# Lab 5 – Apache Spark – Java Guide

CC5212-1 – April 9, 2025

Esta es una guía opcional para ayudarte a programar tu solución en Java. Debes comenzar copiando la clase llamada `AverageSeriesRating.java` en el mismo package con el nombre `InfoSeriesRating.java`. El código proporcionado en `AverageSeriesRating.java` calcula el average rating de los episodios de una series, el cual debes extender para incluir información sobre los nombres de los episodes mejor evaluados y su rating. Puedes extender el código dado con los siguientes snippets para completar el lab. Los snippets pueden estar desordenados pero son suficientes para completar el lab. Ten en cuenta que:

- **Deberás cambiar los nombres de `rdd1`, `rdd2` y `rdd3`** (los nuevos nombres pueden variar cada vez; algunos nombres de RDD podrían ya aparecer en el código de `AverageSeriesRating.java`).
- Es posible que necesites modificar partes del código que ya se te proporcionaron en `AverageSeriesRating.java`.
- También podrías necesitar cambiar el nombre del RDD que guardas en HDFS hacia el final del código.
- Donde veas un signo de interrogación "**?**", esto indica que debes rellenar ese espacio con código.

**Cache:** Esto almacena en caché el RDD `rdd2` como `rdd1` para poder usarlo múltiples veces sin tener que recomputarlo desde cero cada vez. A partir de ahora, debes referirte a `rdd1` en lugar de `rdd2` para evitar esta recomputación. Ten en cuenta que esto podría requerir modificar código ya existente en `AverageSeriesRating.java`.

```
JavaRDD<Tuple3<String, String, Double>> rdd1 = rdd2.cache();
```

**Map-to-pair:** Esto crea un `PairRDD` `rdd1` a partir de `rdd2`.

```
JavaPairRDD<String,Tuple2<String,Double>> rdd1 = rdd2.mapToPair(
                                    ?
);
```

**Reduce-by-key:** Esto crea un nuevo RDD `rdd1` como resultado de una operación reduce-by-key aplicada a `rdd2`.

```
JavaPairRDD<String,Tuple2<String,Double>> rdd1 = rdd2.reduceByKey(
                                    ?
);
```

**Join:** Esto crea un nuevo RDD `rdd1` resultante de un join entre `rdd2` y `rdd3`.

```
JavaPairRDD<String, Tuple2<Tuple2<String, Double>, Double>> rdd1 =  rdd2.join(rdd3);
```

**First lambda:** Esto crea pares de la forma $(s_1, (s_2, d))$ a partir de triples $(s_1, s_2, d)$, donde $s_1$, $s_2$ son strings, y $d$ es un double.

```
tup -> new Tuple2<String,Tuple2<String,Double>>(tup._1(),new Tuple2<String,Double>(tup._2(),tup._3()))
```

**Second lambda:** Esta función lambda recopila un valor máximo y una lista de nombres de elementos que tienen dicho valor máximo.

```
(a, b) ->
  { if(a._2 > b._2) return a;
    else if(b._2 > a._2) return b;
    return new Tuple2<String,Double>(a._1 + "|" + b._1, a._2);
  }
```