BigData Spark Hands-On 5

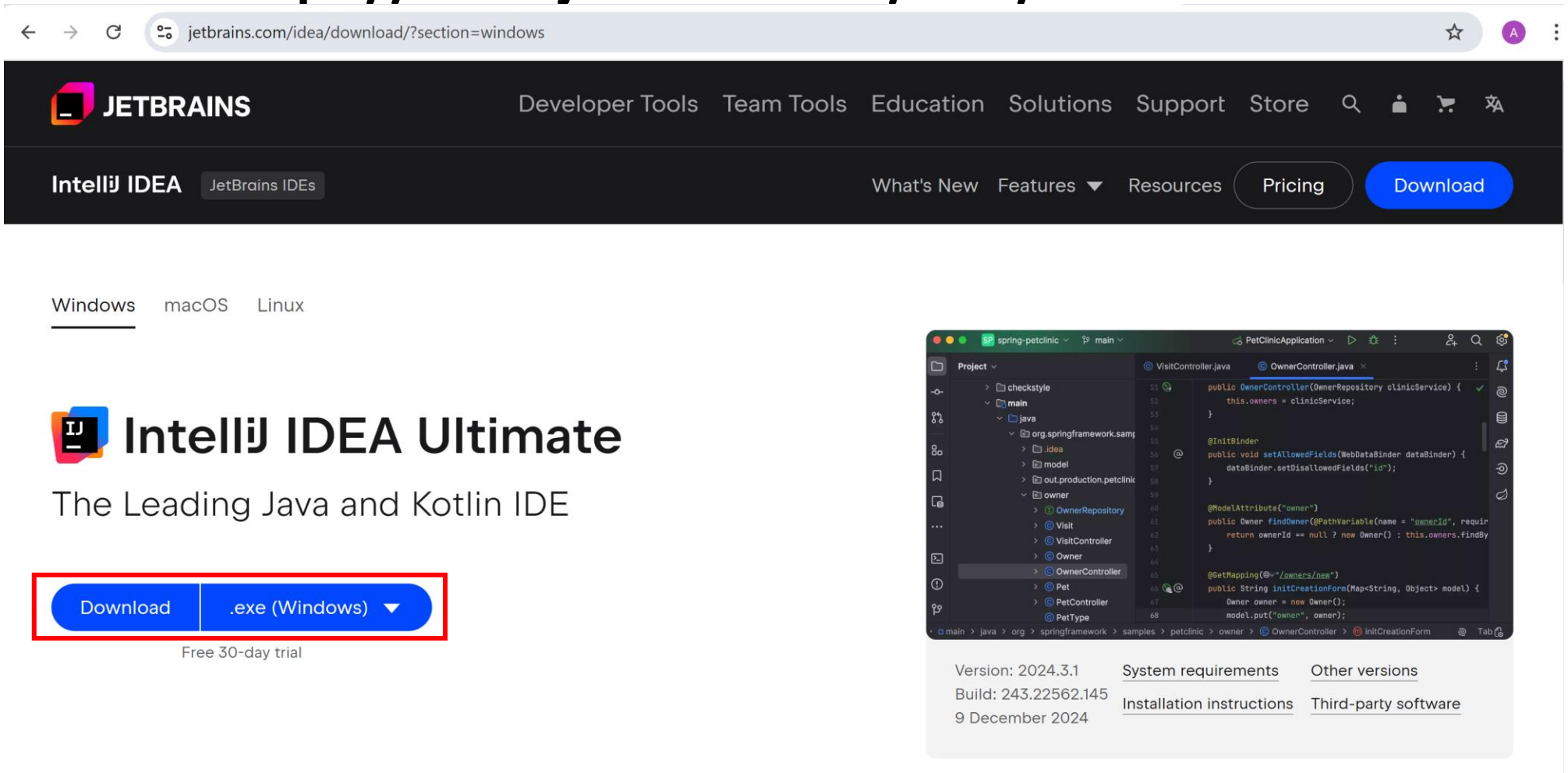
Spark Java

Esilv 2024

arnaud.nauwynck@gmail.com

Pre-Requisite : Install Java IDE (IDEA IntelliJ)

https://www.jetbrains.com/idea/download



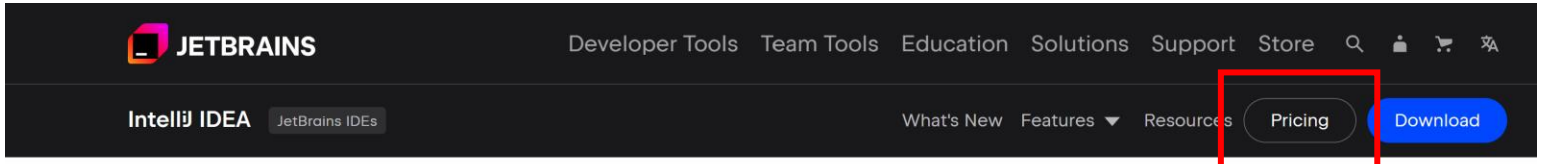
Note : pre-requisites

IntelliJ IDEA "**Community**" Edition is Free (but supports only Java/Kotlin langage)

you can also choose "IntelliJ IDEA **Ultimate**" Edition
free for students => register an account with your "@esilv" email

for doing the TD only => you have free 30 days-trial without registering

Steps to get IntelliJ IDEA "Ultimate" Edition



Subscription Options and Pricing

For Organizations | For Individual Use | Special Categories

Yearly billing save 2 months | Monthly billing

Best offer

IntelliJ IDEA Ultimate

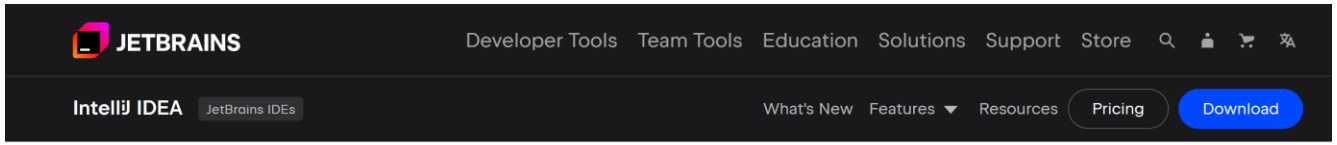
The Leading Java and Kotlin IDE

IntelliJ IDEA

All Products Pack

Get 12 IDEs, 3 extensions, 2 profilers, and a collaborative development service – all in one subscription.

Includes 18 tools



Subscription Options and Pricing

For Organizations | For Individual Use | Special Categories

For students and teachers FREE

Students and academic staff members are eligible to use all JetBrains tools free, upon verification of their university/college domain email or ISIC card.

Learn more

For classroom assistance FREE

Universities, colleges, schools, and non-commercial educational organizations are eligible for free licensing to install all JetBrains tools in classrooms and computer labs for educational purposes.

Learn more

For Open Source projects FREE

Core maintainers of open-source projects are welcome to explore collaboration opportunities and receive free access to JetBrains tools.

Learn more

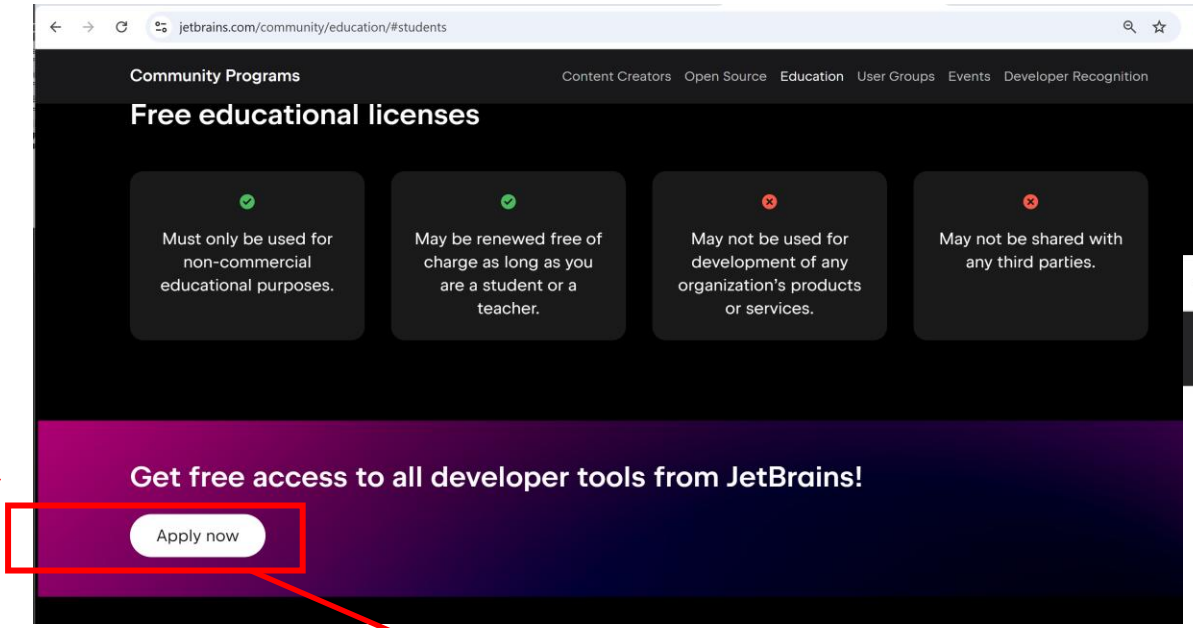
For universities and educational organizations 50% off

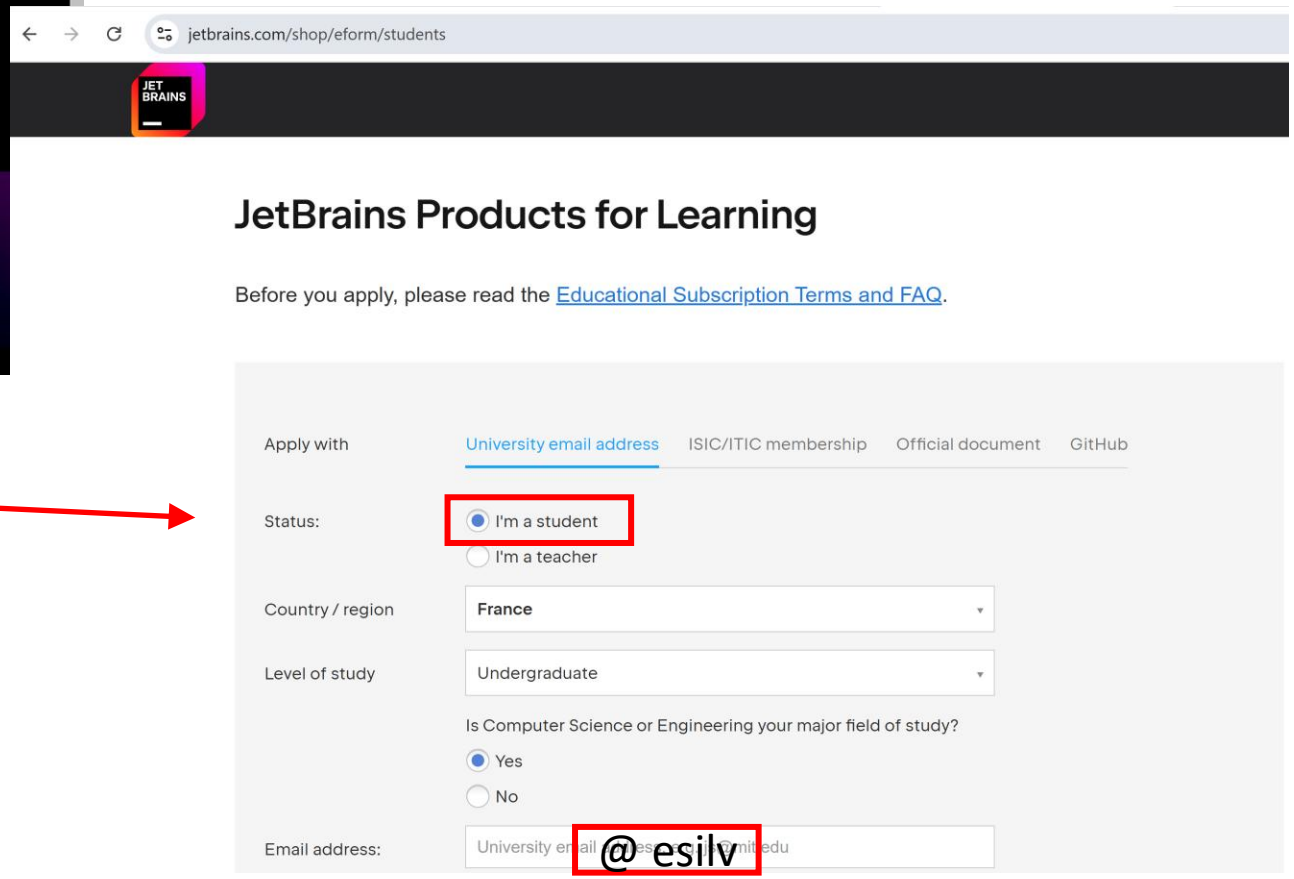
For startups 50% off

Private software development companies which

For training courses, coding schools, and bootcamps FREE

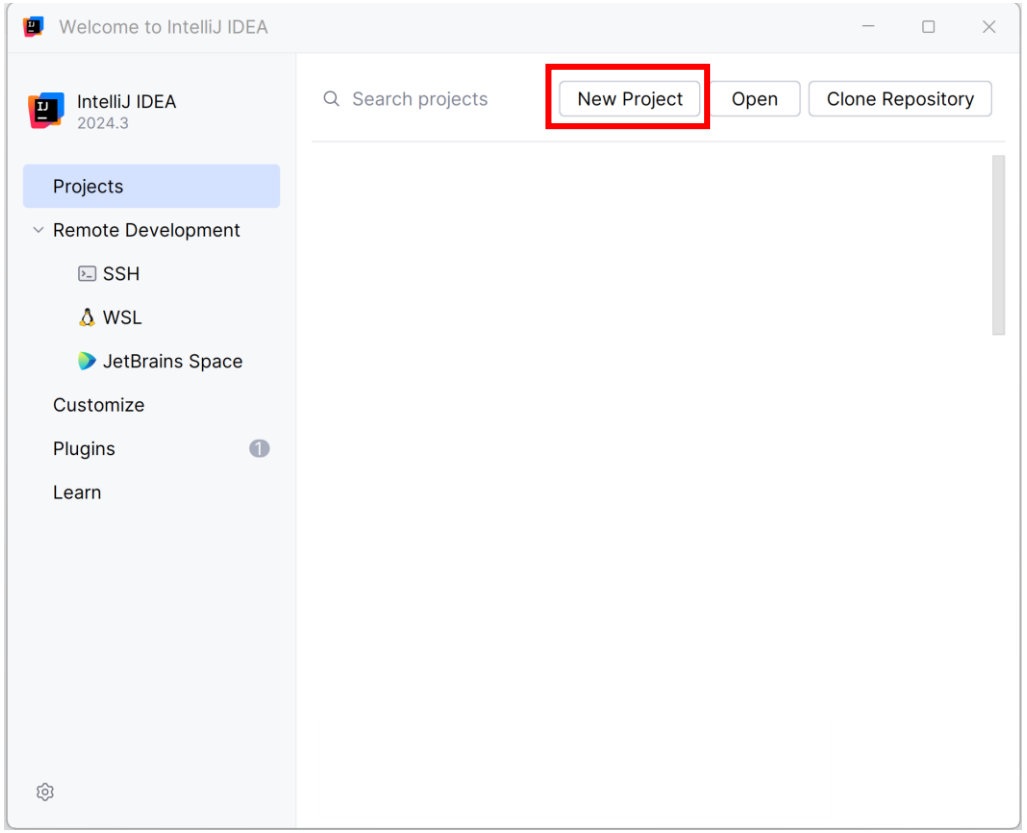
Steps …


jetbrains.com/community/education/#students
Community Programs
Content Creators
Open Source
Education
User Groups
Events
Developer Recognition
Free educational licenses
Must only be used for non-commercial educational purposes.
May be renewed free of charge as long as you are a student or a teacher.
May not be used for development of any organization's products or services.
May not be shared with any third parties.
Get free access to all developer tools from JetBrains!
Apply now

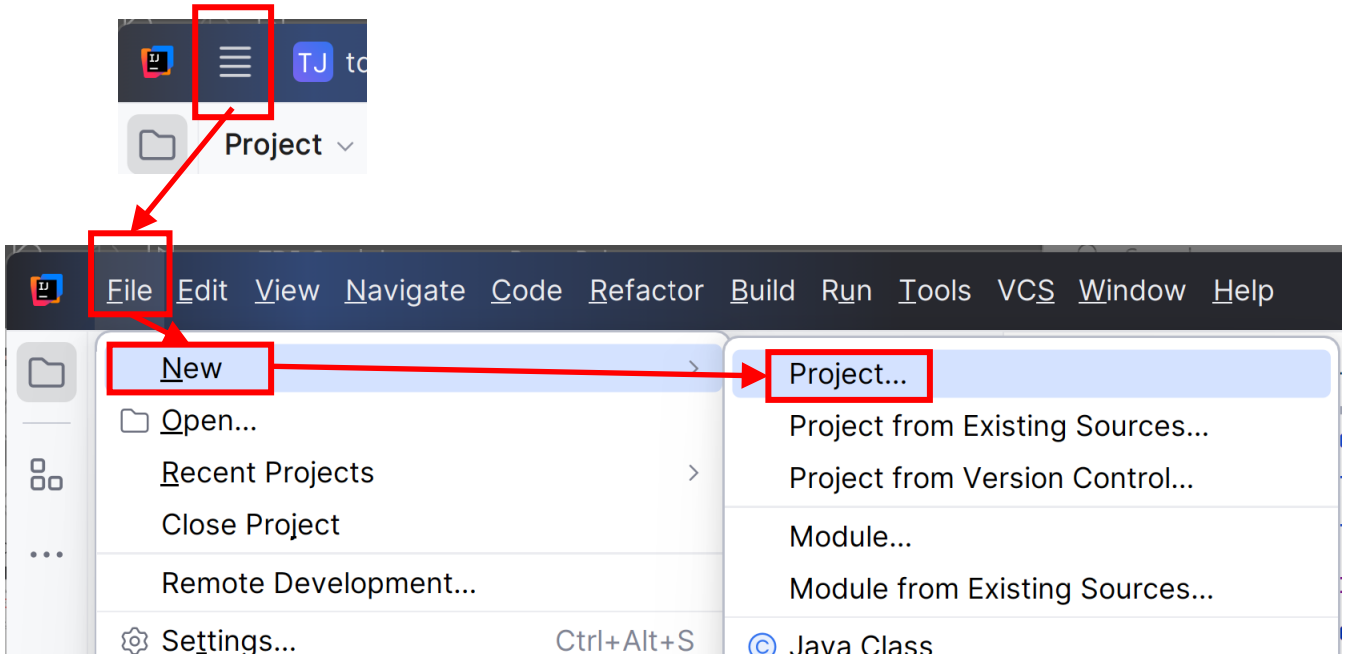

jetbrains.com/shop/eform/students
JET BRAINS
JetBrains Products for Learning
Before you apply, please read the Educational Subscription Terms and FAQ.
Apply with
University email address
ISIC/ITIC membership
Official document
GitHub
Status:
I'm a student
I'm a teacher
Country / region
France
Level of study
Undergraduate
Is Computer Science or Engineering your major field of study?
Yes
No
Email address:
@esilv

Create a New Project

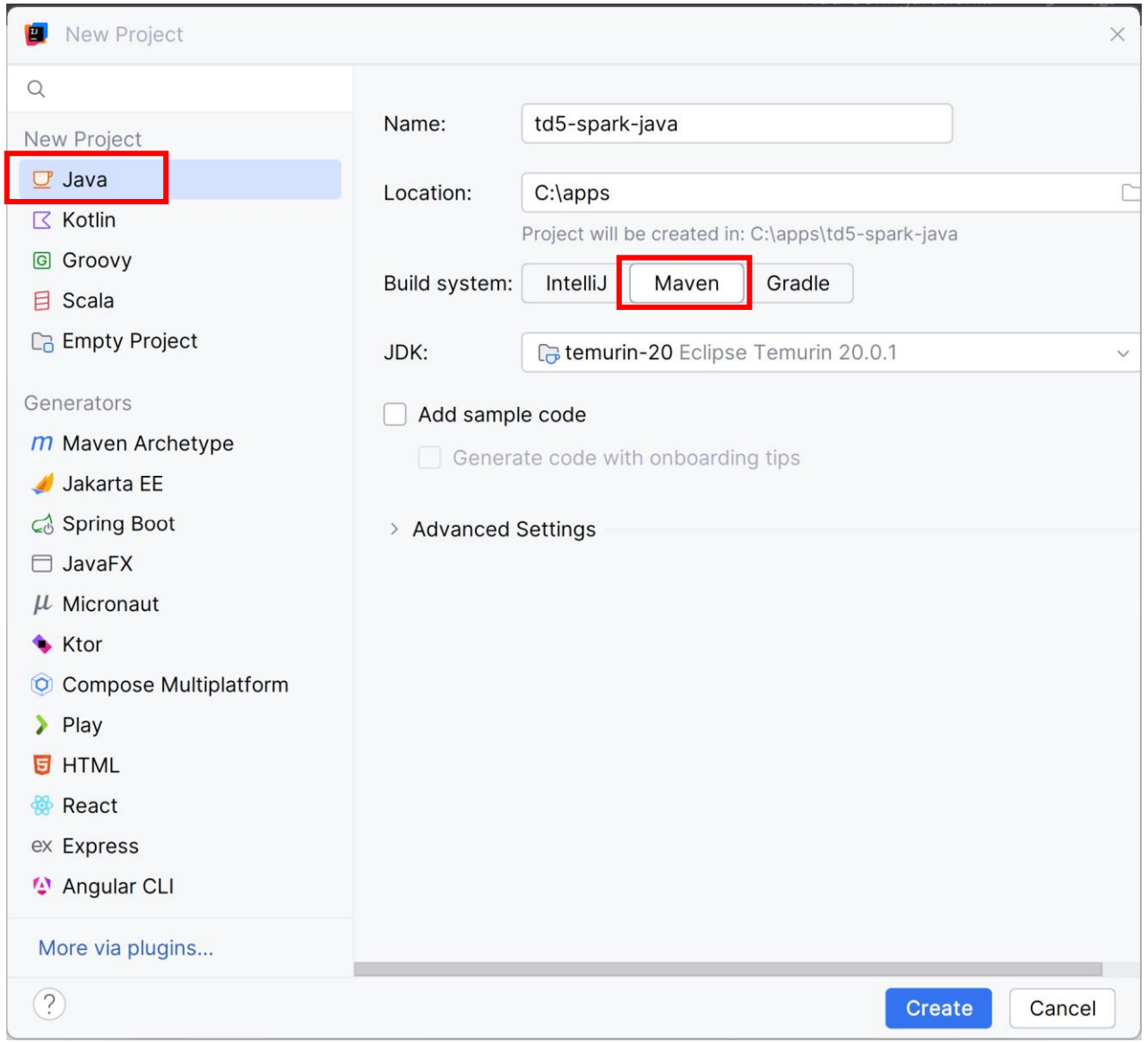
When openning the first time (no current project)



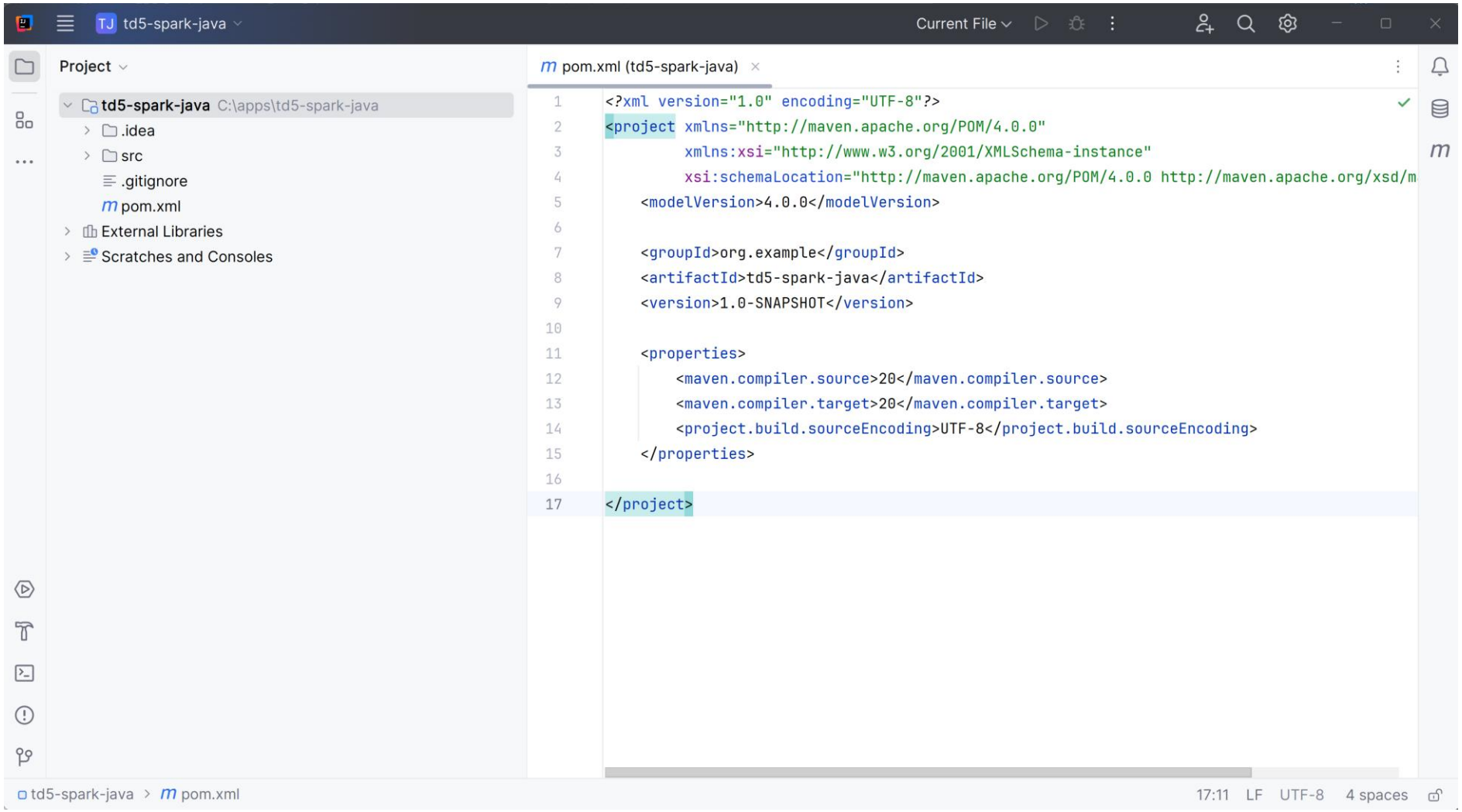
When re-openning (already a current project)



Create New Java (build with Maven) Project

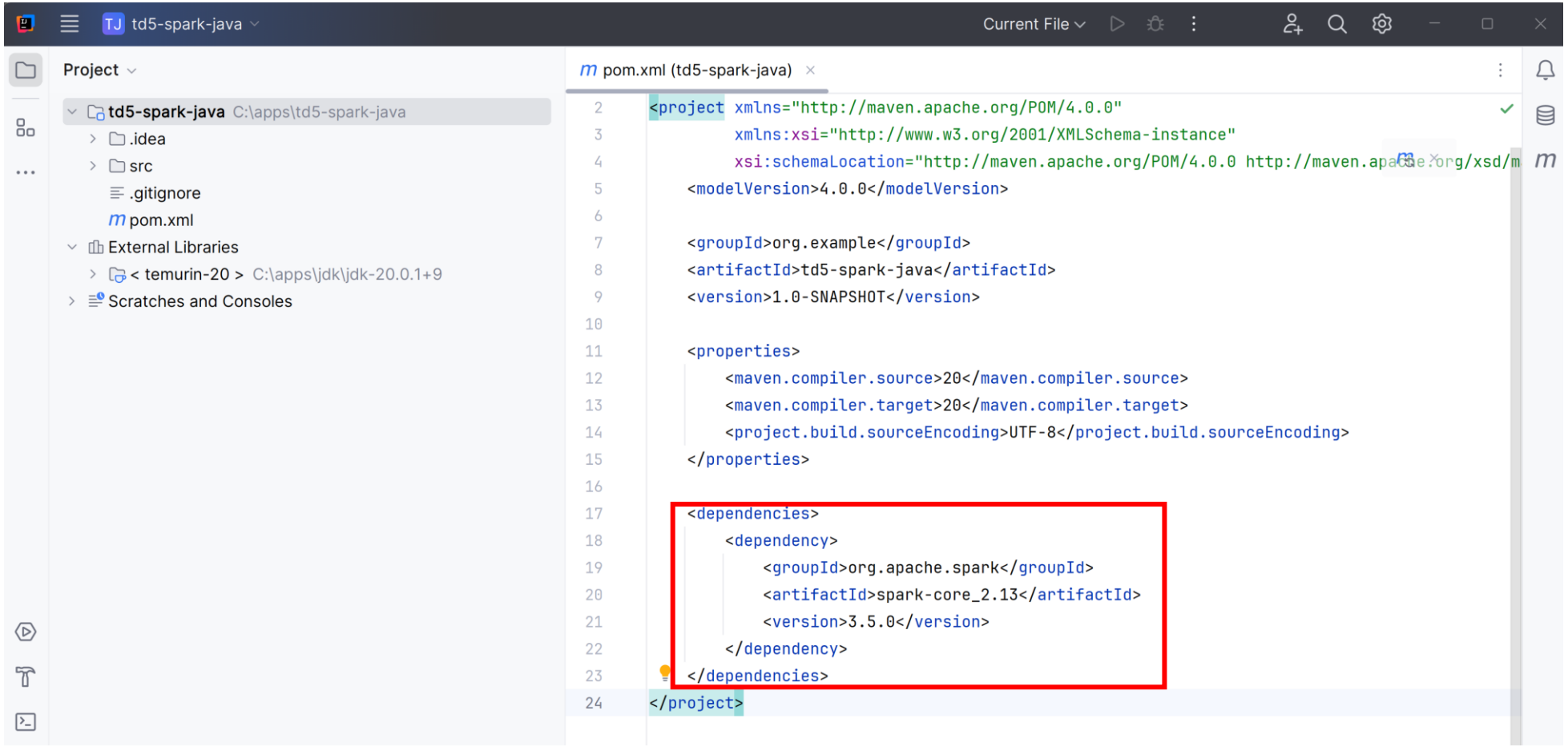


pom.xml file
(pom = Project Object Model... for Maven build)



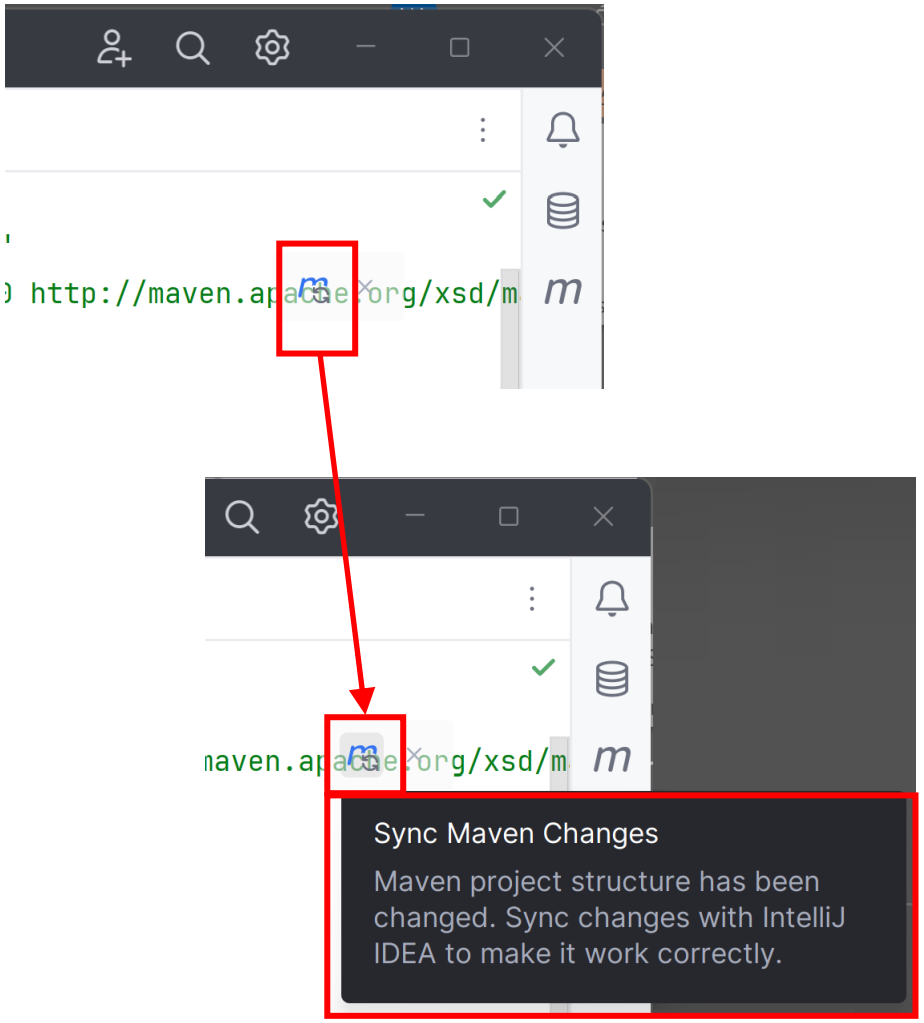
Edit to add maven <dependency>

```
<dependencies>
   <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.13</artifactId>
      <version>3.5.0</version>
   </dependency>
</dependencies>
```

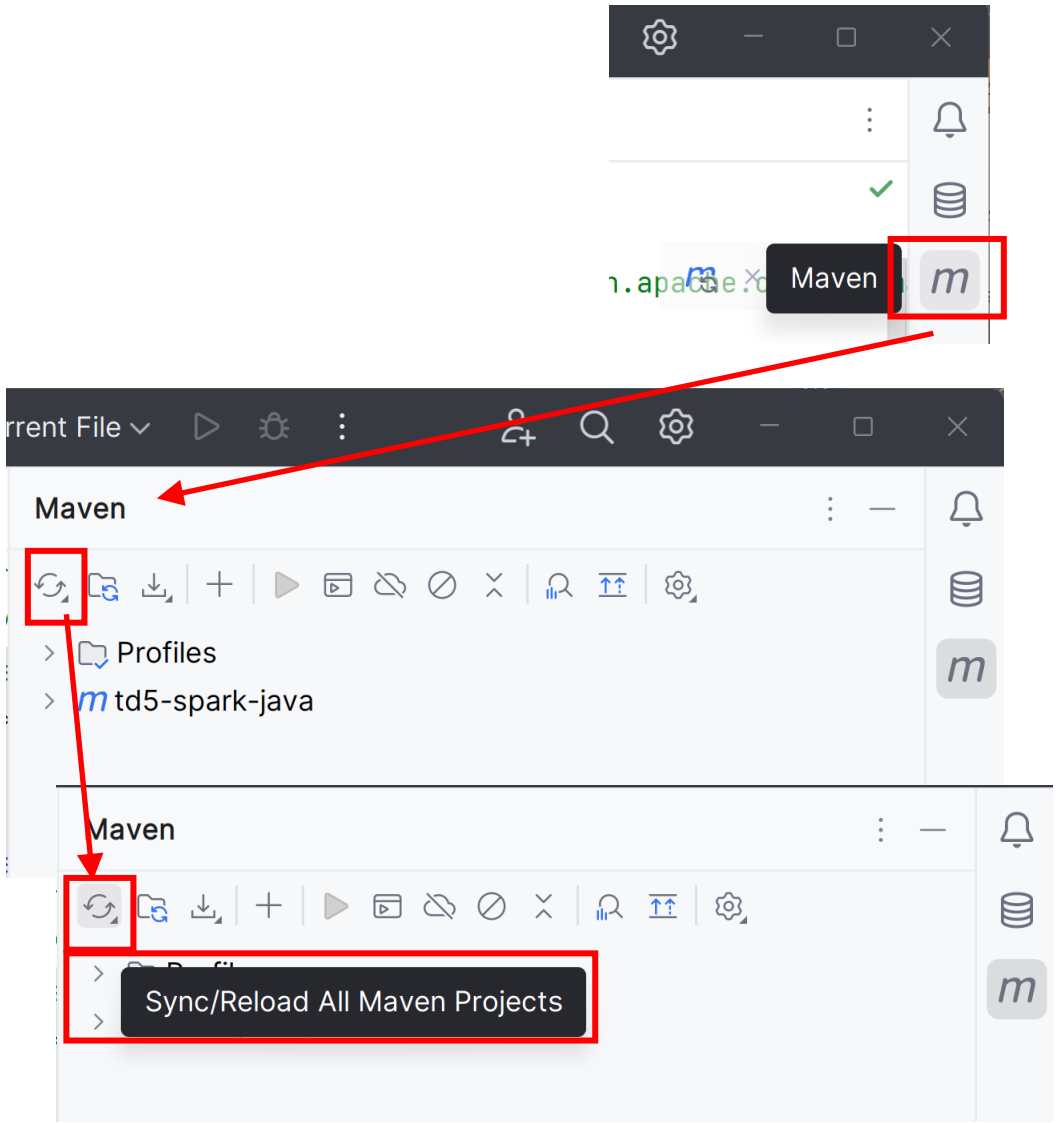


Then Click on "Maven Sync" icon

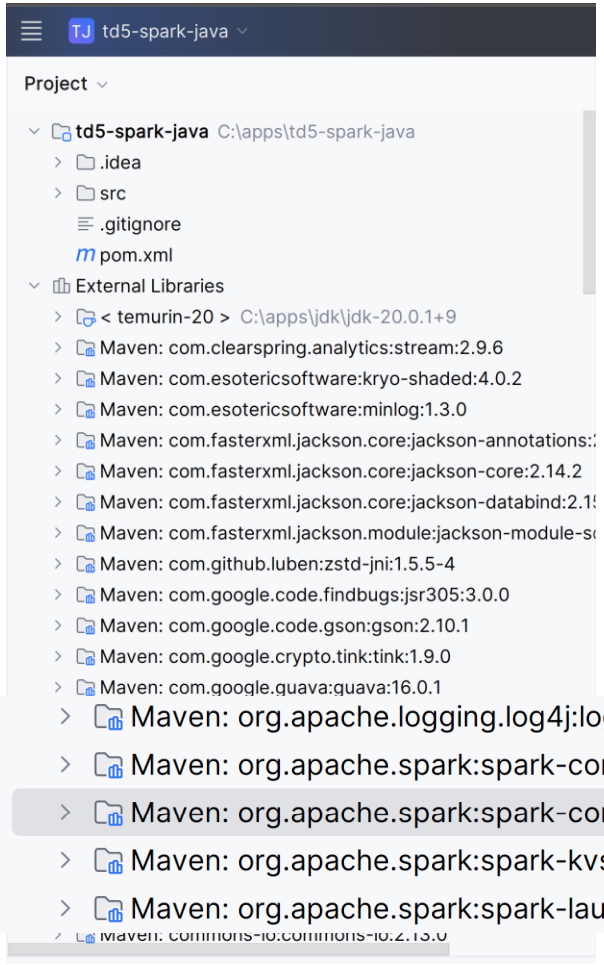
difficult to see inside text editor



can also do from right "m"=Maven button

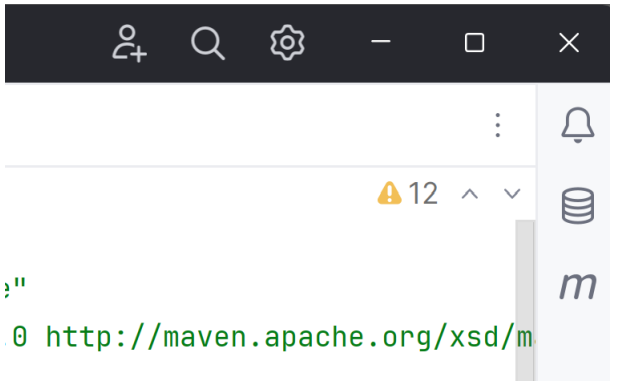


Maven will download dependencies ... can take ~5mn



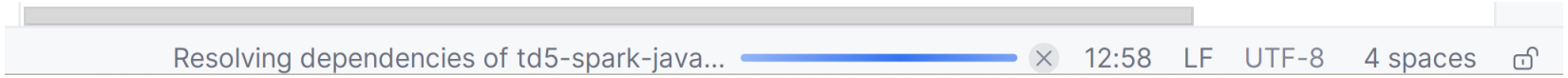
The "External Libraries" now contains hundreds of Maven dependencies

... including spark

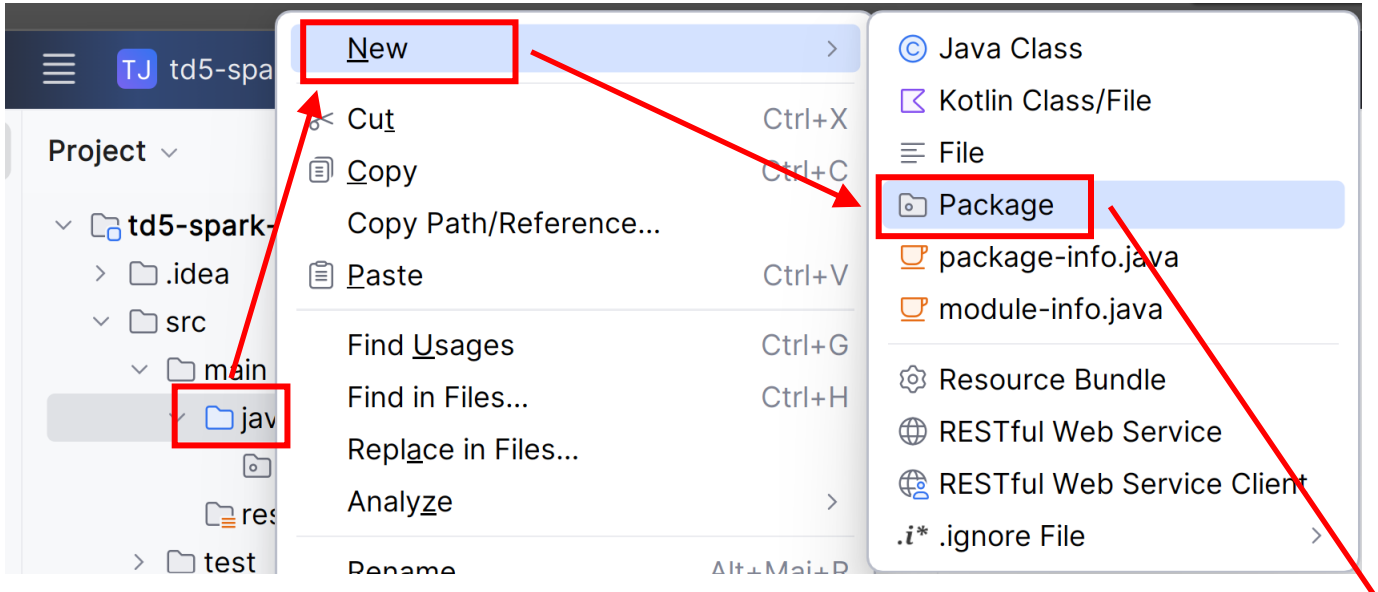


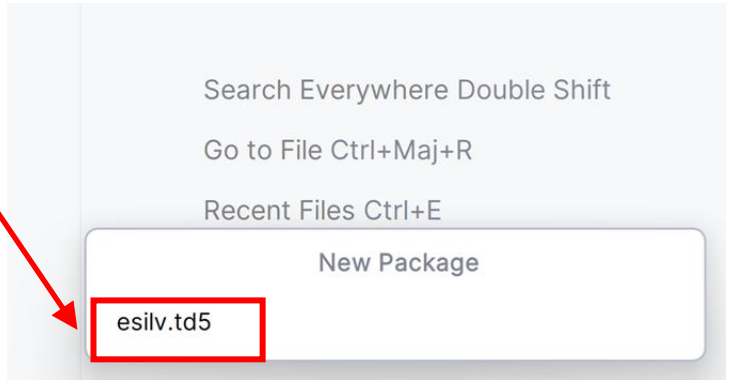
The "Sync" icon will disapear

The progress bar will show "resolving dependencies" / "downloading"

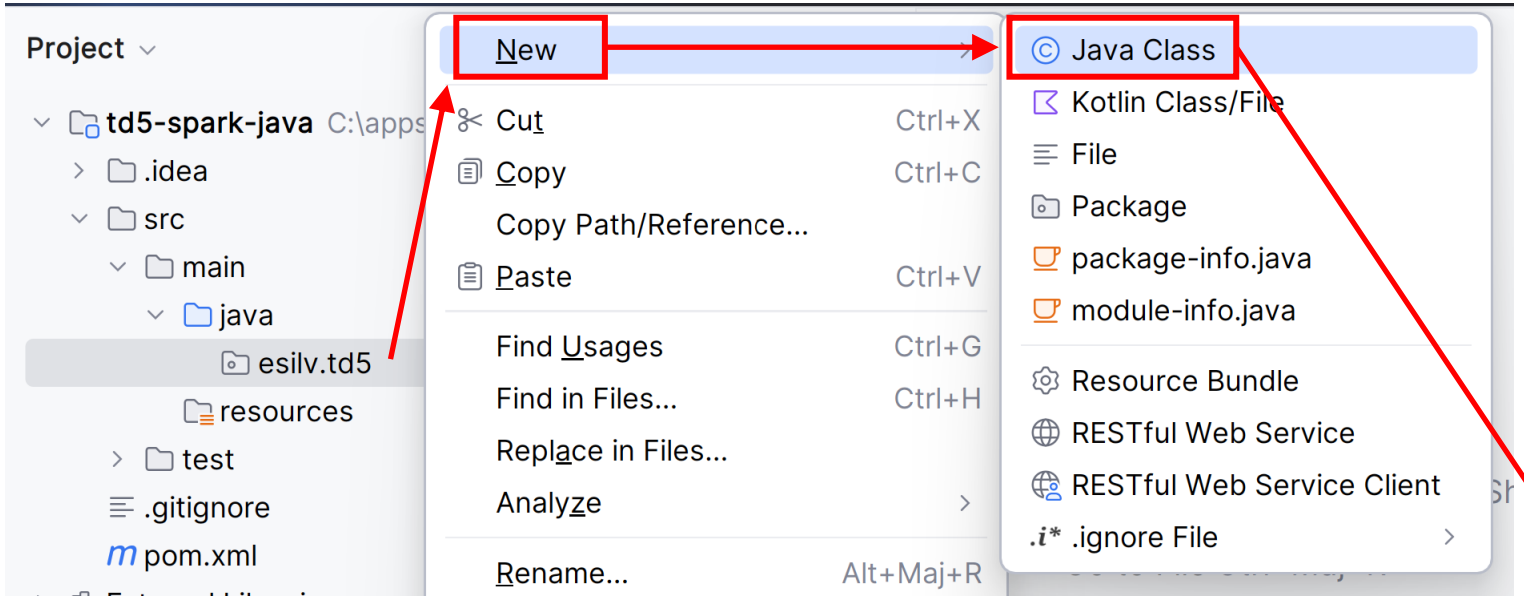


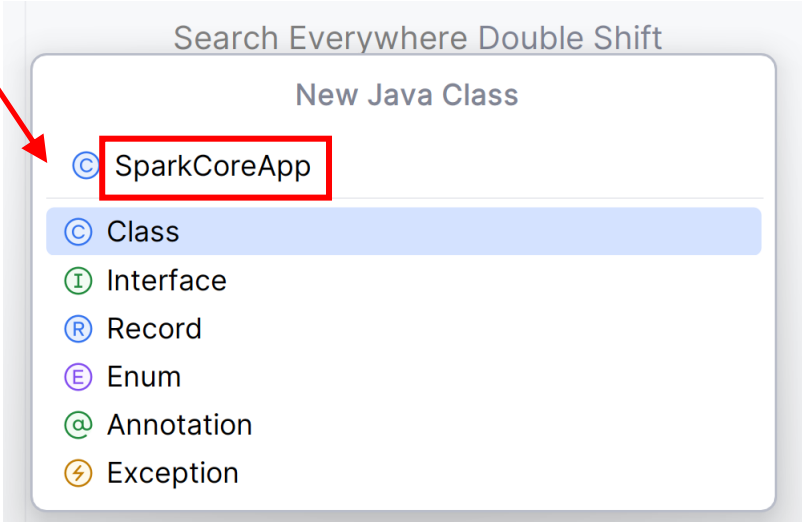
Create a new Package: "esilv.td5"



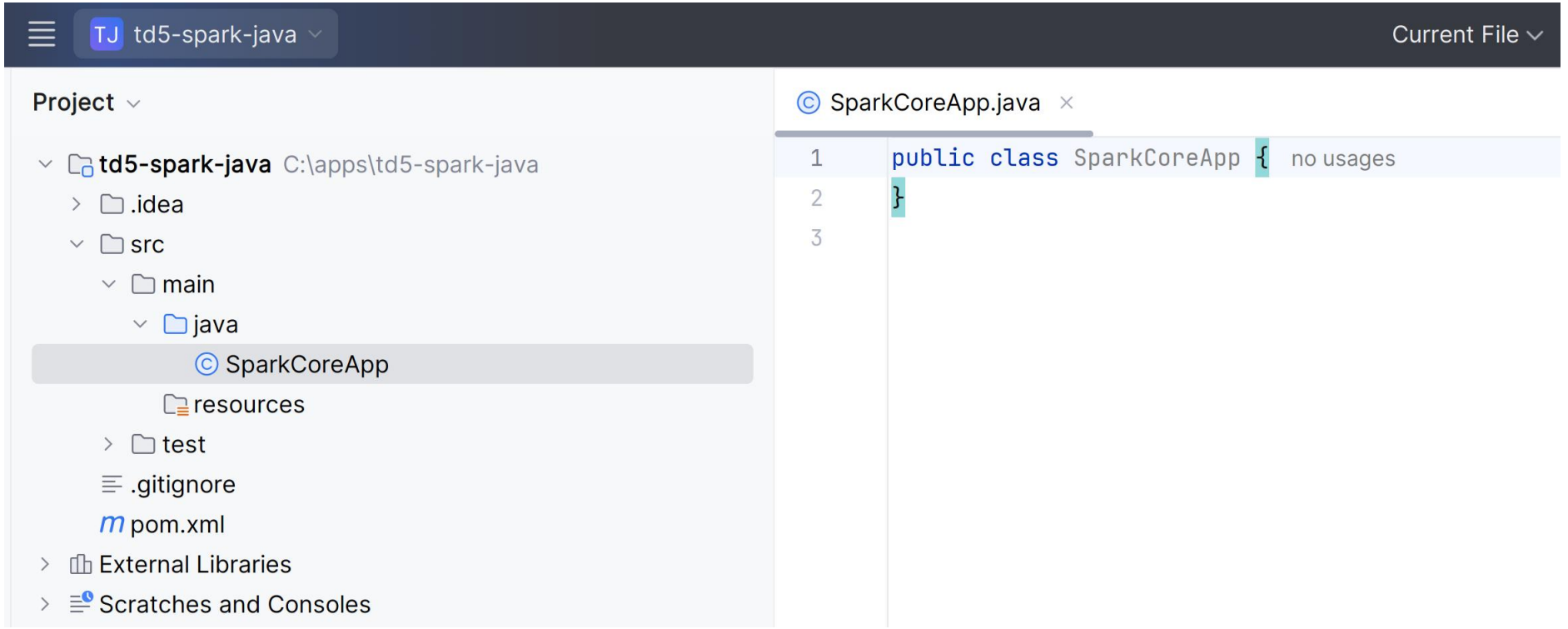


Create a new Java class "SparkCoreApp" in package esilv.td5





BAD ... using no package name

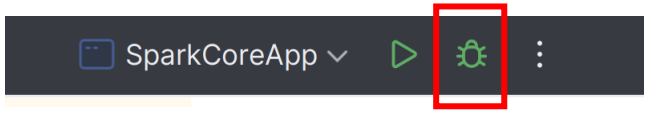


in main()
initialise spark-core "SparkContext" singleton
(we will see later SparkSession in spark-sql)

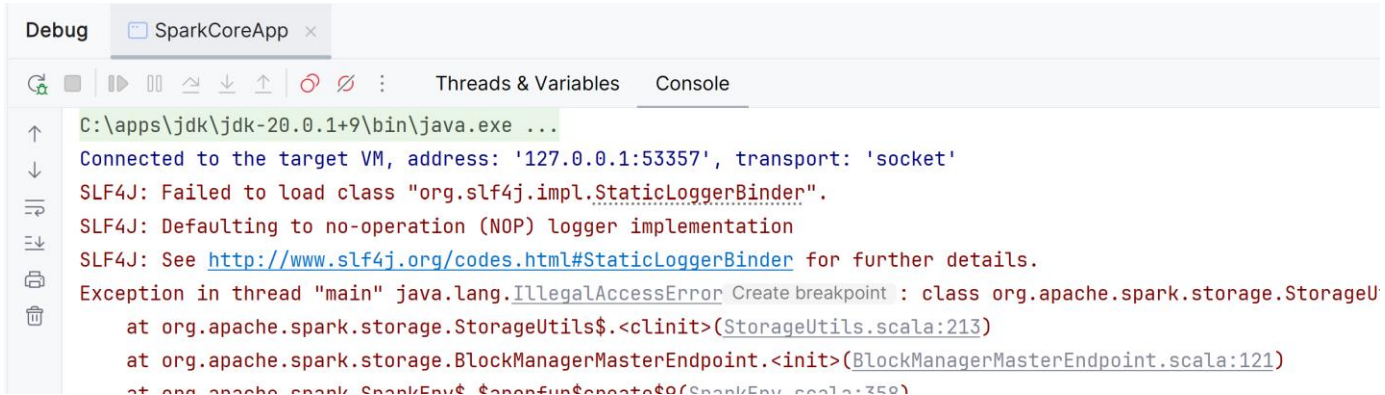
```
import org.apache.spark.SparkConf;
import org.apache.spark.SparkContext;
import org.apache.spark.rdd.RDD;

public class SparkCoreApp {

 public static void main(String[] args) {
      SparkConf sparkConf = new SparkConf()
            .setAppName("td5-sparkcore-app")
            .setMaster("local[*]");
      SparkContext sc = SparkContext.getOrCreate(sparkConf);

      RDD<String> lineRdd = sc.textFile("c:/data/loremIpsum.txt", 1);
      long lineCount = lineRdd.count();
      System.out.println("line count:" + lineCount);
      sc.stop();
   }
}
```

Debug (or Run)
will FAIL on jdk >=11 and work on jdk <11

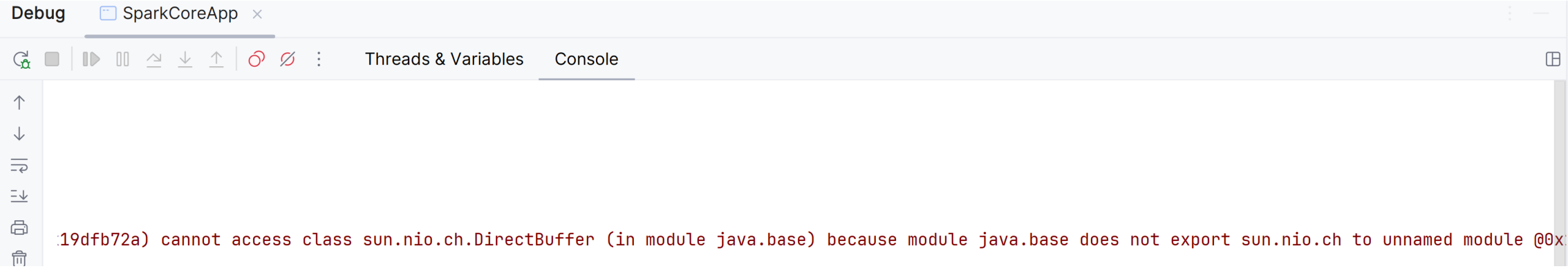
```
7
8   public class SparkCoreApp {
9
10      public static void main(String[] args) {
11          SparkConf sparkConf = new SparkConf()
```



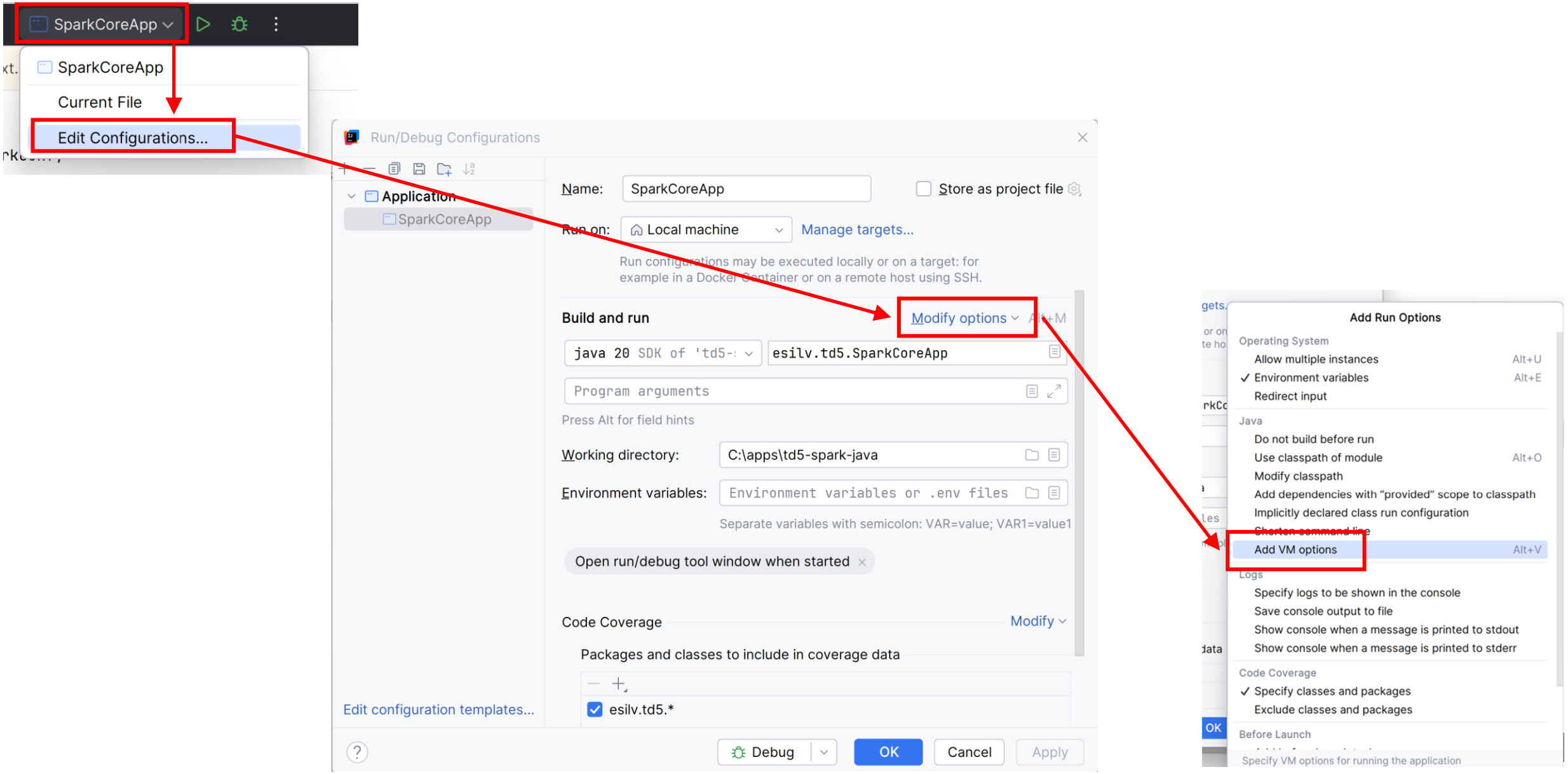

```
9
10  Run 'SparkCoreApp.main()'
    Debug 'SparkCoreApp.main()'
    Run 'SparkCoreApp.main()' with Coverage
    Run with Profiler  Ctrl+Maj+F10
11
12
13
```



Failure cause : java security on modules

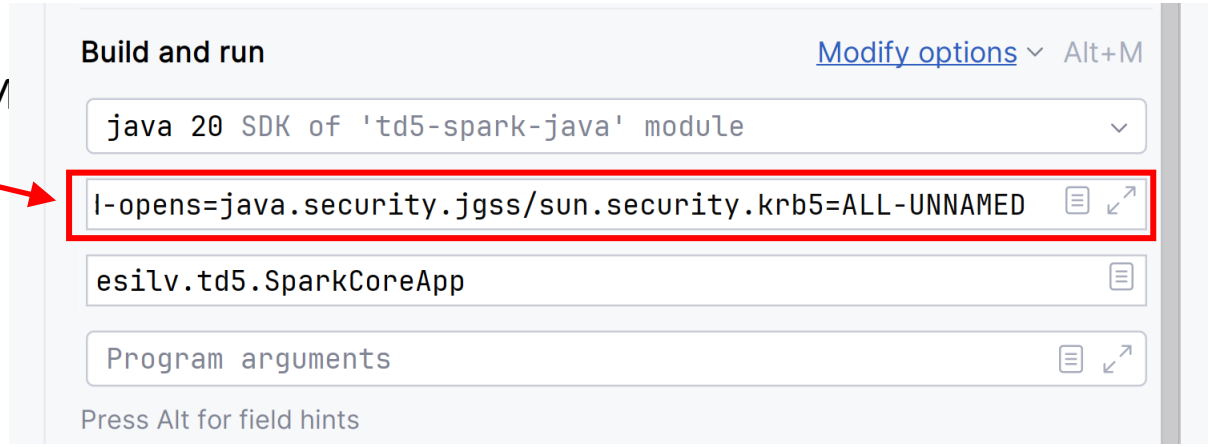


Edit Run/Debug Config - JVM Options



add JVM options

-Xmx3g
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/java.lang.invoke=ALL-UNNAMED
--add-opens=java.base/java.lang.reflect=ALL-UNNAMED
--add-opens=java.base/java.io=ALL-UNNAMED
--add-opens=java.base/java.net=ALL-UNNAMED
--add-opens=java.base/java.nio=ALL-UNNAMED
--add-opens=java.base/java.util=ALL-UNNAMED
--add-opens=java.base/java.util.concurrent=ALL-UNNAMED
--add-opens=java.base/java.util.concurrent.atomic=ALL-UNNAMED
--add-opens=java.base/sun.nio.ch=ALL-UNNAMED
--add-opens=java.base/sun.nio.cs=ALL-UNNAMED
--add-opens=java.base/sun.security.action=ALL-UNNAMED
--add-opens=java.base/sun.util.calendar=ALL-UNNAMED
--add-opens=java.security.jgss/sun.security.krb5=ALL-UNNAM



Relaunch ... it works

Changing to use module spark-sql SparkSession

Edit pom.xml + Maven Sync Changes

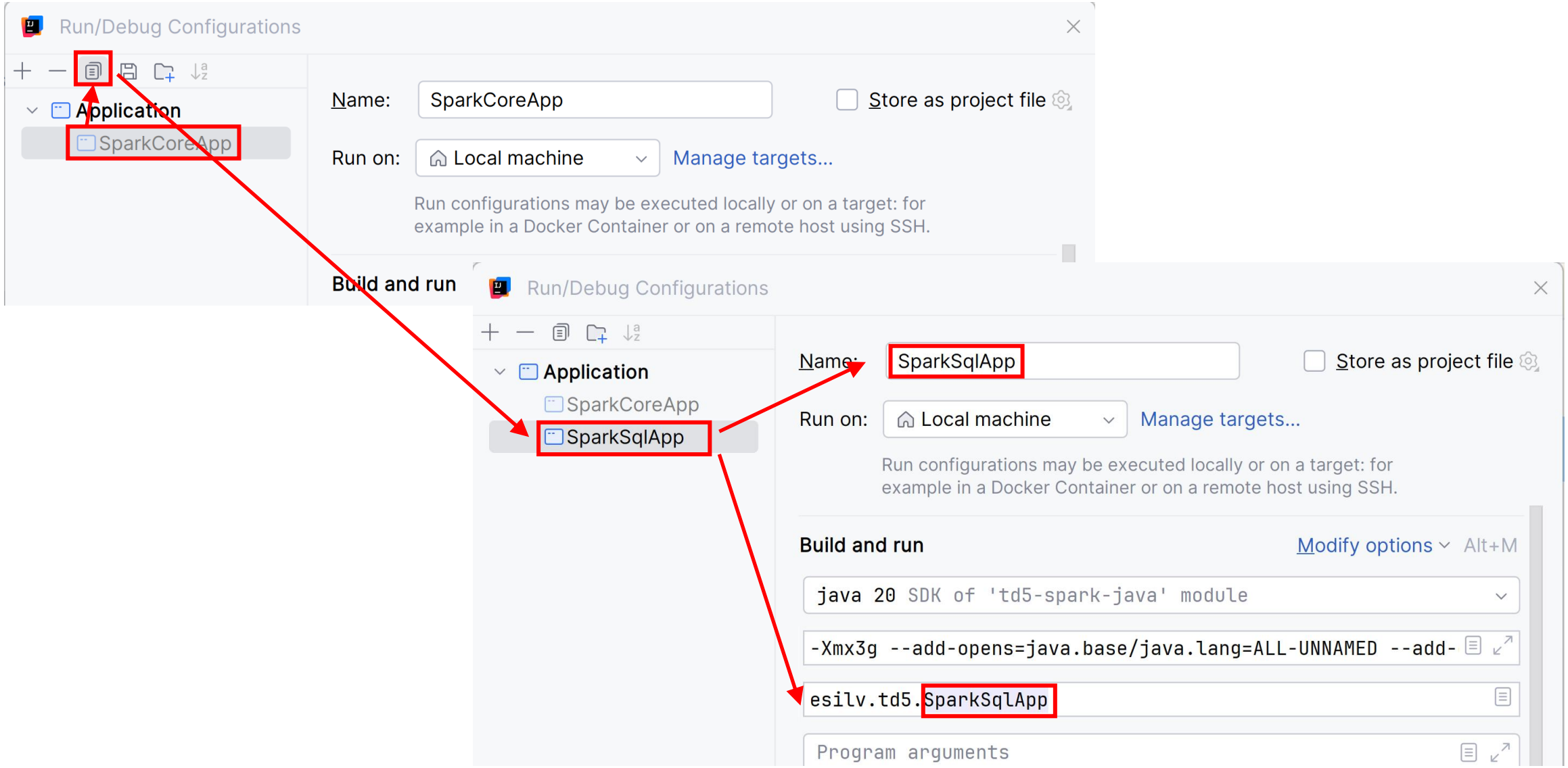
```
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-sql_2.13</artifactId>
    <version>3.5.0</version>
 </dependency>
```

Notice: spark-sql_2.13 dependends on spark-core_2.13
... you can remove spark-core, it is implicit dependency

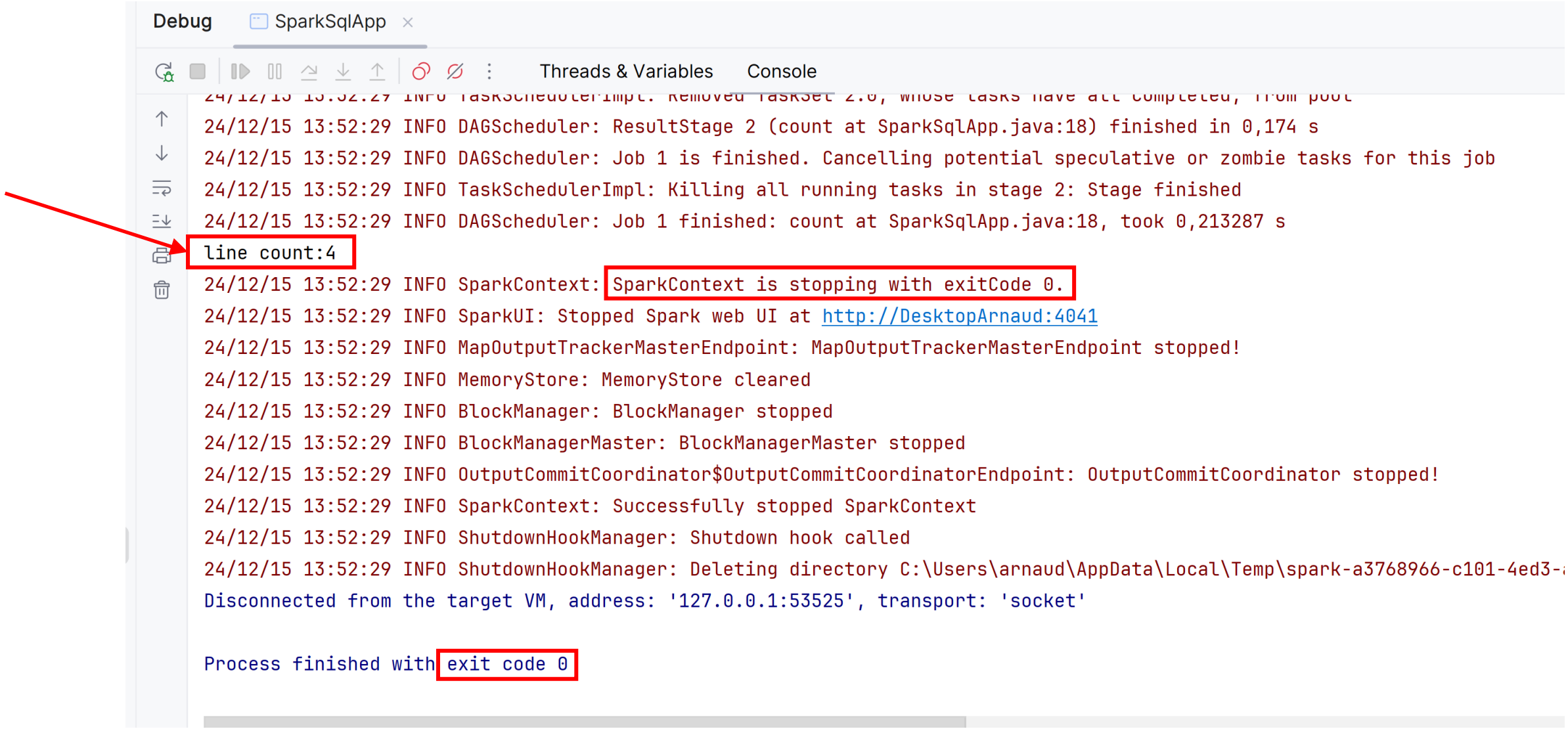
create new class "SparkSqlApp"

```
import org.apache.spark.SparkConf;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.SparkSession;

public class SparkSqlApp {
   public static void main(String[] args) {
      SparkConf sparkConf = new SparkConf()
            .setAppName("td5-sparksql-app")
            .setMaster("local[*]");
      SparkSession spark = SparkSession.builder()
            .config(sparkConf)
            .getOrCreate();

      Dataset<String> lineDs = spark.read().textFile("c:/data/loremIpsum.txt");
      long lineCount = lineDs.count();
      System.out.println("line count:" + lineCount);
      spark.stop();
   }
}
```

Duplicate Run/Debug Configurations (or Edit similar VM Options) ... Debug



Debug ... it works



using Dataset API with Encoders

createDataset .. need Encoder

```
List<String> testList = Arrays.asList("hello", "world");

Dataset<String> testDs = spark.createDataset(   ???????    );      // <==  complete missing code

System.out.println("testDs count:" + testDs.count());
```

Dataset<String> .map() to use upper-cases

```
Dataset<String> testUpperDs = testDs.map(
        ??????   x -> x.toUpperCase(),              // <==  complete missing code
        ??????                                      // <==  complete missing code
 );

System.out.println("testUpperDs.show()");
testUpperDs.show();
```

```
+-----+
|value|
+-----+
|HELLO|
|WORLD|
+-----+
```

Exerise : implement "Words Count" on file "loremIpsum.txt"

```
Dataset<String> wordDs = lineDs.flatMap(
        ????????  line -> {    // <======= complete missing code
           String[] split = line.replaceAll("[,;\\.?!]", " ").replace("  ", " ").split(" ");
           return Arrays.asList(split).iterator();
        },
        ???????? );            // <======= complete missing code

wordDs.cache();

long wordCount = wordDs.count();
System.out.println("words count: " + wordCount);

System.out.println("words:");
wordDs.show();
```

using Java Bean Encoders

Dataset<YourBean> --> to DataFrame ". toDF()"
DataFrame --> ".as()" Dataset<YourBean>

Pre-Requisite : create a Pojo class YourBean

```
public class  YourBean {

    public int field1;
    public String field2;

    public YourBean() {
    }
    public YourBean(int field1, String field2) {
       this.field1 = field1;
       this.field2 = field2;
    }
}
```

Convert List<YourBean> to Dataset<YourBean>

```
List<YourBean> beanLs = IntStream.rangeClosed(0, 10)
        .mapToObj(i -> new YourBean(i, "Hello " + i))
        .collect(Collectors.toList());

Dataset<YourBean> beanDs = spark.createDataset(beanLs, ??????); // <====== complete missing code

System.out.println("beanDs:");
beanDs.show();
```

Complete Dataset<YourBean> to DataFrame = Dataset<Row> and back to Dataset

```
Dataset<Row> beanDF = beanDs. ??????  // <==== complete missing code

System.out.println("beanDF:");
beanDF.show();


Dataset<YourBean> beanDs2 = beanDF. ??????  // <==== complete missing code

System.out.println("beanDs2:");
beanDs2.show();
```

Exercise
Using spark.sql"(select * from hive_table")

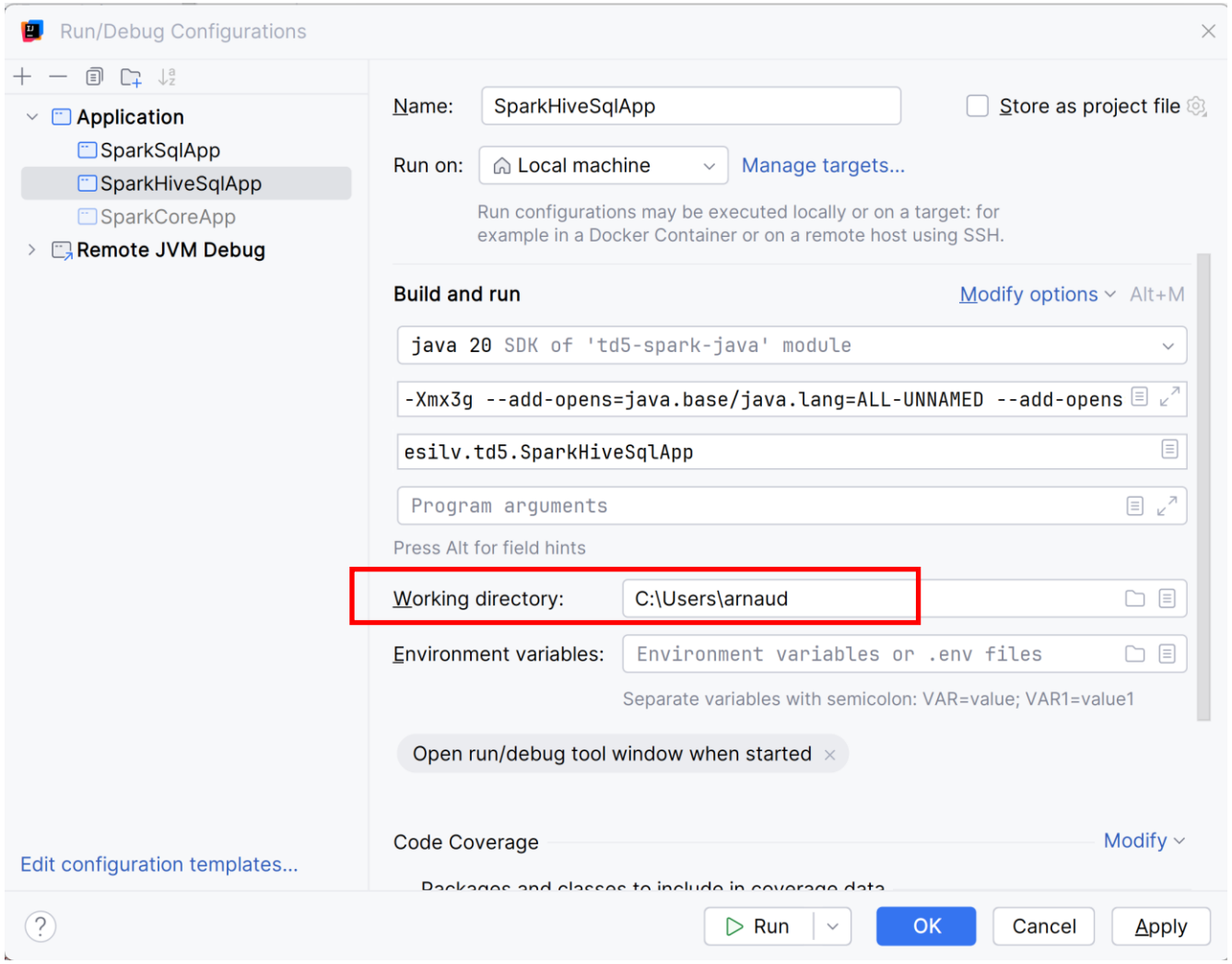
Edit pom.xml again, add dependency

```
<dependency>
    <groupId>org.apache.spark</groupId>
    <artifactId>spark-hive_2.13</artifactId>
    <version>3.5.0</version>
 </dependency>
```

create new class SparkHiveSqlApp

```
public class SparkHiveSqlApp {
    public static void main(String[] args) throws Exception {
        System.out.println("current working dir: " + new File(".").getAbsolutePath());
        SparkConf sparkConf = new SparkConf()
                .setAppName("td5-sparkhivesql-app")
                .setMaster("local[*]");
        SparkSession spark = SparkSession.builder()
                .config(sparkConf)
                .enableHiveSupport()  // <=======  added !!! (need module spark-hive +  working dir ..)
                .getOrCreate();

        spark.sql("show databases").show(100, false);
        spark.sql("show tables in db1").show(100, false);

        Dataset<Row> addrDs = spark.sql("select * from db1.addr");
        System.out.println("addrDs count:" + addrDs.count());

        System.out.println("Finished\n\n");
        spark.stop();
    }
}
```

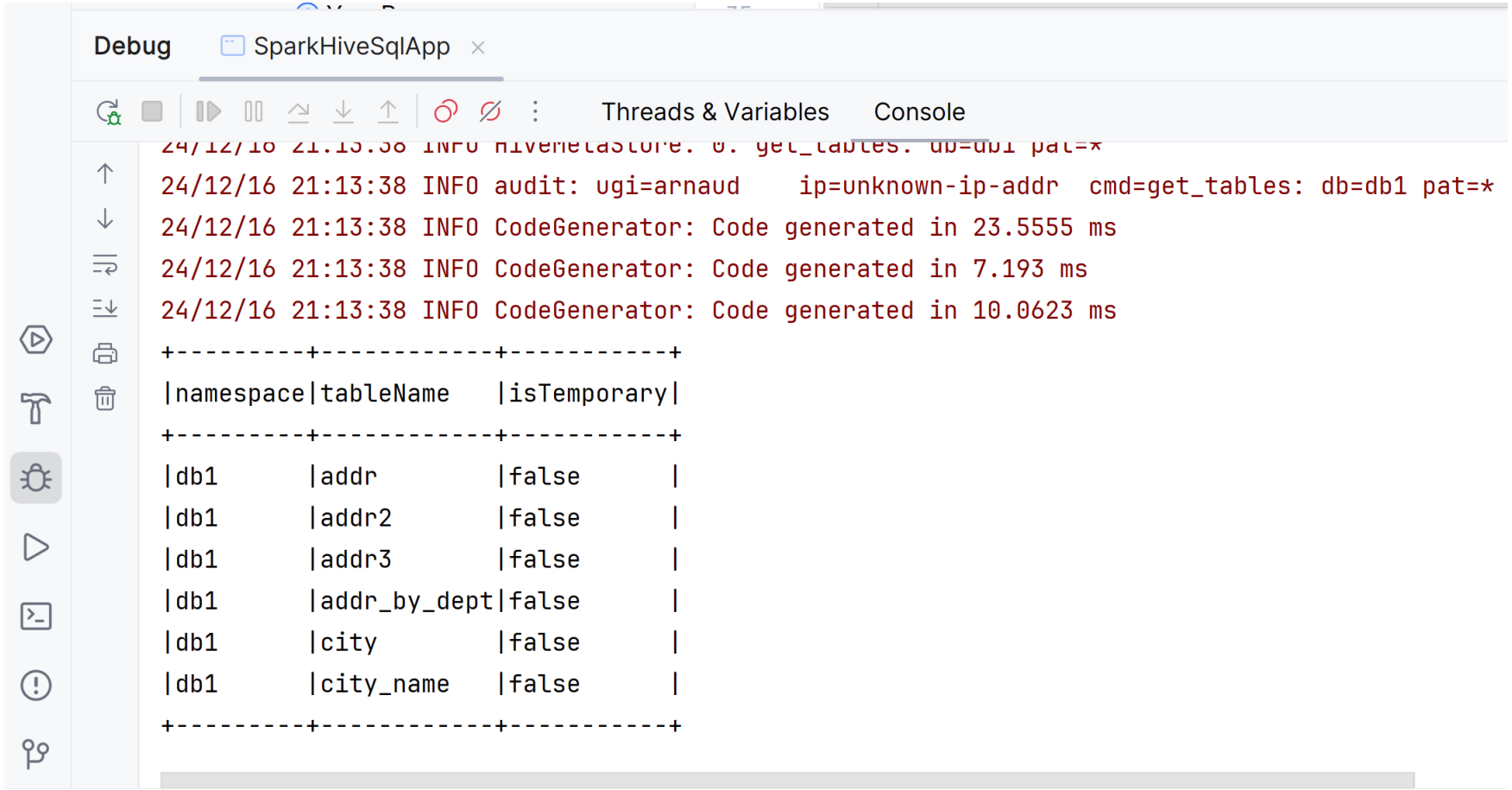
Run... need to set "currentWorkingDir"




```
24/12/16 21:13:31 INFO SharedState: Setting hive.metastore.warehouse.dir ('null') to the value of spark.sql.warehouse.dir.
24/12/16 21:13:31 INFO SharedState: Warehouse path is 'file:/C:/Users/arnaud/spark-warehouse'.
24/12/16 21:13:34 INFO HiveUtils: Initializing HiveMetastoreConnection version 2.3.9 using Spark classes.
24/12/16 21:13:34 INFO HiveClientImpl: Warehouse location for Hive client (version 2.3.9) is file:/C:/Users/arnaud/spark-warehouse
```

Already Locked ??
(=> need to stop all others running "spark-shell")

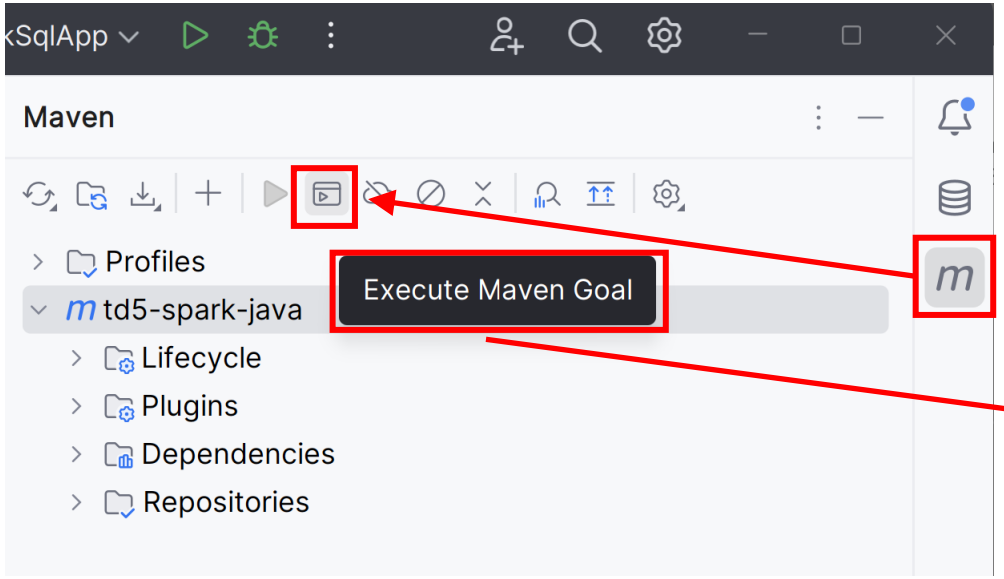
It works..

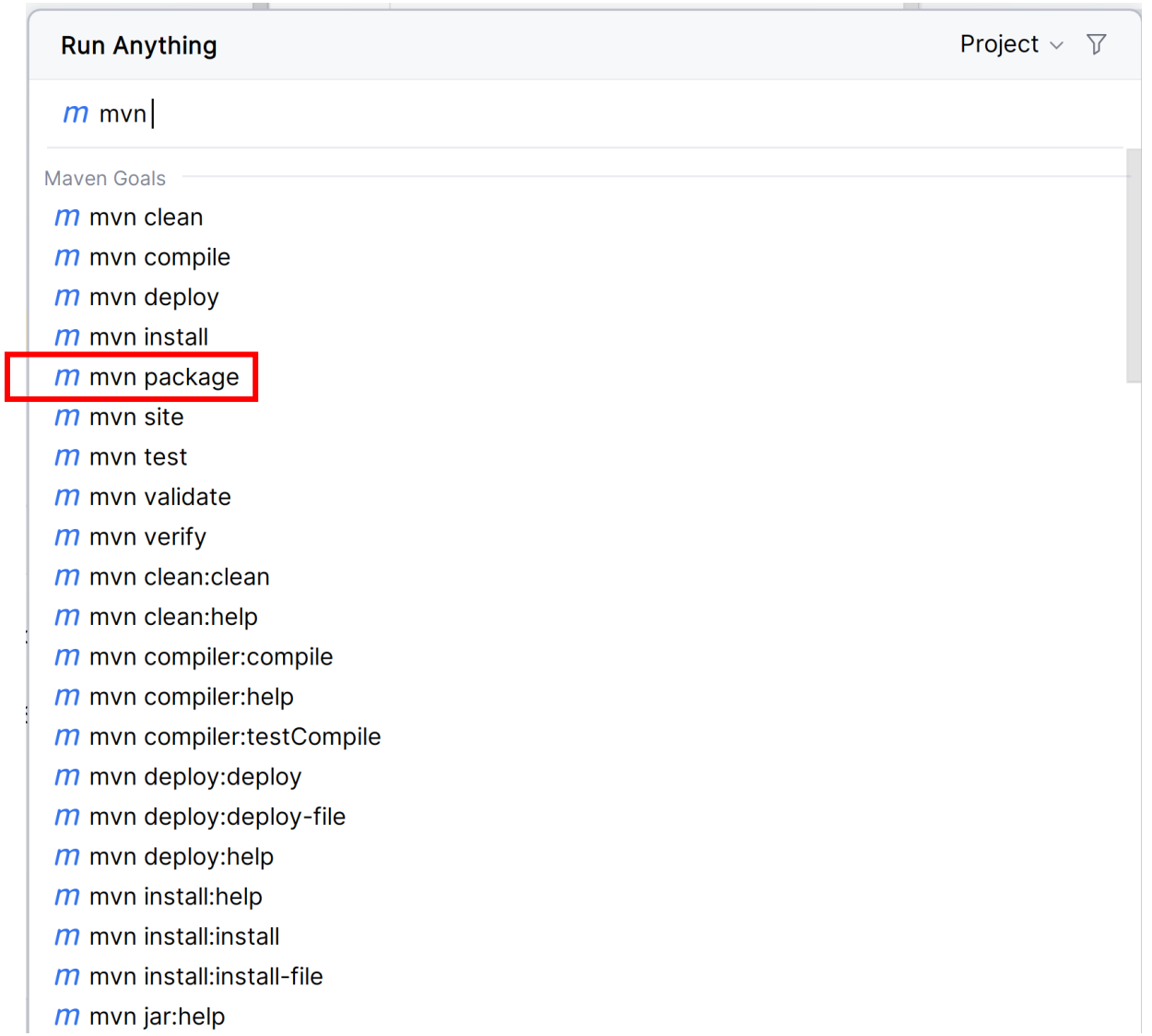



```
24/12/16 21:13:42 INFO TaskSchedulerImpl: Killing all running tasks in stage 2: Stage finished
24/12/16 21:13:42 INFO DAGScheduler: Job 1 finished: count at SparkHiveSqlApp.java:33, took 0,164527 s
addrDs count:26871372
Finished
```

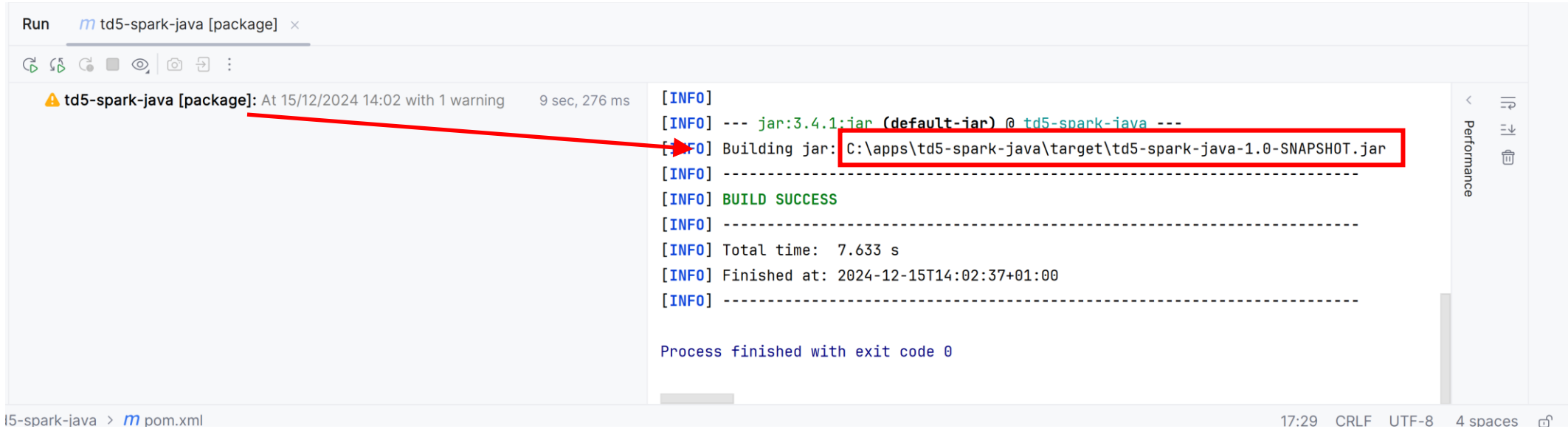
Launching from Terminal "spark-submit"

Build jar using "mvn package"





Built jar: "target/td5-spark-java-1.0-SNAPSHOT.jar"



spark-submit

cd td5-spark-java

spark-submit --class esilv.td5.SparkSqlApp target\td5-spark-java-1.0-SNAPSHOT.jar

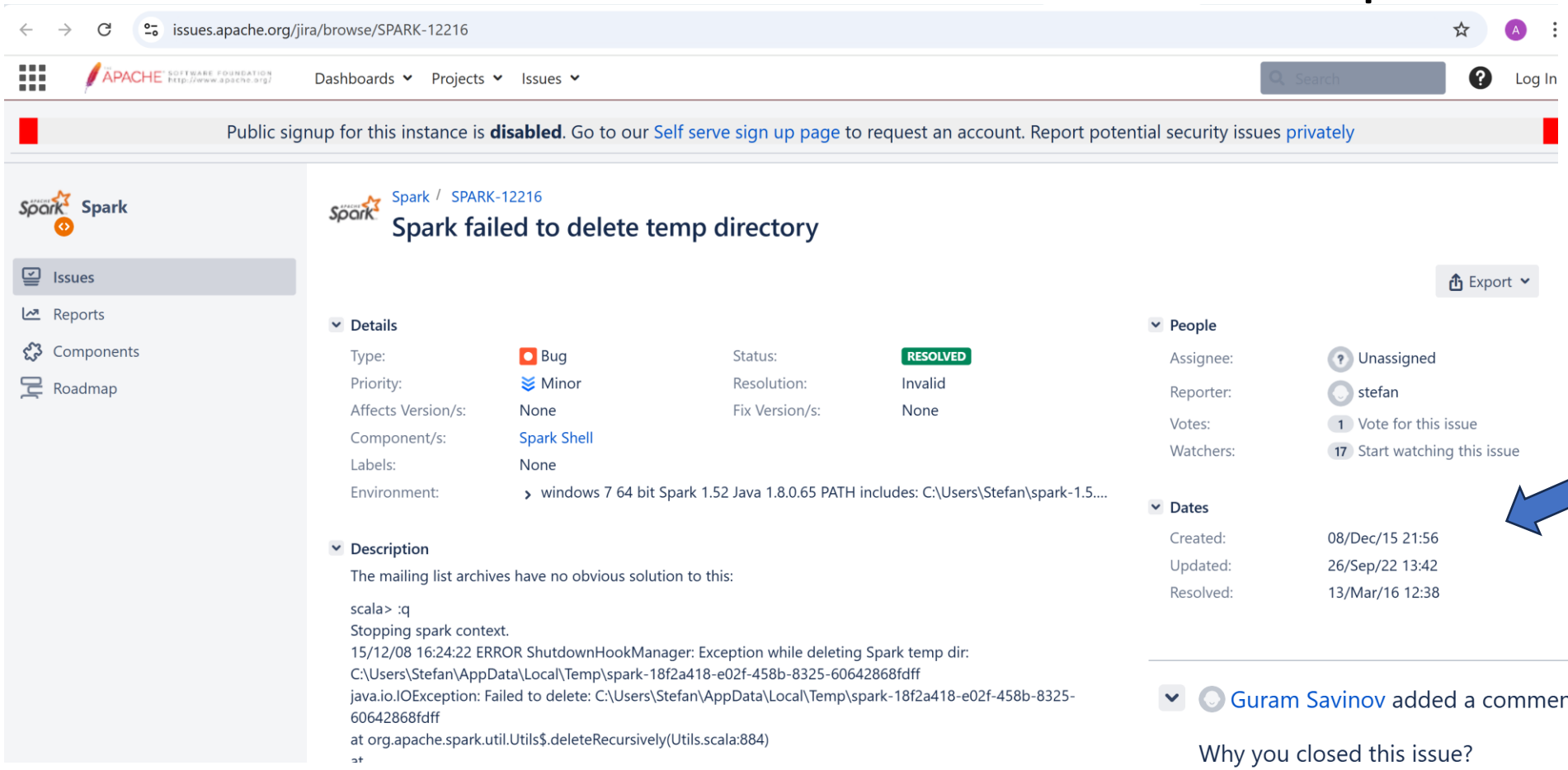
it "works"
... but on Windows, fails to "stop()" cleanly
(delete temp dir where jar locked)

```
C:\apps\td5-spark-java>spark-submit --class esilv.td5.SparkSqlApp target\td5-spark-java-1.0-SNAPSHOT.jar
line count:4
24/12/15 14:07:04 WARN SparkEnv: Exception while deleting Spark temp dir: C:\Users\arnaud\AppData\Local\Temp\spark-a17dec51-9765-4c05-94ba
-f96b0bcc4d27\userFiles-a3eb5451-c70d-489c-b351-b82bcedcd5fe
java.io.IOException: Failed to delete: C:\Users\arnaud\AppData\Local\Temp\spark-a17dec51-9765-4c05-94ba-f96b0bcc4d27\userFiles-a3eb5451-c7
0d-489c-b351-b82bcedcd5fe\td5-spark-java-1.0-SNAPSHOT.jar
        at org.apache.spark.network.util.JavaUtils.deleteRecursivelyUsingJavaIO(JavaUtils.java:146)
        at org.apache.spark.network.util.JavaUtils.deleteRecursively(JavaUtils.java:117)
        at org.apache.spark.network.util.JavaUtils.deleteRecursivelyUsingJavaIO(JavaUtils.java:129)
        at org.apache.spark.network.util.JavaUtils.deleteRecursively(JavaUtils.java:117)
        at org.apache.spark.network.util.JavaUtils.deleteRecursively(JavaUtils.java:90)
        at org.apache.spark.util.SparkFileUtils.deleteRecursively(SparkFileUtils.scala:121)
        at org.apache.spark.util.SparkFileUtils.deleteRecursively$(SparkFileUtils.scala:120)
        at org.apache.spark.util.Utils$.deleteRecursively(Utils.scala:1126)
        at org.apache.spark.SparkEnv.stop(SparkEnv.scala:108)
        at org.apache.spark.SparkContext.$anonfun$stop$25(SparkContext.scala:2310)
        at org.apache.spark.util.Utils$.tryLogNonFatalError(Utils.scala:1375)
        at org.apache.spark.SparkContext.stop(SparkContext.scala:2310)
        at org.apache.spark.SparkContext.stop(SparkContext.scala:2216)
        at org.apache.spark.sql.SparkSession.stop(SparkSession.scala:836)
        at esilv.td5.SparkSqlApp.main(SparkSqlApp.java:21)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
```

even with no explicit .stop()
always called by SparkShutdowHookManager

```
C:\apps\td5-spark-java>spark-submit --class esilv.td5.SparkSqlApp target\td5-spark-java-1.0-SNAPSHOT.jar
line count:4
24/12/15 14:46:35 WARN SparkEnv: Exception while deleting Spark temp dir: C:\Users\arnaud\AppData\Local\Temp\spa
-2a624c17ba5d\userFiles-90bbb73e-0e47-4b69-ae47-af7f7e2ad34d
java.io.IOException: Failed to delete: C:\Users\arnaud\AppData\Local\Temp\spark-fac59255-7d89-47b5-becd-2a624c17
47-4b69-ae47-af7f7e2ad34d\td5-spark-java-1.0-SNAPSHOT.jar
        at org.apache.spark.network.util.JavaUtils.deleteRecursivelyUsingJavaIO(JavaUtils.java:146)
        at org.apache.spark.network.util.JavaUtils.deleteRecursively(JavaUtils.java:117)
        at org.apache.spark.network.util.JavaUtils.deleteRecursivelyUsingJavaIO(JavaUtils.java:129)
        at org.apache.spark.network.util.JavaUtils.deleteRecursively(JavaUtils.java:117)
        at org.apache.spark.network.util.JavaUtils.deleteRecursively(JavaUtils.java:90)
        at org.apache.spark.util.SparkFileUtils.deleteRecursively(SparkFileUtils.scala:121)
        at org.apache.spark.util.SparkFileUtils.deleteRecursively$(SparkFileUtils.scala:120)
        at org.apache.spark.util.Utils$.deleteRecursively(Utils.scala:1126)
        at org.apache.spark.SparkEnv.stop(SparkEnv.scala:108)
        at org.apache.spark.SparkContext.$anonfun$stop$25(SparkContext.scala:2310)
        at org.apache.spark.util.Utils$.tryLogNonFatalError(Utils.scala:1375)
        at org.apache.spark.SparkContext.stop(SparkContext.scala:2310)
        at org.apache.spark.SparkContext.stop(SparkContext.scala:2216)
        at org.apache.spark.SparkContext.$anonfun$new$34(SparkContext.scala:686)
        at org.apache.spark.util.SparkShutdownHook.run(ShutdownHookManager.scala:214)
        at org.apache.spark.util.SparkShutdownHookManager.$anonfun$runAll$2(ShutdownHookManager.scala:188)
        at scala.runtime.java8.JFunction0$mcV$sp.apply(JFunction0$mcV$sp.scala:18)
        at org.apache.spark.util.Utils$.logUncaughtExceptions(Utils.scala:1928)
        at org.apache.spark.util.SparkShutdownHookManager.$anonfun$runAll$1(ShutdownHookManager.scala:188)
        at scala.runtime.java8.JFunction0$mcV$sp.apply(JFunction0$mcV$sp.scala:18)
        at scala.util.Try$.apply(Try.scala:210)
        at org.apache.spark.util.SparkShutdownHookManager.runAll(ShutdownHookManager.scala:188)
        at org.apache.spark.util.SparkShutdownHookManager$$anon$2.run(ShutdownHookManager.scala:178)
        at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:577)
        at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:317)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1144)
```

Notice: Lock considered as Windows bug ... will never be fixed in Spark (!?)

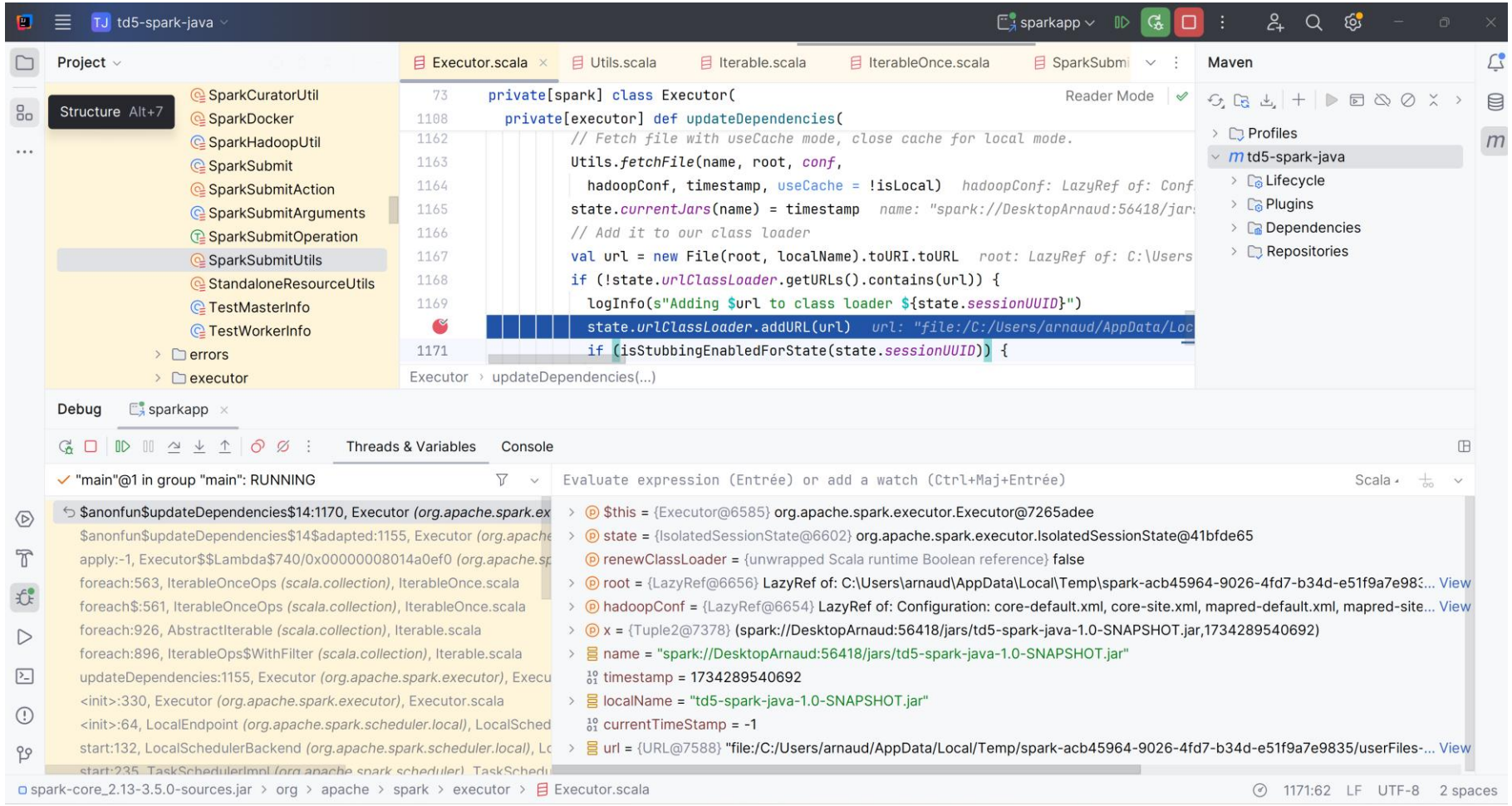


bug since 2015..

closed() with workaround "move to linux"

Curious why there are 2 jars in "local[*] mode?
1 in "driver" part
+ 1 locked (fetched) in "executor" part .. in Temp/

need JVM
remote debug
to see this ...



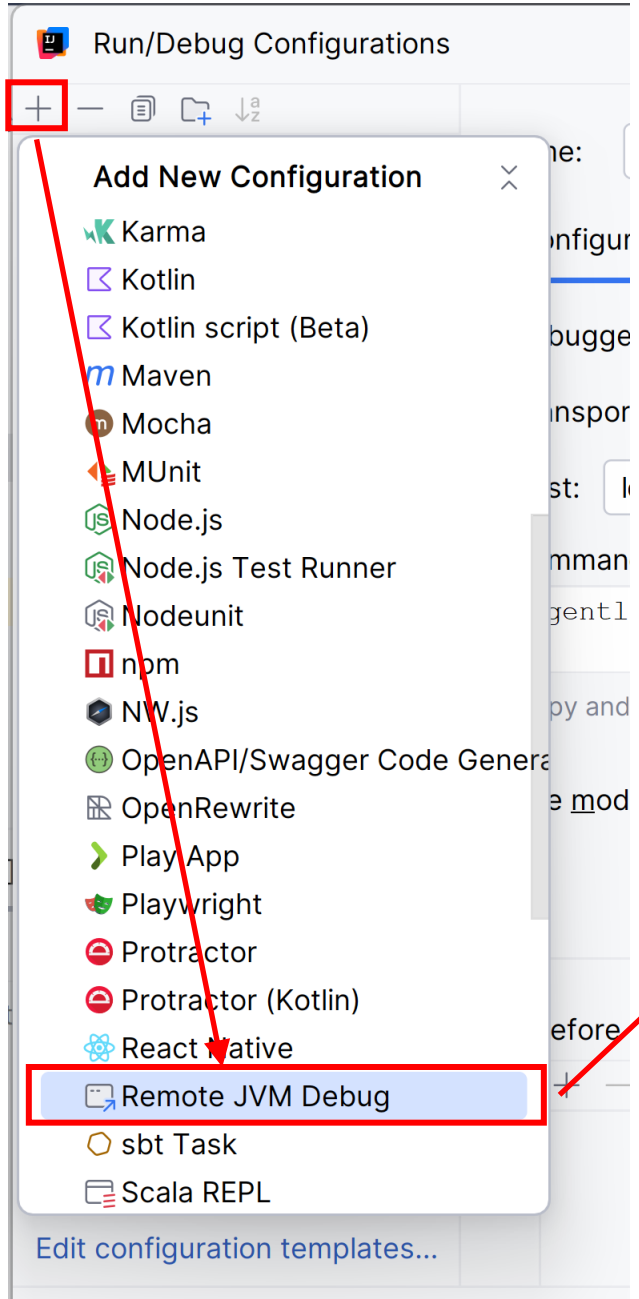
spark-submit with Remote JVM Debugging

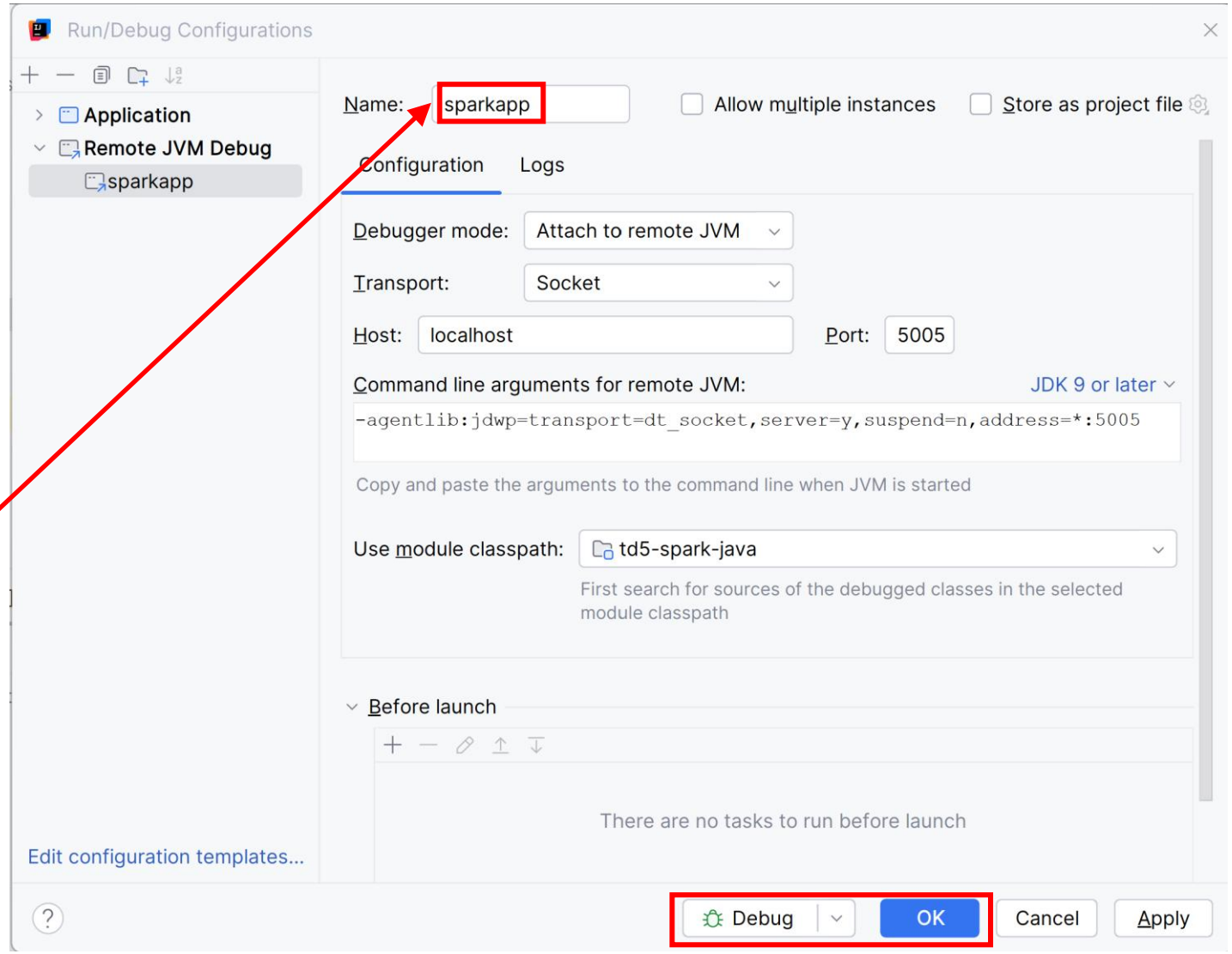
DEBUG_CONF=--conf spark.driver.extraJavaOptions=-agentlib:jdwp=transport=dt_socket,server=y,suspend=y,address=*:5005

spark-submit --master local[*] %DEBUG_CONF% --class esilv.td5.SparkSqlApp target/td5-spark-java-1.0-SNAPSHOT.jar

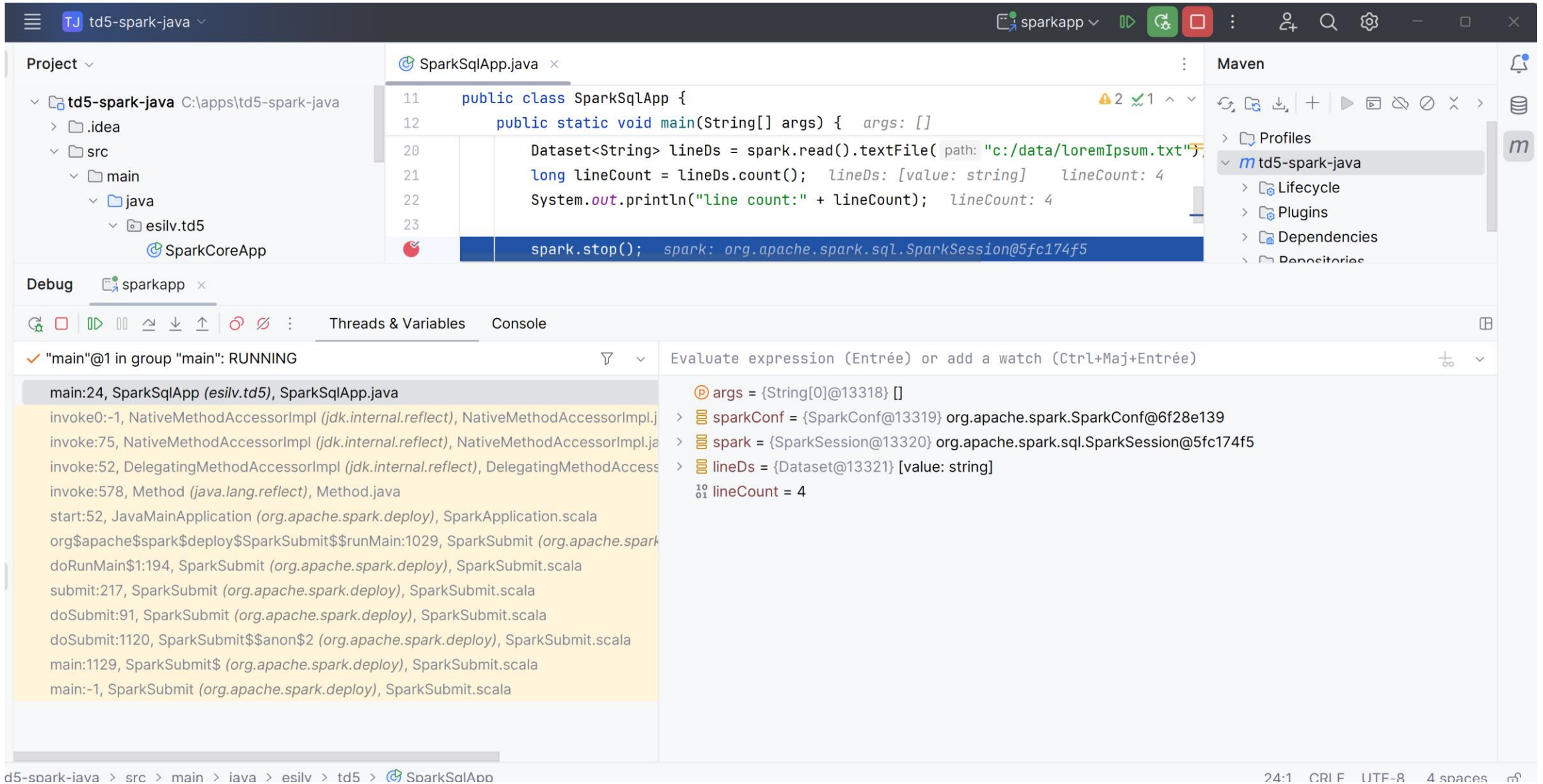
```
C:\apps\td5-spark-java>spark-submit --master local[*] --conf spark.driver.extraJavaOptions=-agentlib:jdwp=transport=dt_socket,server=y,sus
pend=y,address=*:5005 --class esilv.td5.SparkSqlApp target/td5-spark-java-1.0-SNAPSHOT.jar
Listening for transport dt_socket at address: 5005
```

Attaching Remote JVM Debugger





See main class is not "esilv.td5.SparkSqlApp" but "org.apache.spark.deploy.SparkSubmit"



Spark SQL <-> Java
with UDF Function

Exercise 2

The city name

"Laval"

written in **reversed order** is still

"laval"

(ignoring case)

This is a **palindrome**

How many French cities are like this?

Hint : Use Dataset .map() and .filter() and String .toLowerCase() and .reverse()

If you forgot to do ".toLowerCase()" => there would be only 1 City, named "Y"

in fact, there are 21

can you find them all ?

afa
callac
cazac
erre
esse
laval
noyon
...

Exercise Setup

spark.sql("SELECT DISTINCT commune_nom, commune_insee FROM db1.addr")
.write.format("parquet").saveAsTable("db1.city")

spark.sql("select count(*) from db1.city").show() // => 34986

spark.sql("SELECT distinct commune_nom FROM db1.city")
.write.format("parquet").saveAsTable("db1.city_name")

spark.sql("select count(*) from city_name").show() // => 32735 ... almost 2000 duplicate city name in France !

OK, this is NOT BigData... we could "collectAsList()" and perform directly in scala

"String.reverse()" is Java built-in => redo with SQL built-in

Reapeat exercise in SQL (not using Spark Dataset API),

```
spark.sql("""
   select commune_nom from db1.city
   WHERE reverse(lower(commune_nom))=lower(commune_nom)
""").show()
```

Too easy again...

Using SQL and User-Defined Function "UDF"

.. Reapeat without using built-in Sql "REVERSE",
by defining your own UDF function "isPalindrome(text)"

```
SELECT * FROM  db1.city
WHERE    isPalindrome(lower(commune_nom))
```

complete code:
spark.udf.register("isPalindrome", (s: String) => ....)

Exercise

City names

"**Sancerre**" and "**Nanterre**"

have only 2 letters difference.

The postman could do mistakes...

How many cities are error-prone like this.
What is the city with the most numerous errors, for 1 letter of difference?

What is an "Edit Distance" ?

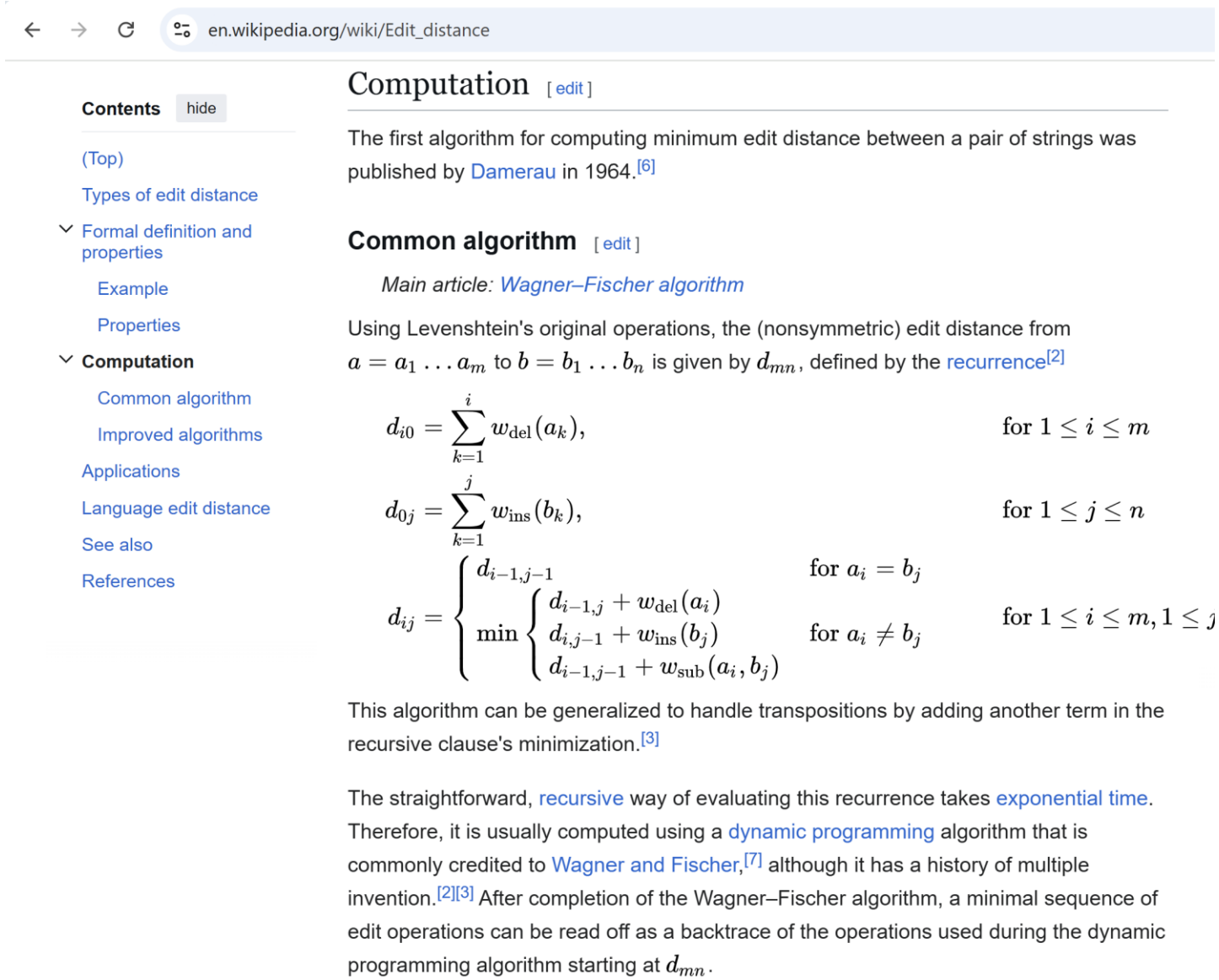
how many letters differ between "paris" and "parés"
 => Response : 1 (change of 1 letter "i" -> "é")

how many letters differ between "paris" and "pris"
 => Response : 1 (deletion of 1 letter "a")

Calculating "changed" letters only is trivial (but not in SQL).

How do you compute efficiently distances for "deletion" + "change" + "insertion" ?
... even in Java, it need special algorithm

Edit Distance with Dynamic Programming Algorithm



Levenstein Distance

en.wikipedia.org/wiki/Levenshtein_distance

WIKIPEDIA
The Free Encyclopedia

Search Wikipedia | Search | Do

Levenshtein distance

36 languages

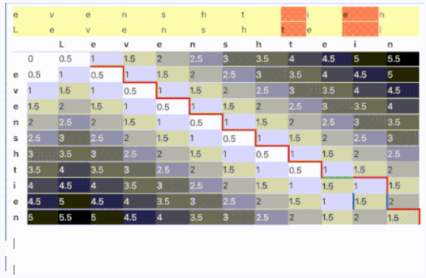
Article | Talk | Read | Edit | View history | Tools

Contents hide

From Wikipedia, the free encyclopedia

In information theory, linguistics, and computer science, the **Levenshtein distance** is a string metric for measuring the difference between two sequences. The Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. It is named after Soviet mathematician Vladimir Levenshtein, who defined the metric in 1965.[1]

Levenshtein distance



Edit distance matrix for two words using cost of substitution as 1 and cost of deletion or insertion as 0.5

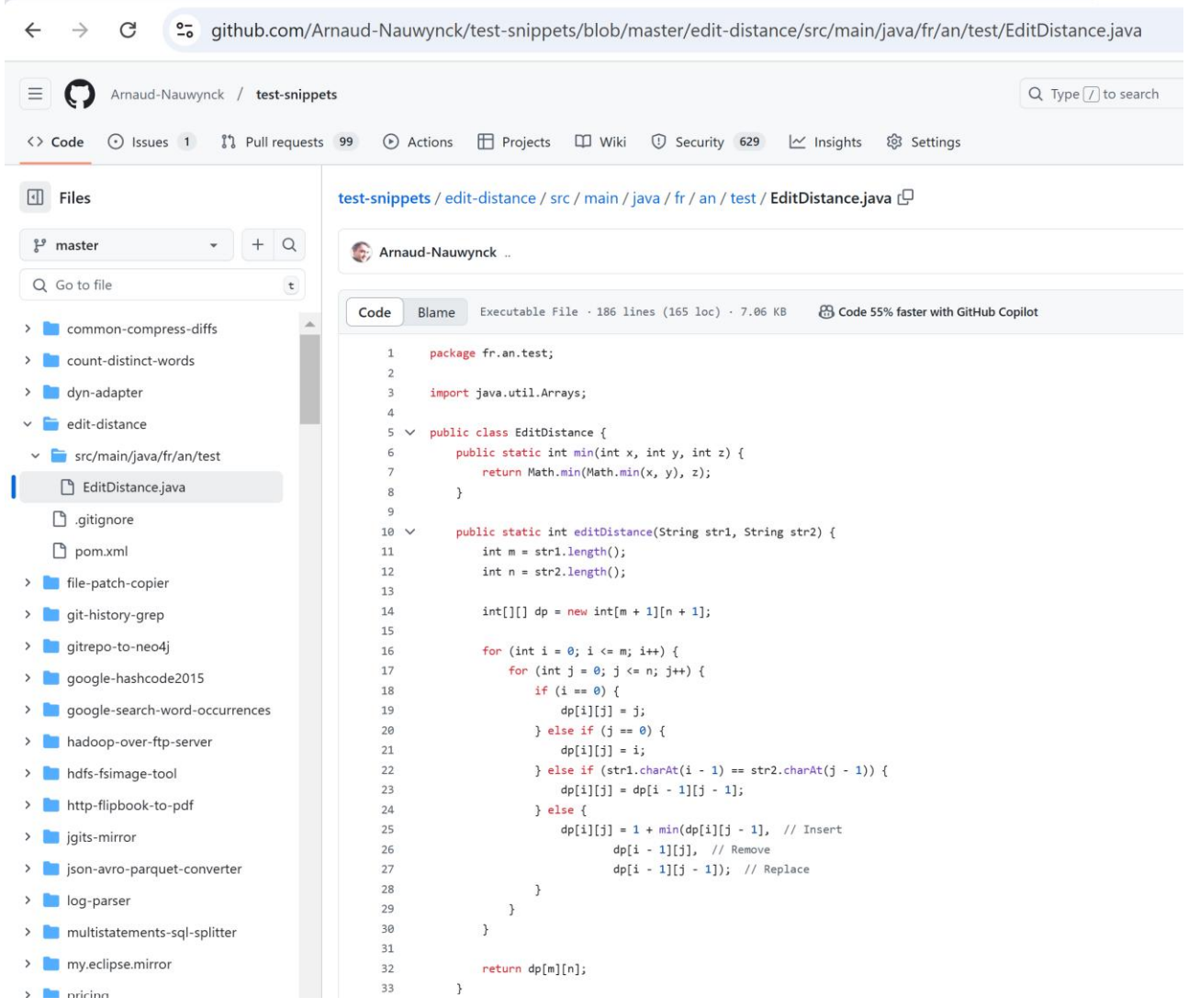
Class measuring the difference between two sequences

Levenshtein distance may also be referred to as *edit distance*, although that term may also denote a larger family of distance metrics known collectively as edit distance.[2]:32 It is closely related to pairwise string alignments.

EditDistance.java Implementation

source code (extracted from apache Commons Lang)

https://github.com/Arnaud-Nauwynck/test-snippets/blob/master/edit-distance/src/main/java/fr/an/test/EditDistance.java



Download jar (or rebuild "mvn package") + relaunch "spark-shell --jars your.jar"

spark-shell --driver-memory=8g --jars edit-distance.jar

or using full path

spark-shell --driver-memory=8g --jars file:///c:/apps/spark/edit-distance.jar

or simply copy file in ${SPARK_HOME}/jars/

Test Class is available in spark-shell scala>

Test in interactive shell that you have <TAB><TAB> Auto-completion
on "fr."
=> autocomplete to package "fr.an"
..
<TAB><TAB>
=> autocomplete to package "fr.an" ... then to class then to methods

```
scala>

final package an

final package an

scala> fr.an.test.EditDistance.editDistance
editDistance(              editDistanceLimitedTo(
```

Test Java methods (in scala)

```
fr.an.test.EditDistance.editDistance("paris", "pris")
// 1



// the variant "LimitedTo" stop computing for a given threshold, and return it as maximum diff
fr.an.test.EditDistance.editDistanceLimitedTo("paris", "pris", 5)
// 1

fr.an.test.EditDistance.editDistanceLimitedTo("paris", "prisées", 10)
// 4

fr.an.test.EditDistance.editDistanceLimitedTo("paris", "prisées", 3)
// 3  ... real diff was 4, but result limited to 3
```

Define UDF functions, Test in SQL

```
spark.udf.register("editDistance", udf( (str1: String, str2: String) =>
fr.an.test.EditDistance.editDistance(str1, str2 )));

spark.udf.register("editDistanceLimitedTo", udf( (str1: String, str2: String, m: Int) =>
fr.an.test.EditDistance.editDistanceLimitedTo(str1, str2, m)));



spark.sql("select editDistance('paris', 'pris')").show()
// 1

spark.sql("select editDistanceLimitedTo('paris', 'prisées', 3)").show()
// 3
```

callUDF() : call using Dataset Column API

val ds2 = ds.withColumn("dist_to_paris", **callUDF("editDistance"**, lit("paris"), $"commune_nom"**)**)

```
scala> ds2.show(5, false)
+----------------------+----------------+
|commune_nom    |dist_to_paris|
+----------------------+----------------+
|Frans                 |4               |
|Coulonges-Cohan|14              |
|Grisolles             |7               |
|Leury                 |5               |
|Deux-Chaises      |10              |
+----------------------+----------------+
```

NOTE: calling explicitly in java

```
import org.apache.spark.sql.functions;
   functions.callUDF("editDistance", functions.lit("paris"),  col(commune_nom"))
```

cross join city1, city2
restrict with city1 < city2
eval d=distanceLimitedTo (.. 2)
filter d<2

what is the algorithm complexity for N cities ?

spark.sql("select count(*) from db1.city").show()   // 34986
using distinct name? 32735

34986 * 34986 / 2 = 612010098   = 612 Millions of city pairs to eval

and each computation, this may take long => so restrict to maximum 2 differences,
we will consider only diff=1 (<2)

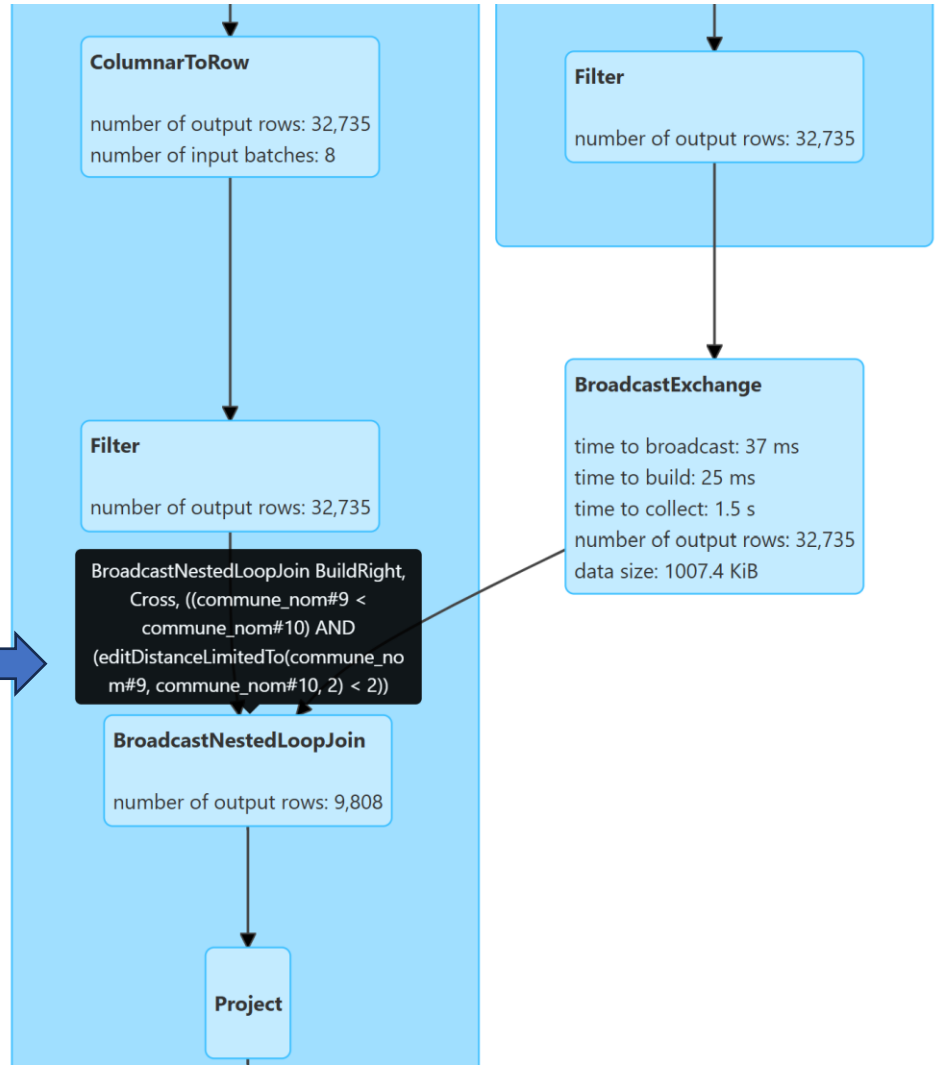
(On Intel Core i7, it takes ~3 minutes)

Spill to Disk ...
use --driver-memory=8g

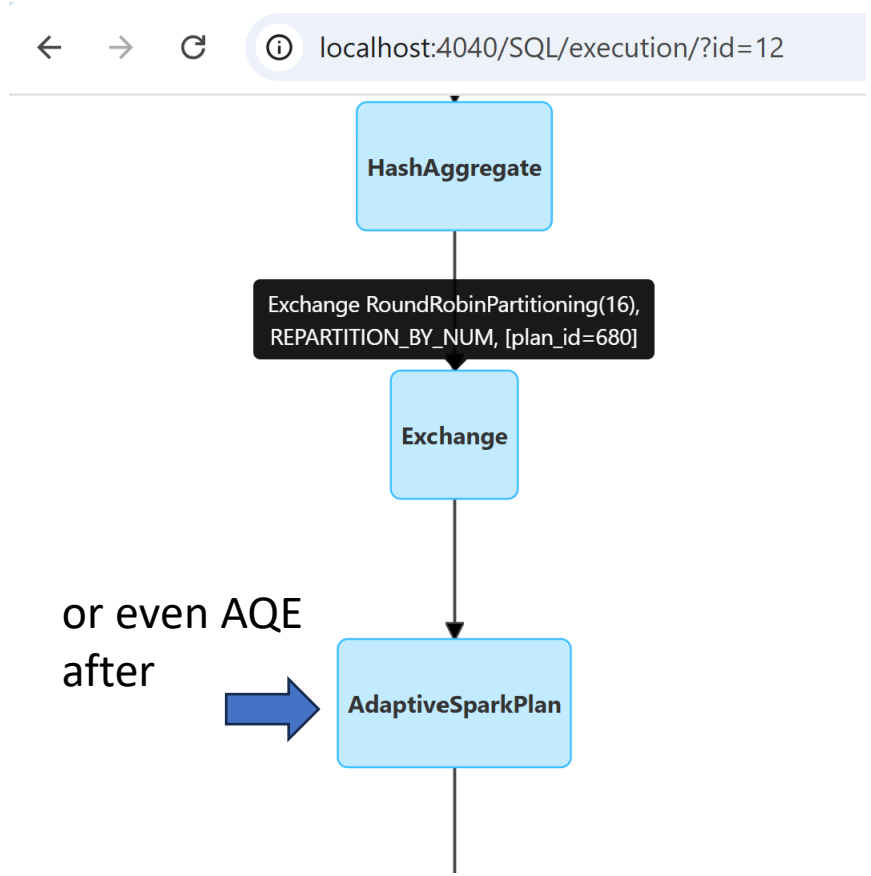
```
scala> crossDist.show(10, false)
24/12/09 01:28:48 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:28:48 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:29:30 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:29:30 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:31:02 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:31:02 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:32:37 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:32:38 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:34:12 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:34:12 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
```

.repartition(16) IGNORED !!
=> only using 1 CPU

push down
in broadcast join !



or even AQE
after



Cross City Pairs with Distance=1

```
scala> crossDist.count()
val res4: Long = 9808
```

```
scala> crossDist.show(10, false)
+-------------------------+-------------------------+---+
|c1_nom                   |c2_nom                   |d  |
+-------------------------+-------------------------+---+
|Auge                     |Augy                     |1  |
|Many                     |Mary                     |1  |
|Lunan                    |Lunay                    |1  |
|Caunay                   |Launay                   |1  |
|Ballons                  |Baâlons                  |1  |
|Sassay                   |Sassy                    |1  |
|Marseille 2e Arrondissement|Marseille 9e Arrondissement|1
|
|Saisy                    |Saizy                    |1  |
|Amécourt                 |Imécourt                 |1  |
|Audon                    |Auxon                    |1  |
+-------------------------+-------------------------+---+
only showing top 10 rows
```

Find the Cities having the most similarities Can you find this ?

c1_nom	count
Bailly	13
Cailly	12
Bessac	10
Aron	10
Bay	10
Paris 10e Arro...	10
Coux	10
Maray	10
Aigny	10
Lailly	10

only showing top 10 rows

13 Cities with Distance=1 to "Bailly"

```
spark.sql("""select c1_nom, d from (select c1.commune_nom as c1_nom,
editDistanceLimitedTo(c1.commune_nom, "Bailly", 2) as d from db1.city_name c1 where c1.commune_nom <>
"Bailly") where d < 2 ORDER BY d """).show(50);
```

```
+-------+---+
| c1_nom|  d|
+-------+---+
| Vailly|  1|
| Nailly|  1|
| Sailly|  1|
| Jailly|  1|
| Mailly|  1|
| Lailly|  1|
| Cailly|  1|
|  Billy|  1|
| Pailly|  1|
| Tailly|  1|
| Wailly|  1|
| Failly|  1|
|Batilly|  1|
|Bacilly|  1|
+-------+---+
```

12 Cities with Distance=1 to "Bessac"

```
spark.sql("""select c1_nom, d from (select c1.commune_nom as c1_nom,
editDistanceLimitedTo(c1.commune_nom, "Bessac", 2) as d from db1.city_name c1 where c1.commune_nom <>
"Bessac") where d < 2 ORDER BY d """).show(50);
```

```
+-------+---+
| c1_nom|  d|
+-------+---+
| Pessac|  1|
| Bessay|  1|
| Messac|  1|
| Bessan|  1|
| Bassac|  1|
|Beyssac|  1|
|Blessac|  1|
| Cessac|  1|
| Bessas|  1|
| Lessac|  1|
| Bussac|  1|
+-------+---+
```

Exercise : UDAF (!= UDF)

User-Defined Aggregate Function

Example of "Aggregate" function : min(), max(), count(), sum(), avg(), stddev(), ...

They are not really "function", but more "statefull accumulator"

Exercise : find the most "round" / "elongated" shape cities

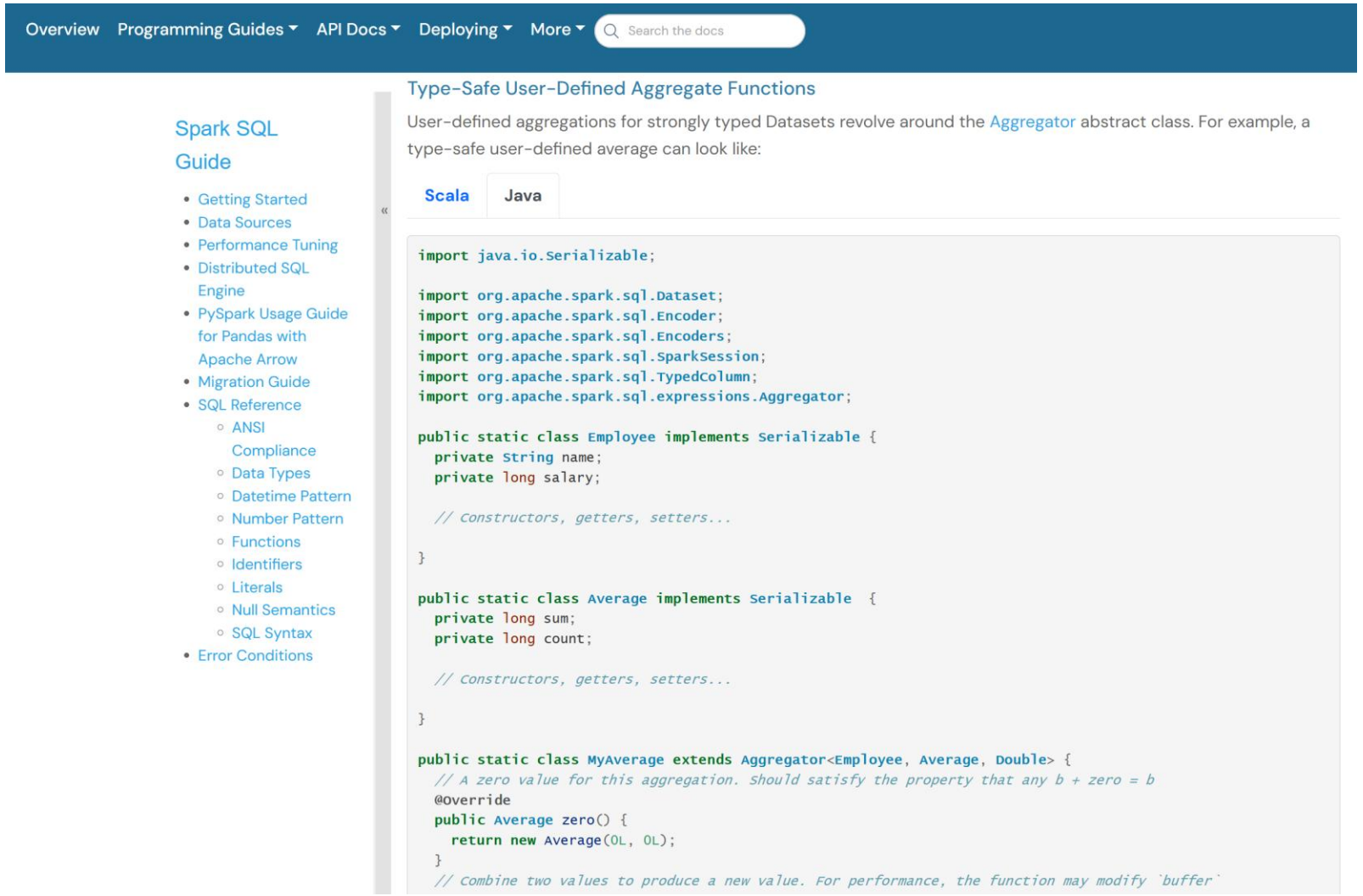
define a UDAF for accumulating points coordinate statistics
(sum x^2, sum x*y, sum y^2)
=> internally add to current covariance Matrix state

=> on finish, decompose Matrix with 2 eigen vectors

=> return ..

see Doc UDAF

https://spark.apache.org/docs/3.5.3/sql-ref-functions-udf-aggregate.html



Aggregator<IN,BUFFER,OUT>

IN => named_struct "Point"

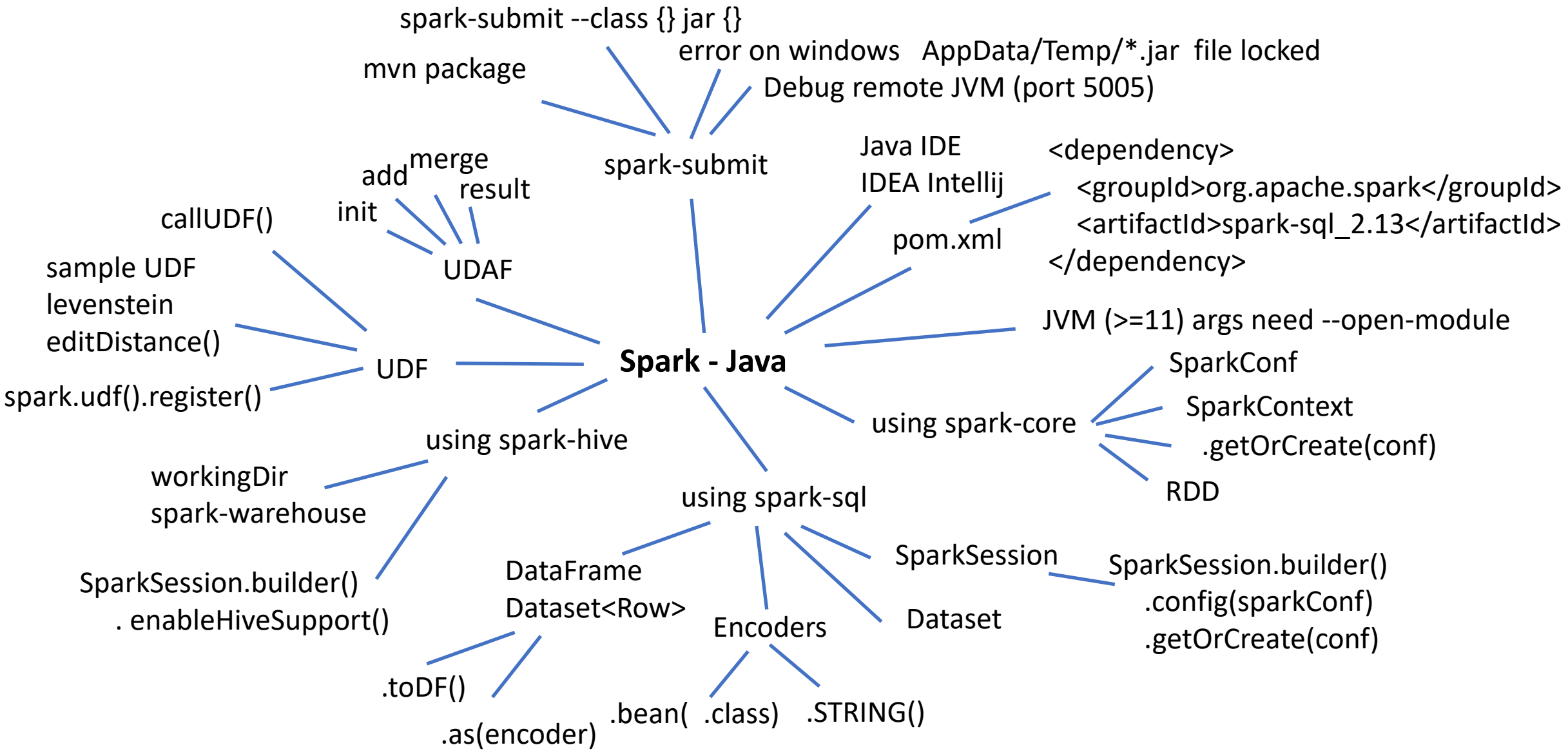
BUFFER => internal sums + number of point

OUT => named_struct for matrix

exercise left to students ...

Exercise :
draw a MindMap of this Hands-on session

MindMap


spark-submit --class {} jar {}
mvn package
error on windows AppData/Temp/*.jar file locked
Debug remote JVM (port 5005)
spark-submit
Java IDE
IDEA Intellij
<dependency>
<groupId>org.apache.spark</groupId>
<artifactId>spark-sql_2.13</artifactId>
</dependency>
pom.xml
add
merge
result
init
UDAF
callUDF()
sample UDF
levenstein
editDistance()
UDF
spark.udf().register()
JVM (>=11) args need --open-module
Spark - Java
SparkConf
SparkContext
.getOrCreate(conf)
RDD
using spark-core
using spark-hive
workingDir
spark-warehouse
SparkSession.builder()
. enableHiveSupport()
using spark-sql
DataFrame
Dataset<Row>
.toDF()
.as(encoder)
Encoders
.bean(.class)
.STRING()
SparkSession
Dataset
SparkSession.builder()
.config(sparkConf)
.getOrCreate(conf)

Questions ?