIngressos;
7
8     @Override
9     protected void setup(Context context) {
10         String llindarString = context.getConfiguration().get("revenue.threshold", "0.0");
11         llindarIngressos = Double.parseDouble(llindarString);
12     }
13
14     @Override
15     protected void map(LongWritable clau, Text valor, Context context) throws IOException, InterruptedException {
16         String[] camps = valor.toString().split(",");
17         // En el csv hi ha aquesta estructura de les columnes: budget,original_title,revenue,title,vote_average,vote_count,GenreAleatori
18         // Per això escollim 1 com el camp de títol, 6 com el camp de gènere...
19         // Si l'estructura té 7 camps, procesa. Sinó és un error i s'ignora (no es processa) aquest camp.
20         // Si a la conversió de números (si el camp 4 i el 2) són números i no ha fallat, procesa.
21         // Sinó és que ha fallat per exemple si fos un nan i s'ignora (no es processa) aquest camp.
22         if (camps.length == 7 && NumberUtils.isNumber(camps[4]) && NumberUtils.isNumber(camps[2])) {
23             String titol = camps[1];
24             String genere = camps[6];
25             double valoracio = Double.parseDouble(camps[4]);
26             double ingressos = Double.parseDouble(camps[2]);
27
28             if (ingressos > llindarIngressos) {
29                 context.write(new Text(genere), new PeliculaWritable(titol, valoracio, ingressos));
30             }
31         } else {
32             System.out.println("El format no és correcte: " + valor.toString());
33         }
34     }
35 }
36
37 class ReduidorPelis extends Reducer<Text, PeliculaWritable, Text, Text> {
38     @Override
39     protected void reduce(Text clau, Iterable<PeliculaWritable> valors, Context context) throws IOException, InterruptedException {
40         int comptador = 0;
41         double valoracioTotal = 0.0;
42
43         for (PeliculaWritable valor : valors) {
44             valoracioTotal += valor.obtenirValoracio().get();
45             comptador++;
46         }
47
48         double valoracioMitjana = valoracioTotal / comptador;
49         context.write(clau, new Text("Valoració promig: " + valoracioMitjana));
50     }
51 }
```

53:40 explicació del codi sencer de java.

## 6 – Quinta pregunta

“Muestra tantos ejemplos de ejecución para demostrar que el código funciona correctamente (incluyendo *edge cases*).”

### 6.1 – Creación de .class .jar por línea de comando y disposición de los archivos

- Después de crear las cuatro clases en Eclipse explicadas, se accede en local con la terminal de la misma máquina virtual para comprobar que están correctamente localizadas en su ruta. (Fuente 7)

```
[training@localhost ~]$ ls
Desktop      kiji-bento-albacore-1.0.5-release.tar.gz  Pictures      src      Videos
Documents    [redacted]                                Public        Templates workspace
Downloads    lib                                         scripts       training workspace.save.dev
eclipse      Music                                     [redacted]    training_materials

[training@localhost ~]$ cd workspace
[training@localhost workspace]$ ls
hive LabRecu LabRecu2 oozie-labs _pasted_code_
[training@localhost workspace]$ cd LabRecu2
[training@localhost LabRecu2]$ ls
bin src
[training@localhost LabRecu2]$ cd src
[training@localhost src]$ ls
LabRecu
[training@localhost src]$ cd LabRecu
[training@localhost LabRecu]$ ls
AnalistaPelis.java MapejadorReduidor.java ParticionadorGenere.java PeliculaWritable.java
[training@localhost LabRecu]$
```

- Con el siguiente commando, se crean las distintas clases dentro de la carpeta “LabRecu”. (Fuente 7)

```
[training@localhost LabRecu]$ javac -classpath `hadoop classpath` MapejadorReduidor.java PeliculaWritable.java
va ParticionadorGenere.java AnalistaPelis.java
```

- Y con el siguiente commando se crea el documento comprimido “LabRecu.jar”. (Fuente 7)

```
[training@localhost LabRecu]$ jar cf LabRecu.jar MapejadorReduidor.java PeliculaWritable.java ParticionadorG
enere.java AnalistaPelis.java _
```

Al volver a acceder a la carpeta se observa que se han creado correctamente los .class a partir de los .java que existían y también que se ha creado el .jar

```
[training@localhost src]$ cd LabRecu
[training@localhost LabRecu]$ ls
AnalistaPelis.class MapejadorPelicules.class ParticionadorGenere.java ReduidorPelicules.class
AnalistaPelis.java MapejadorReduidor.java PeliculaWritable.class
LabRecu.jar ParticionadorGenere.class PeliculaWritable.java
```

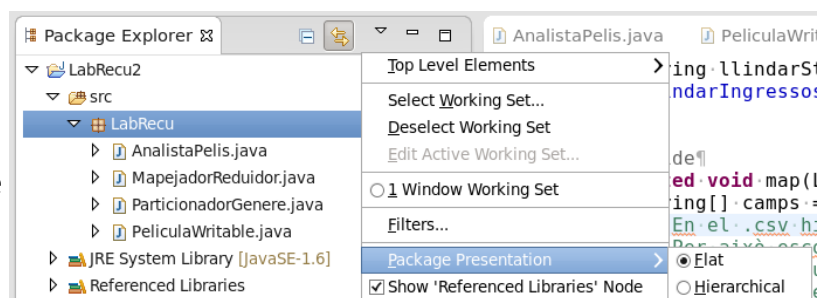
- Queda ordenado así:

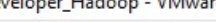
```
[training@localhost ~]$ ls
Desktop      kiji-bento-albacore-1.0.5-release.tar.gz  Pictures      src      Videos
Documents    [redacted]                                Public        Templates workspace
Downloads    lib                                         scripts       training workspace.save.dev
eclipse      Music                                     [redacted]    training_materials

[training@localhost ~]$ cd workspace
[training@localhost workspace]$ ls
hive LabRecu LabRecu2 oozie-labs _pasted_code_
[training@localhost workspace]$ cd LabRecu2
[training@localhost LabRecu2]$ ls
bin      keywords.csv LabRecu.jar links_small.csv ratings.csv src
credits.csv LabRecu2.jar links.csv movies_metadata.csv ratings_small.csv
[training@localhost LabRecu2]$
```

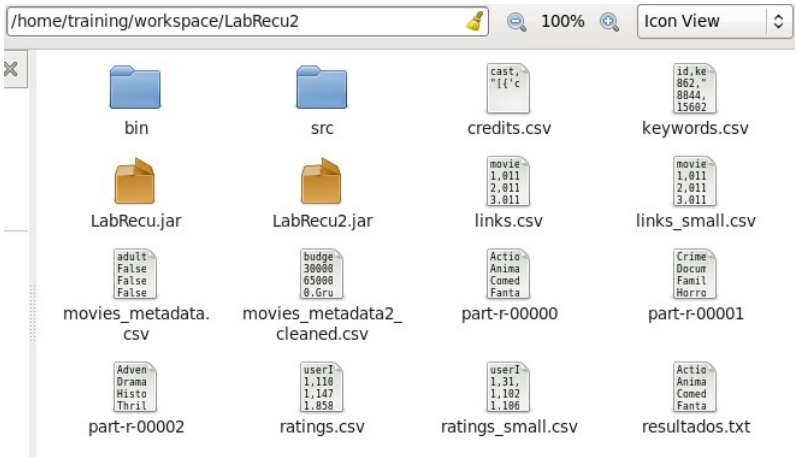
- Se cortan los archivos que están en src (jar y los csv) y se mueven fuera del directorio del proyecto, justo dentro de LabRecu2. Este paso se realiza para no tener errores de compilación ya que sino lo va a incluir dentro de paquete.

Para hacer este corte y movimiento de archivos se va a la vista de archivos dentro del Package Explorer, se le da al triangulito de arriba para seleccionar la opción Package Presentation y escoger que se muestre jerárquico: “Hierarchical”. Como en la imagen. (Fuente 5)



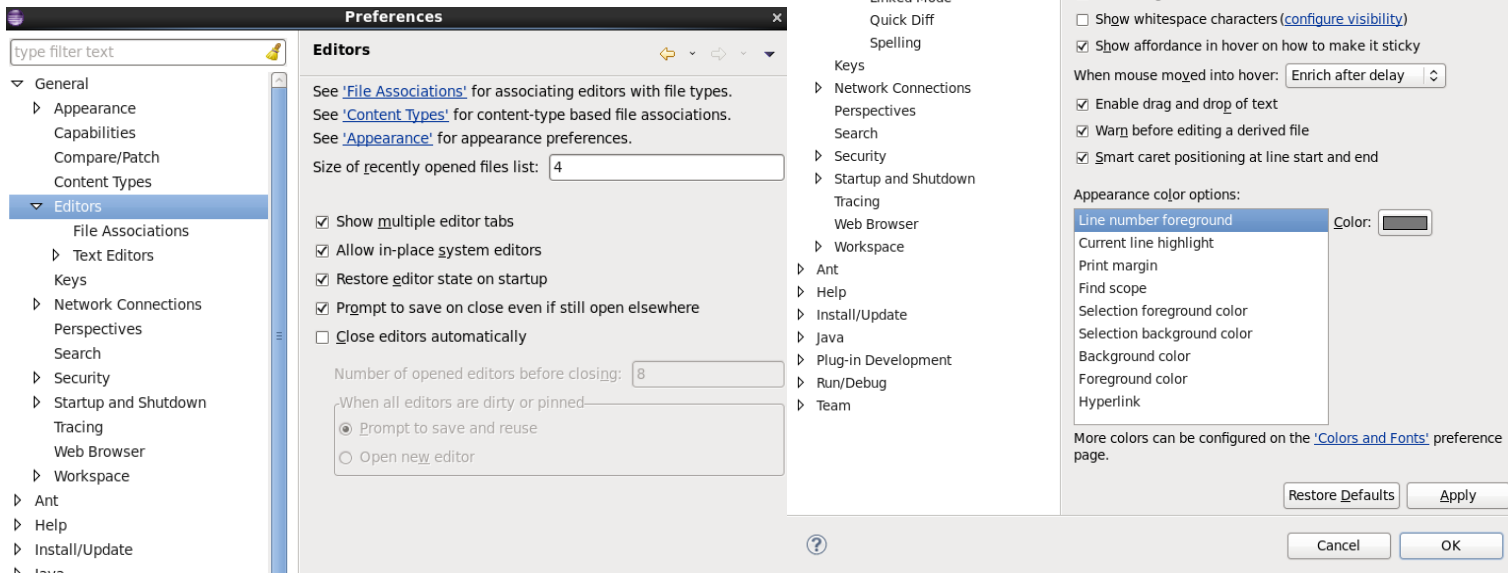


The screenshot shows the VMware Workstation interface. The title bar reads "Developer\_Hadoop - VMware Workstation 17 Player (Non-co...". The menu bar includes "Player", "Applications", "Places", "System", and "File". The "Places" menu is open, displaying "Home Folder" and "Desktop".





• Para facilitar el trabajo de lectura de código de Eclipse, se va a Window>Preferences para añadir “.” para identificar espacios, añadir “¶” para identificar saltos de línea y para cambiar tamaños de letra del código. En la imagen se ven las configuraciones conocidas como bastante óptimas teniendo en cuenta la capacidad humana de procesar líneas más cortas y identificar mejor puntos que espacios etc. (Fuente 5)



## 6.2 – Modificación en el orden en las validaciones del Driver

• En la línea 15 del Driver AnalistaPelis se identifica que `if (args.length !=3) { ... hasta System.exit(-1)` debería ir más abajo entre las líneas 35-36. Se corta esta parte del código y se sitúa dentro del `public static void main throws Exception`. Este paso se hace para realizar la validación de que el número de argumentos de entrada es el correcto en la ejecución del “main” (o sea el punto de entrada del aplicativo) en vez de hacerlo en la ejecución del “job”. De esta forma se hace la verificación de que los argumentos de entrada son los correctos ya al principio (en el main, que es lo primero que se ejecuta siempre en java). Con el cambio el código queda así: (Fuente 5)

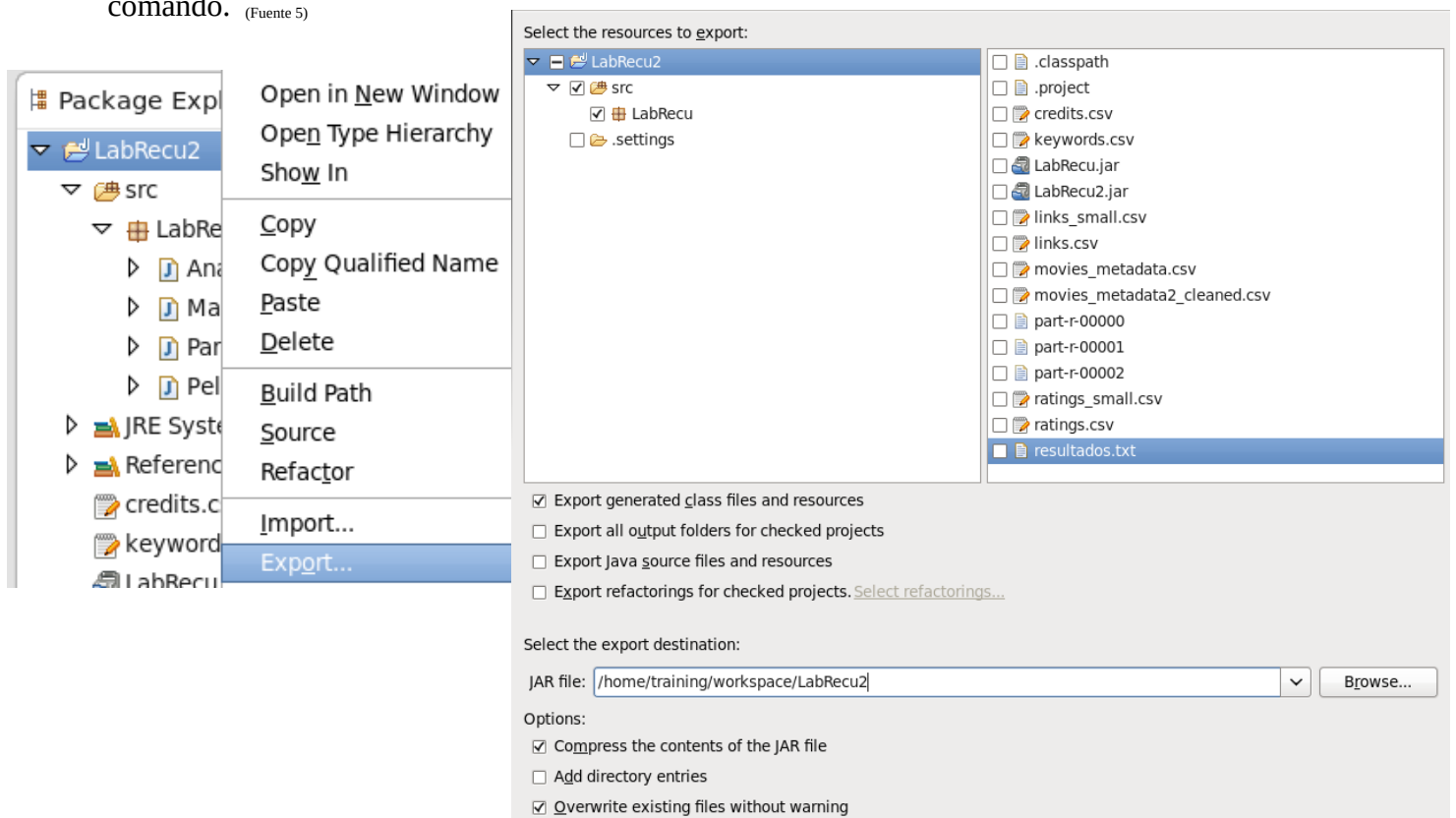
```

1 package LabRecu;
2
3 import org.apache.hadoop.conf.Configured;
4
11
12 public class AnalistaPelis extends Configured implements Tool {
13     @Override
14     public int run(String[] args) throws Exception {
15
16         Job job = Job.getInstance(getConf(), "Anàlisi de Pel·lícules");
17         job.setJarByClass(AnalistaPelis.class);
18
19         job.setMapperClass(MapejadorPelicules.class);
20         job.setReducerClass(ReducidorPelicules.class);
21         job.setPartitionerClass(ParticionadorGenere.class);
22         job.setNumReduceTasks(3);
23
24         job.setOutputKeyClass(Text.class);
25         job.setOutputValueClass(PeliculaWritable.class);
26
27         FileInputFormat.addInputPath(job, new Path(args[0]));
28         FileOutputFormat.setOutputPath(job, new Path(args[1]));
29
30         job.getConfiguration().set("revenue.threshold", args[2]);
31
32         return job.waitForCompletion(true) ? 0 : 1;
33     }
34
35     public static void main(String[] args) throws Exception {
36         if (args.length != 3) {
37             System.err.println("Ús: AnalistaPelis <input path> <output path> <revenue threshold>");
38             System.exit(-1);
39         }
40
41         int exitCode = ToolRunner.run(new AnalistaPelis(), args);
42         System.exit(exitCode);
43     }
44 }

```

## 6.3 – 1ª ejecución con falta de un input

• Para crear el jar que se va a ejecutar en hadoop, se va a Eclipse en el Package Explorer>LabRecu2>click derecho>Export...>Next y se deseleccionan los elementos de la derecha y las opciones de “Overwrite existing files without warning” y “Compress the contents of the JAR file” tal como se ve en las imágenes. Se selecciona main class y finish. Esta es una forma equivalente con la herramienta Eclipse de una forma más intuitiva respecto a ejecutar por línea de comando. (Fuente 5)



- Para ejecutar en hadoop, se escriben las rutas relativas a partir del paquete “LabRecu” según la estructura “paquete.classe”. La estructura “./” se realiza para hacer la ejecución de forma relativa al punto donde se está ejecutando. Por eso es que el código que se ejecuta es el siguiente: (Fuente 3 y 5)

```
hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata.csv ./output
```

que es un código totalmente equivalente al de referenciar explícitamente la ruta absoluta:

```
hadoop jar LabRecu2.jar LabRecu.AnalistaPelis
/home/training/workspace/LabRecu2/movies_metadata.csv
/home/training/workspace/LabRecu2/output
```

Aquí se observa el proceso de ejecución del código:

```
[training@localhost ~]$ ls
Desktop      kiji-bento-albacore-1.0.5-release.tar.gz  Pictures      src      Videos
Documents    [redacted]                               Public        Templates workspace
Downloads    lib                                       scripts       training workspace.save.dev
eclipse      Music                                   spark-1.3.1   training_materials

[training@localhost ~]$ cd workspace
[training@localhost workspace]$ ls
hive LabRecu LabRecu2 oozie-labs _pasted_code_
[training@localhost workspace]$ cd LabRecu2
[training@localhost LabRecu2]$ ls
bin      keywords.csv LabRecu.jar links_small.csv ratings.csv src
credits.csv LabRecu2.jar links.csv movies_metadata.csv ratings_small.csv

[training@localhost LabRecu2]$ pwd
/home/training/workspace/LabRecu2
[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata.csv ./output
Us: AnalistaPelis <input path> <output path> <revenue threshold>
```

Y se ve que el programa devuelve el System error de la línea 37 pidiendo un input path, un output path y un revenue threshold.

## 6.4 – 2ª ejecución con falta de archivo de input

- Se ajusta la primera ejecución redactando el threshold que faltaba en este caso de 5000000.

```
[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata.csv ./output 5000000
```

```
[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata.csv ./output 5000000
24/07/13 06:00:25 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/07/13 06:00:25 INFO mapred.JobClient: Cleaning up the staging area hdfs://0.0.0.0:8020/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/job_202407130519_0001
24/07/13 06:00:25 ERROR security.UserGroupInformation: PrivilegedActionException as:training (auth:SIMPLE) cause:org.apache.hadoop.mapreduce.lib.input.InvalidInputException: Input path does not exist: hdfs://0.0.0.0:8020/user/training/movies_metadata.csv
Exception in thread "main" org.apache.hadoop.mapreduce.lib.input.InvalidInputException: Input path does not exist: hdfs://0.0.0.0:8020/user/training/movies_metadata.csv
    at org.apache.hadoop.mapreduce.lib.input.FileInputFormat.listStatus(FileInputFormat.java:231)
    at org.apache.hadoop.mapreduce.lib.input.FileInputFormat.getSplits(FileInputFormat.java:248)
    at org.apache.hadoop.mapred.JobClient.writeNewSplits(JobClient.java:1064)
    at org.apache.hadoop.mapred.JobClient.writeSplits(JobClient.java:1081)
    at org.apache.hadoop.mapred.JobClient.access$600(JobClient.java:174)
    at org.apache.hadoop.mapred.JobClient$2.run(JobClient.java:993)
    at org.apache.hadoop.mapred.JobClient$2.run(JobClient.java:946)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.JobClient.submitJobInternal(JobClient.java:946)
    at org.apache.hadoop.mapreduce.Job.submit(Job.java:566)
    at org.apache.hadoop.mapreduce.Job.waitForCompletion(Job.java:596)
    at LabRecu.AnalistaPelis.run(AnalistaPelis.java:32)
    at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
    at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:84)
    at LabRecu.AnalistaPelis.main(AnalistaPelis.java:41)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
    at java.lang.reflect.Method.invoke(Method.java:597)
    at org.apache.hadoop.util.RunJar.main(RunJar.java:208)
```

El programa devuelve un InvalidInputException indicando que no encuentra el archivo *movies\_metadata.csv* en la ruta /user/training donde la está buscando. Por esa razón se debe poner el *movies\_metadata.csv* dentro de la ruta hadoop file system con el comando de copia *-put* y comprobar que se ha copiado correctamente con *-ls*. (Fuente 3)

```
[training@localhost LabRecu2]$ hadoop fs -put movies_metadata.csv /user/training
[training@localhost LabRecu2]$ hadoop fs -ls /user/training
Found 1 items
-rw-r--r-- 1 training supergroup 34445126 2024-07-13 06:04 /user/training/movies_metadata.csv
[training@localhost LabRecu2]$
```

## 6.5 – 3ª ejecución procesando datos con formato no esperado

- Ahora, con el archivo faltante ya puesto a sitio, se vuelve a ejecutar el job de hadoop.

```
[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis "/movies_metadata.csv ./output 5000000
24/07/13 06:06:19 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/07/13 06:06:20 INFO input.FileInputFormat: Total input paths to process : 1
24/07/13 06:06:22 INFO mapred.JobClient: Running job: job_202407130519_0002
24/07/13 06:06:23 INFO mapred.JobClient: map 0% reduce 0%
24/07/13 06:06:43 INFO mapred.JobClient: Task Id : attempt_202407130519_0002_m_000000_0, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:23)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)
```

Al ver “ArrayIndexOutOfBoundsException” se identifica que el programa falló debido a que está tratando de acceder a un elemento de una matriz que no existe. Igualmente, se deja que el programa termine. (Fuente 5)

```
24/07/13 06:06:19 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/07/13 06:06:20 INFO input.FileInputFormat: Total input paths to process : 1
24/07/13 06:06:22 INFO mapred.JobClient: Running job: job_202407130519_0002
24/07/13 06:06:23 INFO mapred.JobClient: map 0% reduce 0%
24/07/13 06:06:43 INFO mapred.JobClient: Task Id : attempt_202407130519_0002_m_000000_0, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:23)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)

24/07/13 06:06:53 INFO mapred.JobClient: Task Id : attempt_202407130519_0002_m_000000_1, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:23)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)

24/07/13 06:07:03 INFO mapred.JobClient: Task Id : attempt_202407130519_0002_m_000000_2, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:23)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)

24/07/13 06:07:18 INFO mapred.JobClient: Job complete: job_202407130519_0002
24/07/13 06:07:18 INFO mapred.JobClient: Counters: 7
24/07/13 06:07:18 INFO mapred.JobClient:   Job Counters
24/07/13 06:07:18 INFO mapred.JobClient:     Failed map tasks=1
24/07/13 06:07:18 INFO mapred.JobClient:     Launched map tasks=4
24/07/13 06:07:18 INFO mapred.JobClient:     Data-local map tasks=4
24/07/13 06:07:18 INFO mapred.JobClient:     Total time spent by all maps in occupied slots (ms)=50510
24/07/13 06:07:18 INFO mapred.JobClient:     Total time spent by all reduces in occupied slots (ms)=0
24/07/13 06:07:18 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving slots (ms)=0
24/07/13 06:07:18 INFO mapred.JobClient:     Total time spent by all reduces waiting after reserving slots (ms)=0
```

Se identifica que falló en la línea 23 del MapejadorReduidor.java. Esta línea inicialmente era:

```
protected void map(LongWritable clau, Text valor, Context context) throws IOException, InterruptedException {
    ... String[] camps = valor.toString().split(";");
    ... String titol = camps[1];
    ... String generes = camps[2];
    ... double valoracio = Double.parseDouble(camps[3]);
    ... double ingressos = Double.parseDouble(camps[4]); ...
}
```

- Se realizan cambios para añadir condicionantes que evitan que el programa procese datos que no tienen estructura de 7 campos, datos de ingresos y valoraciones no numéricos o datos NaN. De esta forma esta línea de código queda modificada tal que así: (Fuente 5)

```
protected void map(LongWritable clau, Text valor, Context context) throws IOException, InterruptedException {
    ... String[] camps = valor.toString().split(";");
    ... // Si l'estructura té 7 camps, procesa. Sinó és un error i s'ignora (no es processa) aquest camp.
    ... // Si a la conversió de números (si el camp 3 i el 4) són números i no ha fallat, procesa.
    ... // Sinó és que ha fallat per exemple si fos un nan i s'ignora (no es processa) aquest camp.
    ... if (camps.length == 7 && NumberUtils.isNumber(camps[4]) && NumberUtils.isNumber(camps[2])) {
    ... >>
    ... >> String titol = camps[1];
    ... >> String genere = camps[2];
    ... >> double valoracio = Double.parseDouble(camps[3]);
    ... >> double ingressos = Double.parseDouble(camps[4]);
    ... } else {
    ... >> System.out.println("El format no és correcte: " + valor.toString());
    ... }
```

Lo que hace concretamente es coger el “valor” de entrada. Pasarlo a string i separarlo por ;. Después verifica los casos nombrados y si se cumplen las condiciones, se asigna a las variables titol, genere, valoració e ingressos los datos de interés a coger. En caso que no se verifiquen las condiciones (else) el sistema sale y imprime mensaje diciendo que el formato no es correcto, concatenando seguidamente el valor de entrada dado en formato de caracteres. (Fuente 5)

## 6.6 – 4ª ejecución sin previo borrado del directorio de salida

- Con los cambios del punto anterior se ejecuta de nuevo el job de hadoop.

```
[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata.csv ./output 5000000
24/07/13 06:12:16 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/07/13 06:12:17 INFO mapred.JobClient: Cleaning up the staging area hdfs://0.0.0.0:8020/var/lib/hadoop-hdfs/cache/mapred/mapred/staging/training/.staging/job_202407130519_0003
24/07/13 06:12:17 ERROR security.UserGroupInformation: PrivilegedActionException as:training (auth:SIMPLE) cause:org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory output already exists
Exception in thread "main" org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory output already exists
    at org.apache.hadoop.mapreduce.lib.output.FileOutputFormat.checkOutputSpecs(FileOutputFormat.java:132)
    at org.apache.hadoop.mapred.JobClient$2.run(JobClient.java:985)
    at org.apache.hadoop.mapred.JobClient$2.run(JobClient.java:946)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.JobClient.submitJobInternal(JobClient.java:946)
    at org.apache.hadoop.mapreduce.Job.submit(Job.java:566)
    at org.apache.hadoop.mapreduce.Job.waitForCompletion(Job.java:596)
    at LabRecu.AnalistaPelis.run(AnalistaPelis.java:32)
    at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:70)
    at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:84)
    at LabRecu.AnalistaPelis.main(AnalistaPelis.java:41)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
    at java.lang.reflect.Method.invoke(Method.java:597)
    at org.apache.hadoop.util.RunJar.main(RunJar.java:208)
[training@localhost LabRecu2]$
```

El job falló porque nos indica que “FileAlreadyExistsException: Output directory output already exists” esto sucede porque para cada job de hadoop automáticamente se crea la carpeta output para el resultado y debido a que anteriormente ya creó esta carpeta, es necesario borrarla antes de seguir con la ejecución porque sino falla.

- Lo que se debe hacer entonces es ir a borrar la carpeta output. A través de la línea de comandos se hace el borrado. Se hacen varios intentos de borrado (-rf, -f, -r) hasta que se encuentra la opción que borra correctamente el directorio. (Fuente 3)

```
[training@localhost LabRecu2]$ hadoop fs -rm -rf /user/training/output
-rm: Illegal option -rf
Usage: hadoop fs [generic options] -rm [-f] [-r|-R] [-skipTrash] <src> ...
[training@localhost LabRecu2]$ hadoop fs -rm -f /user/training/output
rm: `/user/training/output': Is a directory
[training@localhost LabRecu2]$ hadoop fs -rm -r /user/training/output
Deleted /user/training/output
```



## 6.7 – 5ª ejecución con campo género en formato json

Tras el borrado de la carpeta output, se vuelve a ejecutar el job.

```
[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata.csv ./output 5000000
24/07/13 06:15:47 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/07/13 06:15:47 INFO input.FileInputFormat: Total input paths to process : 1
24/07/13 06:15:48 INFO mapred.JobClient: Running job: job_202407130519_0004
24/07/13 06:15:49 INFO mapred.JobClient: map 0% reduce 0%
24/07/13 06:16:04 INFO mapred.JobClient: Task Id : attempt_202407130519_0004_m_000000_0, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:24)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)

24/07/13 06:16:17 INFO mapred.JobClient: Task Id : attempt_202407130519_0004_m_000000_1, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:24)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)

24/07/13 06:16:28 INFO mapred.JobClient: Task Id : attempt_202407130519_0004_m_000000_2, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:24)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)
```

Falló tres veces en cada uno de los tres particionadores en un mismo error. Se identifica que el origen del fallo es en la línea 24 del MapejadorReduidor.java. Esta línea inicialmente era:

```
String[] camps = valor.toString().split(";");
```

Se modifica para que la separación sea por , en vez de por ; ya que csv particiona los campos por ,

```
String[] camps = valor.toString().split(",");
```

A parte de este error también hay que tener en cuenta se tienen otros problemas. Por una parte hay el problema de que hay comas dentro de los campos del csv. Por otra parte hay el problema de que una película está identificada con varios géneros a la vez por lo que el programa puede confundirse a la hora de particionarse por género. Para solucionar este asunto se escribe un programa auxiliar en Python adjunto al trabajo “Python\_LabRecu.ipynb”. (Elaboración propia)

- El programa en Python procesa la columna de géneros del dataframe generado con el csv cuyos elementos están en formato json array y crea otra columna seleccionando de forma aleatoria un género de las películas. De esta forma ese genero se identifica como si esa película fuese únicamente de ese género. También tiene en cuenta el caso de las películas sin genero. La idea de hacer la eliminación aleatoria y no hacerla

GenereAleatori	
Drama	11042
Comedy	7343
Documentary	3297
Thriller	3050
Romance	2725
SenseGenere	2539
Action	2427
Horror	2389
Crime	1550
Adventure	1203
Science Fiction	1154
Family	1093
Animation	860
Mystery	851
Fantasy	769
Music	707
Foreign	619
Western	605
War	541
History	501
TV Movie	295
LStar Capital	1
Odyssey Media	1
Fides Films	1
Mwana Productions	1
Sentai Filmworks	1
Vision View Entertainment	1
TÃ©lÃ©vision Suisse-Romande (TSR)	1
WhiteFlame Productions	1
The Mirisch Corporation	1
Serenade Films	1
First Motion Pictures Unit US Army Air Forces	1
F Comme Film	1
Name: count, dtype: int64	

ordenada quedándonos con el primer género que aparece en la lista de géneros es evitar que los géneros alfabéticamente superiores queden sobrerrepresentados respecto a los últimos del alfabeto. Todo el código de puede consultar adjunto al trabajo. El resultado son las siguientes clasificaciones en la imagen a derecha. (Elaboración propia)

A su vez el programa modifica los ingresos que no son numéricos para que sean 0 así como identifica el máximo de ingresos.

```
1 df["revenue"] = df["revenue"].replace({"False": 0.0})
2 df["revenue"] = pd.to_numeric(df["revenue"], errors='coerce').fillna(0)
3 df["revenue"].max()
```

2787965087.0

El nuevo archivo modificado se guarda como movies\_metadata2\_cleaned.

- Se descarga, sitúa el movies\_metadata2\_cleaned en LabRecu2 y se copia en /user/training.

```
[training@localhost LabRecu2]$ hadoop fs -put movies_metadata2_cleaned.csv /user/training/
[training@localhost LabRecu2]$ hadoop fs -ls
^[[Found 3 items
-rw-r--r--  1 training supergroup  34445126 2024-07-13 06:04 movies_metadata.csv
-rw-r--r--  1 training supergroup   2695322 2024-07-13 07:29 movies_metadata2_cleaned.csv
drwxr-xr-x  - training supergroup      0 2024-07-13 06:16 output
```

## 6.8 – 6ª ejecución con columnas asignadas erróneamente y bucle for innecesario

- Ahora se ejecuta el job con el nuevo archivo de input movies\_metadata2\_cleaned. El output se cambia también a output\_cleaned para diferenciarlo y para no tener que borrar la carpeta ya creada de output que saltaría error.

```
[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata2_cleaned.csv ./output_cleaned 5000000
```

```
[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata2_cleaned.csv ./output_cleaned 5000000
24/07/13 07:31:11 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/07/13 07:31:12 INFO input.FileInputFormat: Total input paths to process : 1
24/07/13 07:31:13 INFO mapred.JobClient: Running job: job_202407130519_0005
24/07/13 07:31:14 INFO mapred.JobClient: map 0% reduce 0%
24/07/13 07:31:35 INFO mapred.JobClient: Task Id : attempt_202407130519_0005_m_000000_0, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:24)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)

24/07/13 07:31:44 INFO mapred.JobClient: Task Id : attempt_202407130519_0005_m_000000_1, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:24)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)

24/07/13 07:31:52 INFO mapred.JobClient: Task Id : attempt_202407130519_0005_m_000000_2, Status : FAILED
java.lang.ArrayIndexOutOfBoundsException: 1
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:24)
    at LabRecu.MapejadorPelicules.map(MapejadorReduidor.java:1)
    at org.apache.hadoop.mapreduce.Mapper.run(Mapper.java:140)
    at org.apache.hadoop.mapred.MapTask.runNewMapper(MapTask.java:673)
    at org.apache.hadoop.mapred.MapTask.run(MapTask.java:331)
    at org.apache.hadoop.mapred.Child$4.run(Child.java:268)
    at java.security.AccessController.doPrivileged(Native Method)
    at javax.security.auth.Subject.doAs(Subject.java:396)
    at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1408)
    at org.apache.hadoop.mapred.Child.main(Child.java:262)
```

El programa falló porqué la columna del género quedó situada al final. Se modifica el número final de columna del género para que cuadre con la que es, que es la 6. A su vez se mura que cuadren todas las columnas según la posición donde están. En orden las columnas son:

`budget,original_title,revenue,title,vote_average,vote_count,GenereAleatori`

Por el motivo de cuadrar las columnas el código de MapejadorReduidor se modifica así:

```
protected void map(LongWritable clau, Text valor, Context context) throws IOException, InterruptedException {
    String[] camps = valor.toString().split(",");
    // En el csv hi ha aquesta estructura de les columnes: budget,original_title,revenue,title,vote_average,vote_count,GenereAleatori
    // Per això escollim 1 com el camp de títol, 6 com el camp de gènere..
    // Si l'estructura té 7 camps, procesa. Sinó és un error i s'ignora (no es processa) aquest camp.
    // Si a la conversió de números (si el camp 4 i el 2) són números i no ha fallat, procesa.
    // Sinó és que ha fallat per exemple si fos un nan) i s'ignora (no es processa) aquest camp.
    if (camps.length == 7 && NumberUtils.isNumber(camps[4]) && NumberUtils.isNumber(camps[2])) {
        String titol = camps[1];
        String genere = camps[6];
        double valoracio = Double.parseDouble(camps[4]);
        double ingressos = Double.parseDouble(camps[2]);
        if (ingressos > llindarIngressos) {
            for (String genere : genres.split("\\|")) {
                context.write(new Text(genere), new PeliculaWritable(titol, valoracio, ingressos));
            }
        } else {
            System.out.println("El format no és correcte: " + valor.toString());
        }
    }
}
```

• Por lo que respecta al bucle *for* que hay en la condición de que los ingresos sean mayores al umbral de ingresos ya no sirve porque ya no es necesario escribir tantas veces como géneros haya ya que género es un solo campo y no hace falta iterar. De esta forma, al suprimir el bucle *for*, el Mapeador queda finalmente así: (Fuente 5)



```

.....protected void map(LongWritable clau, Text valor, Context context) throws IOException, InterruptedException {
.....String[] camps = valor.toString().split(",");
.....// En el csv hi ha aquesta estructura de les columnes: budget,original title,revenue,title,vote_average,vote_count,GenreAleatori
.....// Per això escollim 1 com el camp de títol, 6 com el camp de gènere...
.....// Si l'estructura té 7 camps, procesa. Sinó és un error i s'ignora (no es processa) aquest camp.
.....// Si a la conversió de números (si el camp 4 i el 2) són números i no ha fallat, procesa.
.....// Sinó és que ha fallat per exemple si fos un nan i s'ignora (no es processa) aquest camp.
.....if (camps.length == 7 && NumberUtils.isNumber(camps[4]) && NumberUtils.isNumber(camps[2])) {
.....    String titol = camps[1];
.....    String genere = camps[6];
.....    double valoracio = Double.parseDouble(camps[4]);
.....    double ingressos = Double.parseDouble(camps[2]);
.....    if (ingressos > 0) {
.....        context.write(new Text(genere), new PeliculaWritable(titol, valoracio, ingressos));
.....    }
.....} else {
.....    System.out.println("El format no és correcte: " + valor.toString());
.....}
}

```

Y el reducer suma calculando la media de las valoraciones de las películas de un mismo género:

```

class ReduidorPeliculas extends Reducer<Text, PeliculaWritable, Text, Text> {
.....@Override
.....protected void reduce(Text clau, Iterable<PeliculaWritable> valors, Context context) throws IOException, InterruptedException {
.....    int comptador = 0;
.....    double valoracioTotal = 0.0;
.....    for (PeliculaWritable valor : valors) {
.....        valoracioTotal += valor.obtenirValoracio().get();
.....        comptador++;
.....    }
.....    double valoracioMitjana = valoracioTotal / comptador;
.....    context.write(clau, new Text("Valoració promig: " + valoracioMitjana));
.....}
}

```

## 6.9 – 7ª ejecución. Ejecución correcta.

- Con todas estas modificaciones de código se vuelve a crear el jar que se va a ejecutar en hadoop, se va a Eclipse en el Package Explorer>LabRecu2>click derecho>Export...>Next y se deseleccionan los elementos de la derecha y las opciones de “Overwrite existing files without warning” y “Compress the contents of the JAR file” tal como se ha hecho anteriormente. Se selecciona main class y finish. (Fuente 5)

- Se borran los directorios outputs anteriores.

```

[training@localhost LabRecu2]$ hadoop fs -rm -r /user/training/output
Deleted /user/training/output
[training@localhost LabRecu2]$ hadoop fs -rm -r /user/training/output_cleaned
Deleted /user/training/output_cleaned

```

- Se ejecuta de nuevo el código

```

[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata2_cleaned.csv ./output_cleaned 5000000

```

```

[training@localhost LabRecu2]$ hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata2_cleaned.csv ./output_cleaned 5000000
24/07/13 08:04:41 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should implement Tool for the same.
24/07/13 08:04:42 INFO input.FileInputFormat: Total input paths to process : 1
24/07/13 08:04:43 INFO mapred.JobClient: Running job: job_202407130519_0006
24/07/13 08:04:45 INFO mapred.JobClient: map 0% reduce 0%
24/07/13 08:05:10 INFO mapred.JobClient: map 100% reduce 0%
24/07/13 08:05:27 INFO mapred.JobClient: map 100% reduce 66%
24/07/13 08:05:34 INFO mapred.JobClient: map 100% reduce 100%
24/07/13 08:05:39 INFO mapred.JobClient: Job complete: job_202407130519_0006
24/07/13 08:05:39 INFO mapred.JobClient: Counters: 32
24/07/13 08:05:39 INFO mapred.JobClient: File System Counters
24/07/13 08:05:39 INFO mapred.JobClient: FILE: Number of bytes read=213678
24/07/13 08:05:39 INFO mapred.JobClient: FILE: Number of bytes written=1196752
24/07/13 08:05:39 INFO mapred.JobClient: FILE: Number of read operations=0
24/07/13 08:05:39 INFO mapred.JobClient: FILE: Number of large read operations=0
24/07/13 08:05:39 INFO mapred.JobClient: FILE: Number of write operations=0
24/07/13 08:05:39 INFO mapred.JobClient: HDFS: Number of bytes read=2695449
24/07/13 08:05:39 INFO mapred.JobClient: HDFS: Number of bytes written=910
24/07/13 08:05:39 INFO mapred.JobClient: HDFS: Number of read operations=4
24/07/13 08:05:39 INFO mapred.JobClient: HDFS: Number of large read operations=0
24/07/13 08:05:39 INFO mapred.JobClient: HDFS: Number of write operations=3
24/07/13 08:05:39 INFO mapred.JobClient: Job Counters
24/07/13 08:05:39 INFO mapred.JobClient: Launched map tasks=1
24/07/13 08:05:39 INFO mapred.JobClient: Launched reduce tasks=3
24/07/13 08:05:39 INFO mapred.JobClient: Data-local map tasks=1
24/07/13 08:05:39 INFO mapred.JobClient: Total time spent by all maps in occupied slots (ms)=25766
24/07/13 08:05:39 INFO mapred.JobClient: Total time spent by all reduces in occupied slots (ms)=36385
24/07/13 08:05:39 INFO mapred.JobClient: Total time spent by all maps waiting after reserving slots (ms)=0
24/07/13 08:05:39 INFO mapred.JobClient: Total time spent by all reduces waiting after reserving slots (ms)=0
24/07/13 08:05:39 INFO mapred.JobClient: Map-Reduce Framework
24/07/13 08:05:39 INFO mapred.JobClient: Map input records=45573
24/07/13 08:05:39 INFO mapred.JobClient: Map output records=4959
24/07/13 08:05:39 INFO mapred.JobClient: Map output bytes=203732
24/07/13 08:05:39 INFO mapred.JobClient: Input split bytes=127
24/07/13 08:05:39 INFO mapred.JobClient: Combine input records=0
24/07/13 08:05:39 INFO mapred.JobClient: Combine output records=0
24/07/13 08:05:39 INFO mapred.JobClient: Reduce input groups=20
24/07/13 08:05:39 INFO mapred.JobClient: Reduce shuffle bytes=213678
24/07/13 08:05:39 INFO mapred.JobClient: Reduce input records=4959
24/07/13 08:05:39 INFO mapred.JobClient: Reduce output records=20
24/07/13 08:05:39 INFO mapred.JobClient: Spilled Records=9918
24/07/13 08:05:39 INFO mapred.JobClient: CPU time spent (ms)=7430
24/07/13 08:05:39 INFO mapred.JobClient: Physical memory (bytes) snapshot=480878592
24/07/13 08:05:39 INFO mapred.JobClient: Virtual memory (bytes) snapshot=2909040640
24/07/13 08:05:39 INFO mapred.JobClient: Total committed heap usage (bytes)=163123200
[training@localhost LabRecu2]$

```

se observa que la ejecución se ha realizado correctamente sin errores.

## 6.10 – Visualización y agrupación de los resultados

- Se comprueban los outputs que se han generado: El archivo success, el archivo de los logs y los archivos de los tres reducers.

```

[training@localhost LabRecu2]$ hadoop fs -ls /user/training/output_cleaned
Found 5 items
-rw-r--r-- 1 training supergroup 0 2024-07-13 08:05 /user/training/output_cleaned/_SUCCESS
drwxr-xr-x - training supergroup 0 2024-07-13 08:04 /user/training/output_cleaned/_logs
-rw-r--r-- 1 training supergroup 358 2024-07-13 08:05 /user/training/output_cleaned/part-r-00000
-rw-r--r-- 1 training supergroup 370 2024-07-13 08:05 /user/training/output_cleaned/part-r-00001
-rw-r--r-- 1 training supergroup 182 2024-07-13 08:05 /user/training/output_cleaned/part-r-00002

```

- Se abre el archivo el primer reducer de tres y se observa que los resultados son satisfactorios y son lo que se esperaba. Porque aparece el género seguido de la valoración promedio de ese género. Se indica la opción *less* para se pueda desplazar por las flechas después de mostrar el *output*. Para cambiar la pantalla de muestra de resultados a la pantalla de código se usa “q” de *quit*. (Fuente 5)

```
hadoop fs -cat /user/training/output_cleaned/part-r-00000 | less
```

---

Action	Valoració promig:	6.154176072234763
Animation	Valoració promig:	6.473529411764705
Comedy	Valoració promig:	6.0373467112597545
Fantasy	Valoració promig:	6.184285714285714
Foreign	Valoració promig:	5.699999999999999
Music	Valoració promig:	6.533333333333332
Mystery	Valoració promig:	6.419834710743805
Romance	Valoració promig:	6.31260273972602

- Se traen los tres archivos que se han generado a través del comando -get. (Fuente 5)

```
hadoop fs -get /user/training/output_cleaned/part-r-00000
hadoop fs -get /user/training/output_cleaned/part-r-00001
hadoop fs -get /user/training/output_cleaned/part-r-00002
```

Y se comprueba que realmente se han traído en LabRecu2. Se puede ver todo lo que hay en LabRecu2 con `ls` o con `ls -al`

```
[training@localhost LabRecu2]$ ls
bin          LabRecu2.jar  links_small.csv  part-r-00000  ratings.csv
credits.csv  LabRecu.jar   movies_metadata2_cleaned.csv  part-r-00001  ratings_small.csv
keywords.csv links.csv      movies_metadata.csv  part-r-00002  src

[training@localhost LabRecu2]$ ls -al
total 924344
drwxrwxr-x 5 training training 4096 Jul 13 08:13 .
drwxrwxr-x 8 training training 4096 Jul 10 14:41 ..
drwxrwxr-x 3 training training 4096 Jul 13 05:27 bin
-rw-rw-r-- 1 training training 3632 Jul 10 15:09 .classpath
-rwxrw-rw- 1 training training 189917659 Sep 21 2019 credits.csv
-rwxrw-rw- 1 training training 6231943 Sep 21 2019 keywords.csv
-rw-rw-r-- 1 training training 5827 Jul 13 08:02 LabRecu2.jar
-rw-rw-r-- 1 training training 3990 Jul 12 14:03 LabRecu.jar
-rwxrw-rw- 1 training training 989107 Sep 21 2019 links.csv
-rwxrw-rw- 1 training training 183372 Sep 21 2019 links_small.csv
-rwxrw-rw- 1 training training 2695322 Jul 13 07:26 movies_metadata2_cleaned.csv
-rwxrw-rw- 1 training training 34445126 Sep 21 2019 movies_metadata.csv
-rwxr-xr-x 1 training training 358 Jul 13 08:12 part-r-00000
-rwxr-xr-x 1 training training 370 Jul 13 08:13 part-r-00001
-rwxr-xr-x 1 training training 182 Jul 13 08:12 part-r-00002
-rw-rw-r-- 1 training training 367 Jul 10 14:41 .project
-rwxrw-rw- 1 training training 709550327 Sep 21 2019 ratings.csv
-rwxrw-rw- 1 training training 2438266 Sep 21 2019 ratings_small.csv
drwxrwxr-x 2 training training 4096 Jul 10 14:41 .settings
drwxrwxr-x 3 training training 4096 Jul 13 05:26 src
```

O también se puede ver las tres partes reducidas “part” que hay en concreto con `ls -al part*`

```
[training@localhost LabRecu2]$ ls -al part*
-rwxr-xr-x 1 training training 358 Jul 13 08:12 part-r-00000
-rwxr-xr-x 1 training training 370 Jul 13 08:13 part-r-00001
-rwxr-xr-x 1 training training 182 Jul 13 08:12 part-r-00002
```

Equivalentemente los archivos también se pueden abrir con Geany con Eclipse en la opción Open with>Geany (Fuente 5)



Si se abren los 3 se pueden consultar los resultados finales correspondientes a los 20 géneros que cumplen que tienen películas con un límite de ingresos de 5000000

part-r-00000 X	part-r-00001 X	part-r-00002 X
1	Action	Valoració promig: 6.154176072234763
2	Animation	Valoració promig: 6.473529411764705
3	Comedy	Valoració promig: 6.0373467112597545
4	Fantasy	Valoració promig: 6.184285714285714
5	Foreign	Valoració promig: 5.699999999999999
6	Music	Valoració promig: 6.533333333333332
7	Mystery	Valoració promig: 6.419834710743805
8	Romance	Valoració promig: 6.31260273972602
9		

part-r-00000 X	part-r-00001 X	part-r-00002 X
1	Crime	Valoració promig: 6.403690036900371
2	Documentary	Valoració promig: 6.807142857142857
3	Family	Valoració promig: 6.224468085106386
4	Horror	Valoració promig: 5.823788546255501
5	Science Fiction	Valoració promig: 6.192222222222224
6	SenseGenre	Valoració promig: 3.8333333333333335
7	War	Valoració promig: 6.898275862068961
8	Western	Valoració promig: 6.7524999999999995
9		

part-r-00000 X	part-r-00001 X	part-r-00002 X
1	Adventure	Valoració promig: 6.153287197231837
2	Drama	Valoració promig: 6.647194388777546
3	History	Valoració promig: 6.8500000000000005
4	Thriller	Valoració promig: 6.205831533477317
5		

- Como último paso se va a convertir estos tres archivos a uno solo. Se limpia la consola con Cntrl+L. Con el comando `touch` se crea un `.txt` llamado “resultados” vacío. Con el comando `cat` se leen los tres archivos generados y se inyectan en el `resultados.txt`. Finalmente con el comando `less` se lee el archivo `resultados.txt`. (Fuente 5)

```
[training@localhost LabRecu2]$ touch resultados.txt
[training@localhost LabRecu2]$ ls
bin          LabRecu2.jar  links_small.csv  part-r-00000  ratings.csv    src
credits.csv  LabRecu2.jar  movies_metadata2_cleaned.csv  part-r-00001  ratings_small.csv
keywords.csv links.csv      movies_metadata.csv  part-r-00002  resultados.txt
[training@localhost LabRecu2]$ cat part-r-00000 part-r-00001 part-r-00002 > resultados.txt
[training@localhost LabRecu2]$ less resultados.txt
[training@localhost LabRecu2]$
```

```

Action Valoració promig: 6.154176072234763
Animation Valoració promig: 6.473529411764705
Comedy Valoració promig: 6.0373467112597545
Fantasy Valoració promig: 6.184285714285714
Foreign Valoració promig: 5.699999999999999
Music Valoració promig: 6.533333333333332
Mystery Valoració promig: 6.419834710743805
Romance Valoració promig: 6.31260273972602
Crime Valoració promig: 6.403690036900371
Documentary Valoració promig: 6.807142857142857
Family Valoració promig: 6.224468085106386
Horror Valoració promig: 5.823788546255501
Science Fiction Valoració promig: 6.192222222222224
SenseGenere Valoració promig: 3.833333333333333
War Valoració promig: 6.898275862068961
Western Valoració promig: 6.752499999999999
Adventure Valoració promig: 6.153287197231837
Drama Valoració promig: 6.647194388777546
History Valoració promig: 6.8500000000000005
Thriller Valoració promig: 6.205831533477317
resultados.txt (END)

```

## 6.11 – Edge cases

Como “Edge” cases o casos extremos de ingresos se piensan, analizan y consideran que son:

**1) Entrada con un valor negativo. Menores o iguales a 0** → El resultado esperado es que devuelva todos los géneros con su promedio ya que todos los ingresos son mayores a un valor negativo.

```

hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata2_cleaned.csv
./edgecase1 -1

```

**2) Todos los que tengan valor mayor a 0** → El resultado esperado es que devuelva todos los géneros con su promedio ya que todos los ingresos son 0 o mayores.

```

hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata2_cleaned.csv
./edgecase2 0

```

**3) Entre 0 y el mayor** → El resultado esperado es que devuelva todos los géneros que tengan ingreso x dado entre 0 y el mayor con su promedio.

```

hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata2_cleaned.csv ./
edgecase3 5000000

```

**4) El mayor** → El resultado esperado es que ningún género sea devuelto, o sea una lista vacía. Porque no hay ninguna película de mayor ingreso que la película que tiene más ingreso posible.

```

hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata2_cleaned.csv
./edgecase4 2787965087

```

**5) Mayores al mayor** → El resultado esperado es que ningún género sea devuelto, o sea una lista vacía. Porque no hay ninguna película de mayor ingreso que la película que tiene más ingreso posible.

```

hadoop jar LabRecu2.jar LabRecu.AnalistaPelis ./movies_metadata2_cleaned.csv
./edgecase5 2800000000

```

Se recuerda que el máximo de ingresos es 2 787 965 087.0 , número encontrado con el programa auxiliar en Python.

## 7– Fuentes de información y consulta

- 1- Fuente totalmente sacada de práctica 9 con co-autores Albert Ripoll y José David Angulo. (Punto del trabajo de instalación de Spark).
- 2- Apuntes de clase de Laura Mestres.
- 3- Presentación “2-Apache Hadoop Core part 2 Map Reduce” de clase de Laura Mestres.  
ToolRunner = diapos 58 a 63. Partitioner = diapos 80 a 89.
- 4- Herramienta online de búsqueda.
- 5- Ayuda de estudiante Edwin (6h llamada).
- 6- Ayuda de Data Scientist @ Bostin Consulting Group Alex Llobet (3h llamada).
- 7- Fuente sacada de práctica 4 con co-autores Albert Ripoll y José David Angulo.