

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>

The screenshot shows the official JetBrains website for IntelliJ IDEA. The URL in the address bar is [jetbrains.com/idea/download/?section=windows](https://www.jetbrains.com/idea/download/?section=windows). The page has a dark header with the JetBrain logo, navigation links like Developer Tools, Team Tools, Education, Solutions, Support, Store, and a search bar. Below the header, there's a navigation bar with IntelliJ IDEA, JetBrains IDEs, What's New, Features, Resources, Pricing, and a prominent blue Download button. The main content area is for Windows, showing the IntelliJ IDEA Ultimate logo and the text "The Leading Java and Kotlin IDE". It features two download buttons: "Download" and ".exe (Windows) ▾", with the second one being highlighted with a red box. Below these buttons is a "Free 30-day trial" link. To the right, there's a screenshot of the IntelliJ IDEA interface showing a project structure and some Java code. At the bottom, there are links for Version: 2024.3.1, System requirements, Other versions, Installation instructions, and Third-party software.

jetbrains.com/idea/download/?section=windows

JETBRAINS

Developer Tools Team Tools Education Solutions Support Store

IntelliJ IDEA JetBrains IDEs What's New Features Resources Pricing Download

Windows macOS Linux

IntelliJ IDEA Ultimate
The Leading Java and Kotlin IDE

Download .exe (Windows) ▾

Free 30-day trial

Version: 2024.3.1 System requirements Other versions
Build: 243.22562.145 Installation instructions Third-party software
9 December 2024

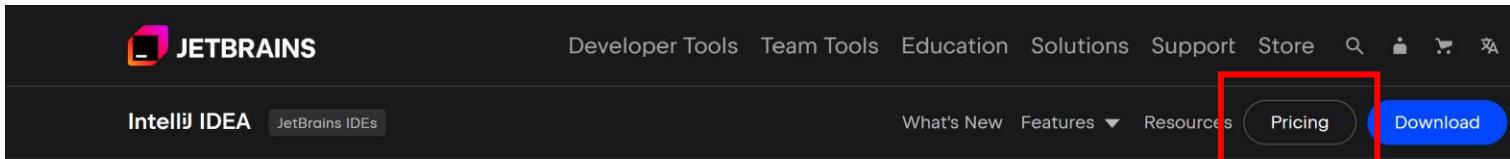
Note : pre-requisites

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

This screenshot shows the 'Subscription Options and Pricing' page. It features three main categories: 'For Organizations', 'For Individual Use', and 'Special Categories'. Below these are two billing options: 'Yearly billing save 2 months' and 'Monthly billing'. Two products are listed: 'IntelliJ IDEA Ultimate' and 'All Products Pack'. A red arrow points from the 'Special Categories' button on the left to the 'Special Categories' button on the right.

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

This screenshot shows the 'Subscription Options and Pricing' page with several special categories listed. The 'Special Categories' button is highlighted with a red box. Other categories include 'For students and teachers' (highlighted with a red box), 'For classroom assistance', 'For Open Source projects', 'For training courses, coding schools, and bootcamps', and 'For startups'. Red arrows point from the 'Special Categories' button on the left to the 'Special Categories' button on the right, and from the 'For students and teachers' section to its 'Learn more' link.

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

Subscription Options and Pricing

This screenshot shows the 'Subscription Options and Pricing' page with detailed descriptions for each special category. The 'Special Categories' button is highlighted with a red box. The categories and their descriptions are:

- 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 training courses, coding schools, and bootcamps FREE**: Private software development companies which
- For universities and educational organizations 50% off**: Private software development companies which
- For startups 50% off**: Private software development companies which

Steps ...

The diagram illustrates the process of applying for a JetBrains educational license. It starts with the 'Free educational licenses' page on the left, which includes a red box around the 'Apply now' button. A red arrow points from this button to the 'Apply now' button on the right. The right side shows the 'JetBrains Products for Learning' application form, where the 'University email address' field contains '@esilv'.

Community Programs jetbrains.com/community/education/#students

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

JetBrains Products for Learning

Before you apply, please read the [Educational Subscription Terms and FAQ](#).

Apply with: University email address

Status: I'm a student I'm a teacher

Country / region: France

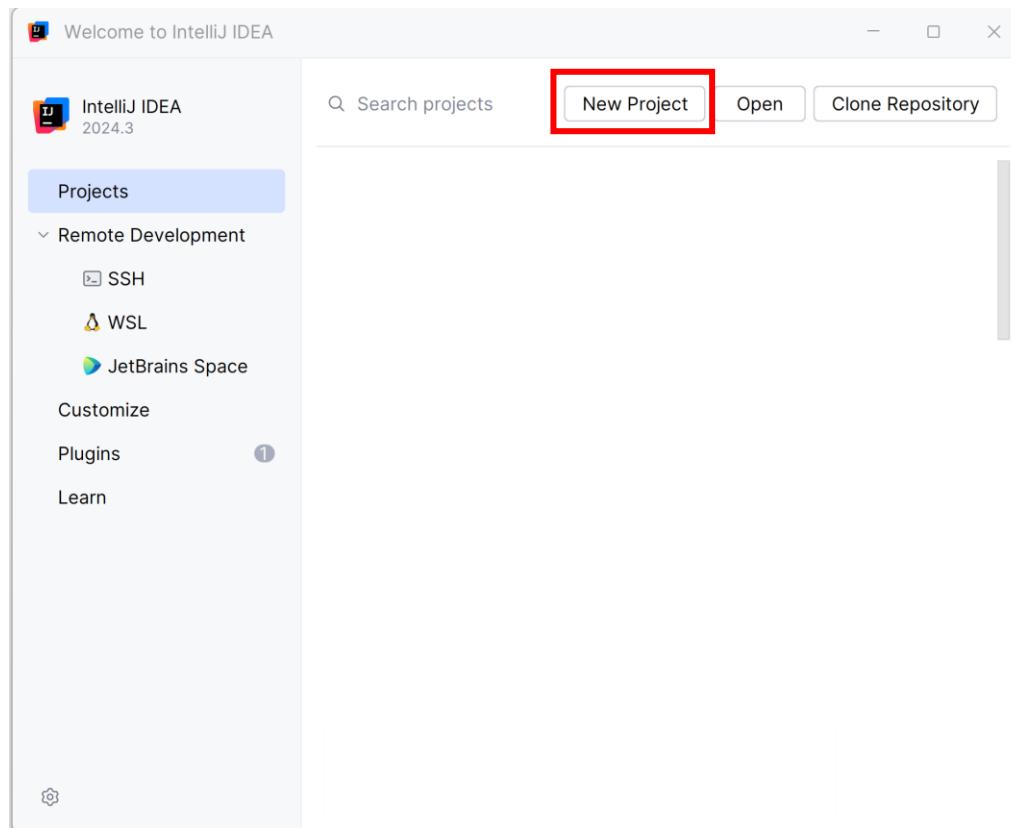
Level of study: Undergraduate

Is Computer Science or Engineering your major field of study?
 Yes No

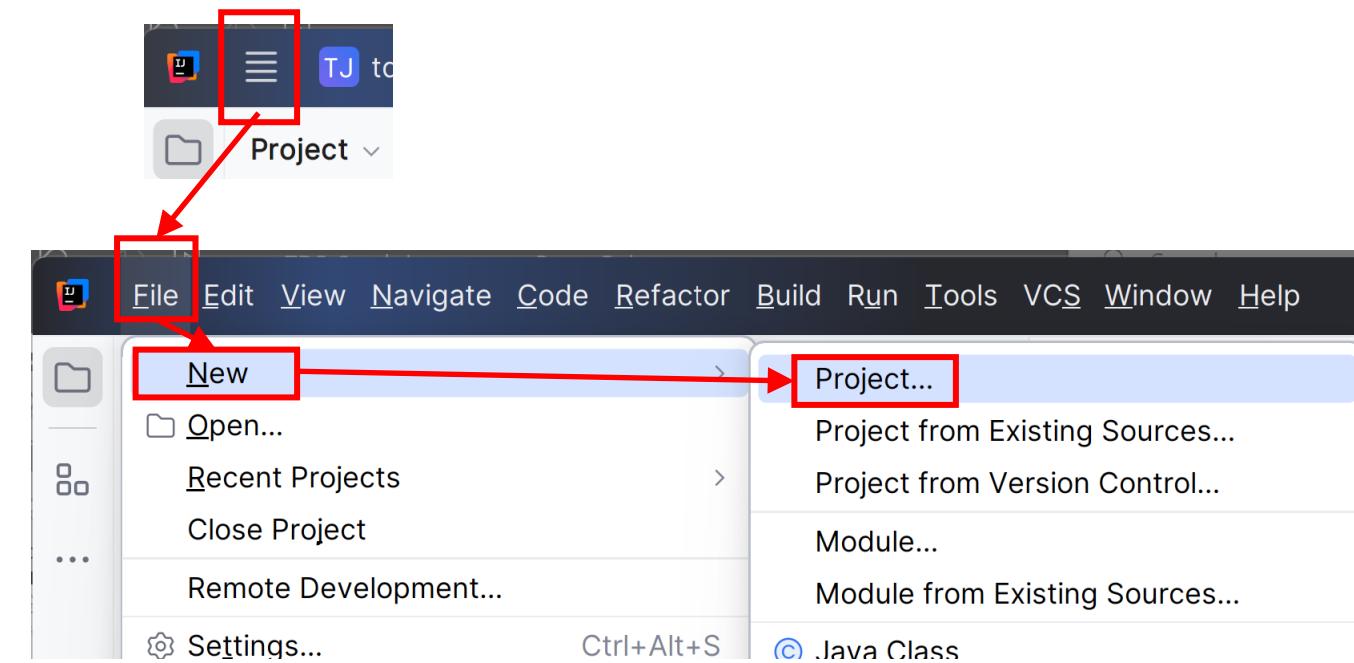
Email address: University email address @esilv

Create a New Project

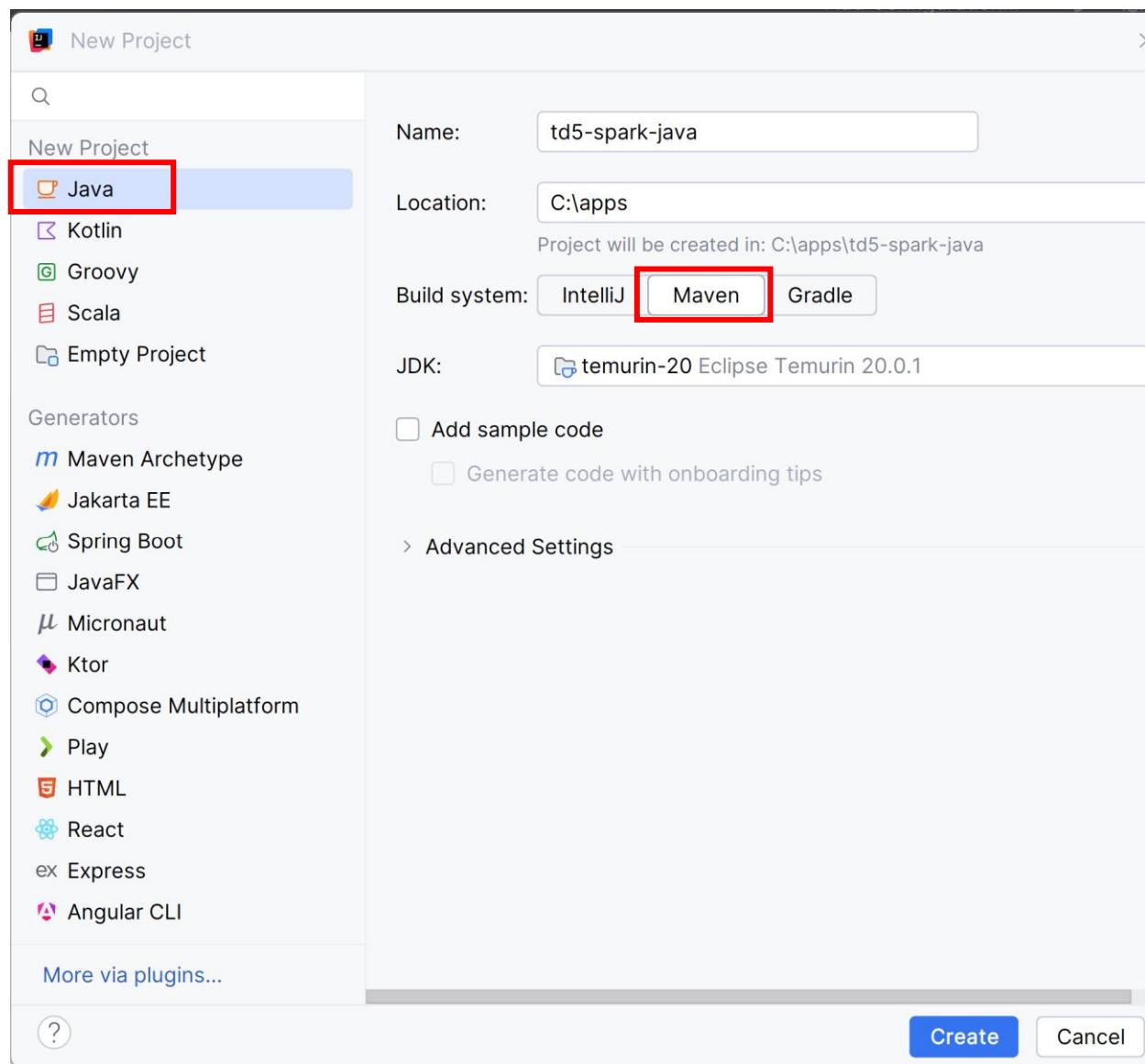
When opening the first time (no current project)



When re-opening (already a current project)

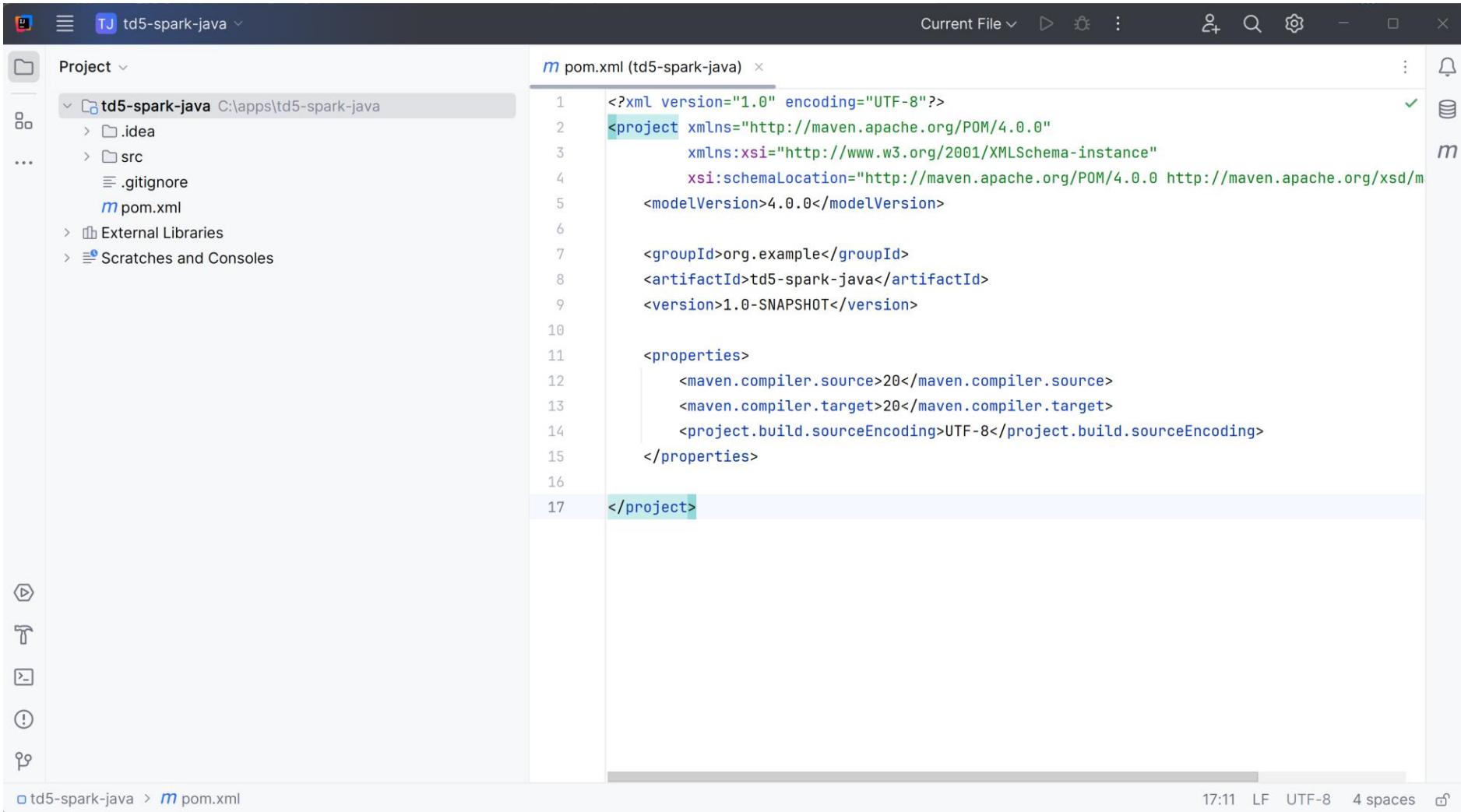


Create New Java (build with Maven) Project



pom.xml file

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

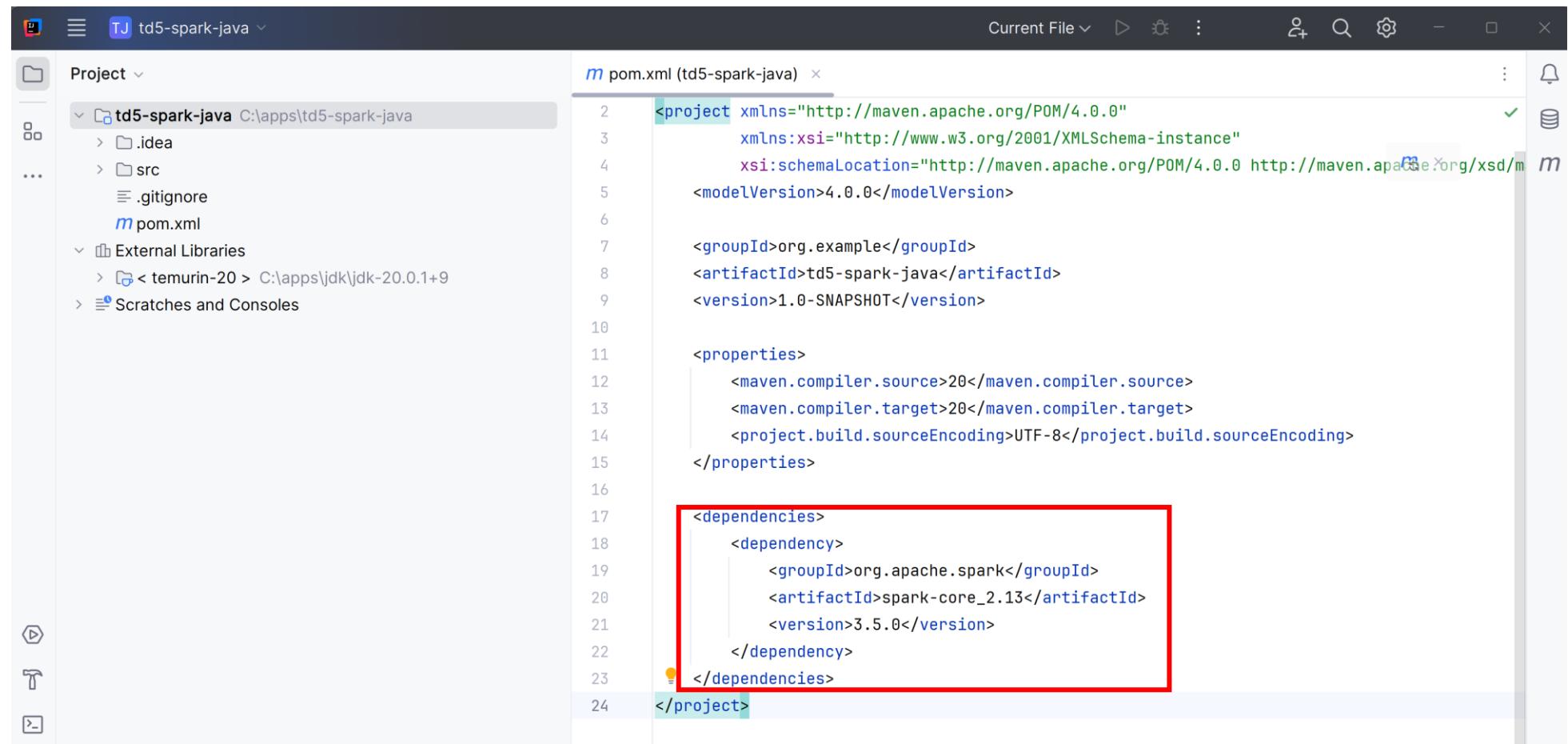


The screenshot shows a Java development environment with a project named "td5-spark-java". The project structure on the left includes ".idea", "src", ".gitignore", and "pom.xml". The "pom.xml" file is open in the main editor window, displaying its XML content. The code is color-coded for syntax highlighting.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>td5-spark-java</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <maven.compiler.source>20</maven.compiler.source>
        <maven.compiler.target>20</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
</project>
```

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>
```

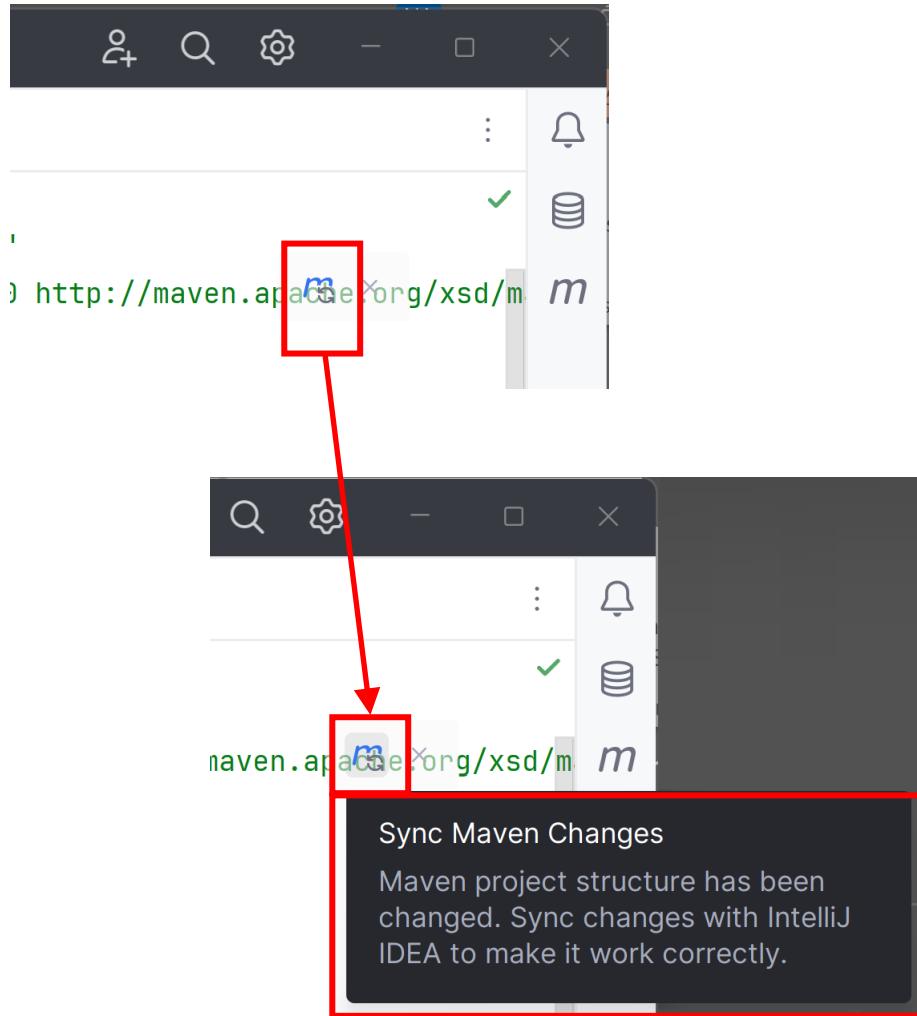


The screenshot shows an IDE interface with a project named "td5-spark-java" selected in the left sidebar. The main editor window displays the `pom.xml` file. A red box highlights the `<dependencies>` section, which contains the dependency configuration shown in the code block above. The rest of the `pom.xml` file includes standard Maven project metadata like group ID, artifact ID, version, and properties.

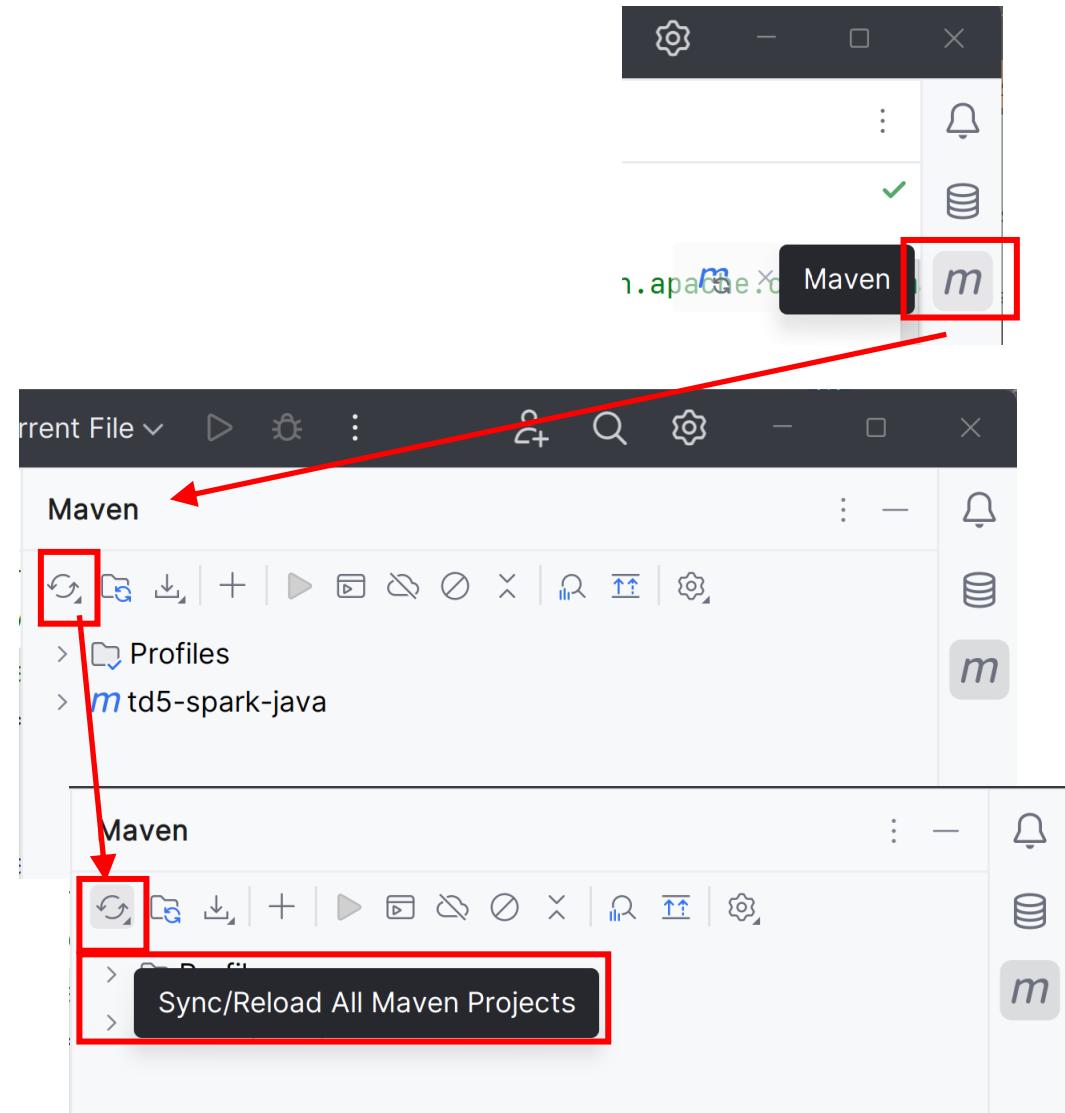
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.example</groupId>
  <artifactId>td5-spark-java</artifactId>
  <version>1.0-SNAPSHOT</version>
  <properties>
    <maven.compiler.source>20</maven.compiler.source>
    <maven.compiler.target>20</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.apache.spark</groupId>
      <artifactId>spark-core_2.13</artifactId>
      <version>3.5.0</version>
    </dependency>
  </dependencies>
</project>
```

Then Click on "Maven Sync" icon

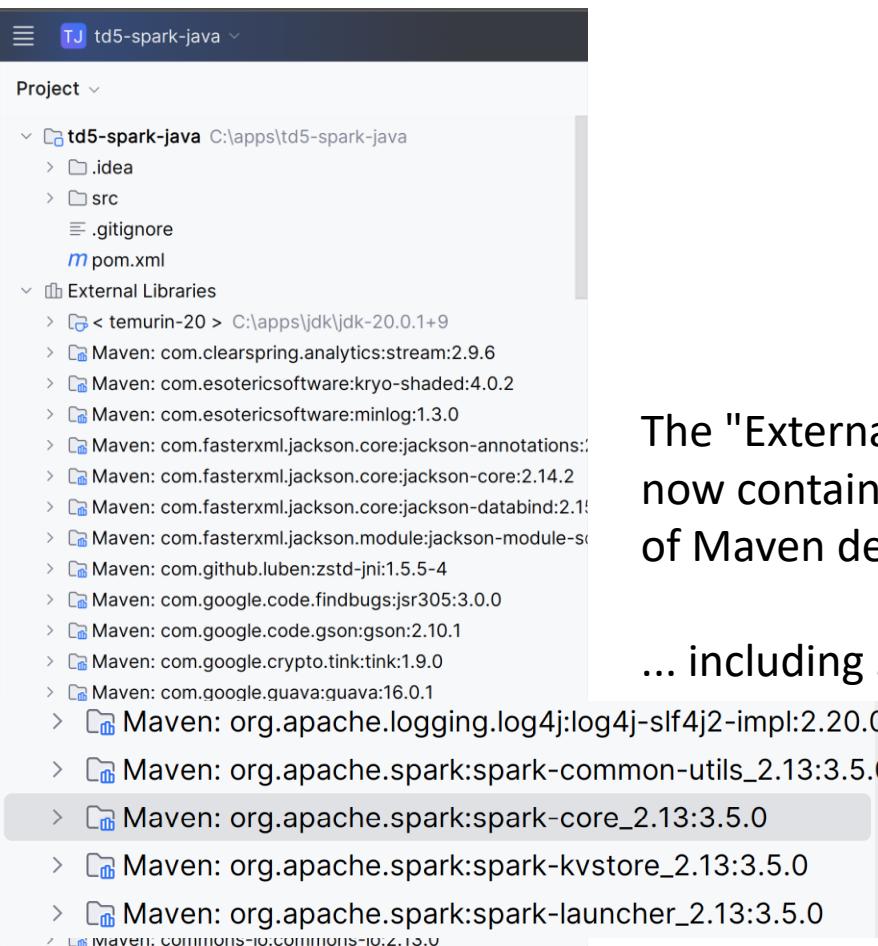
difficult to see inside text editor



can also do from right "m"=Maven button

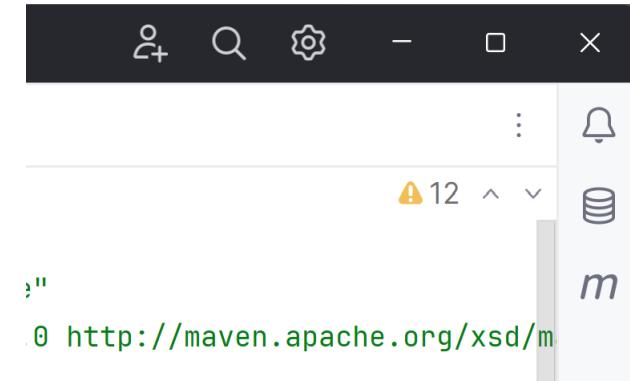


Maven will download dependencies ... can take ~5mn



The "External Libraries"
now contains hundreds
of Maven dependencies

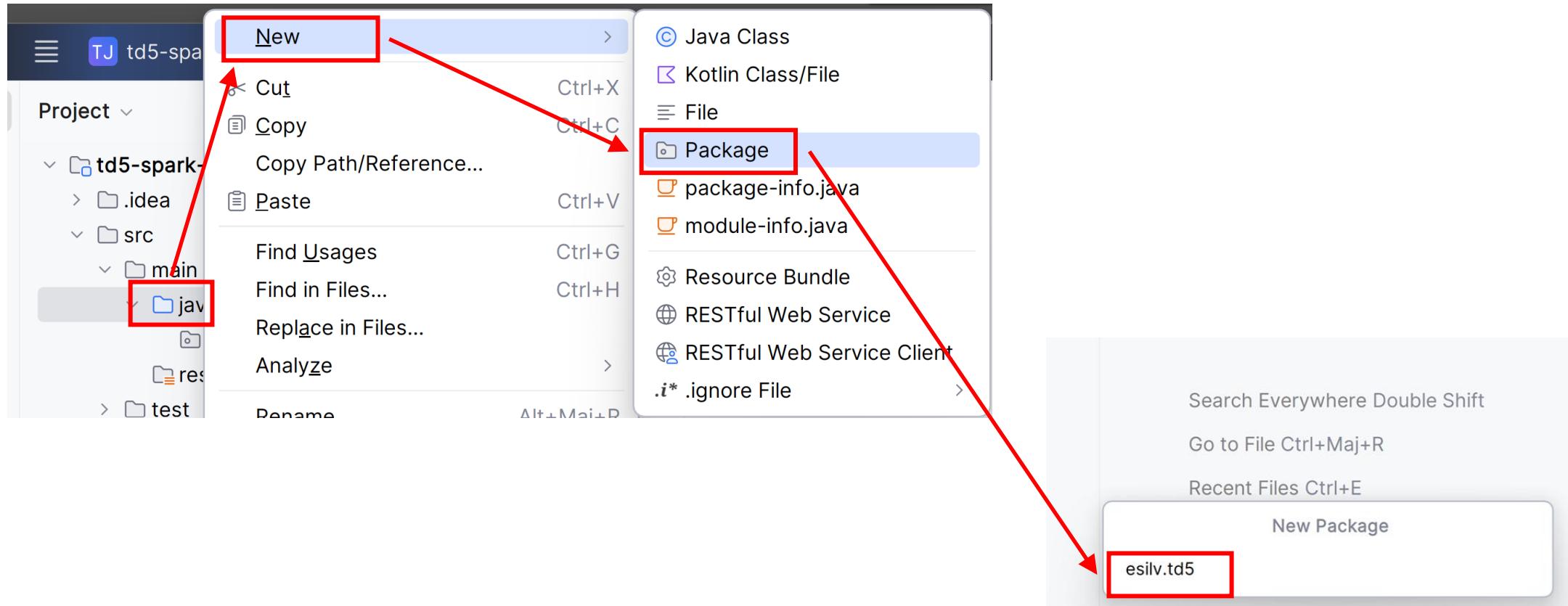
... including spark



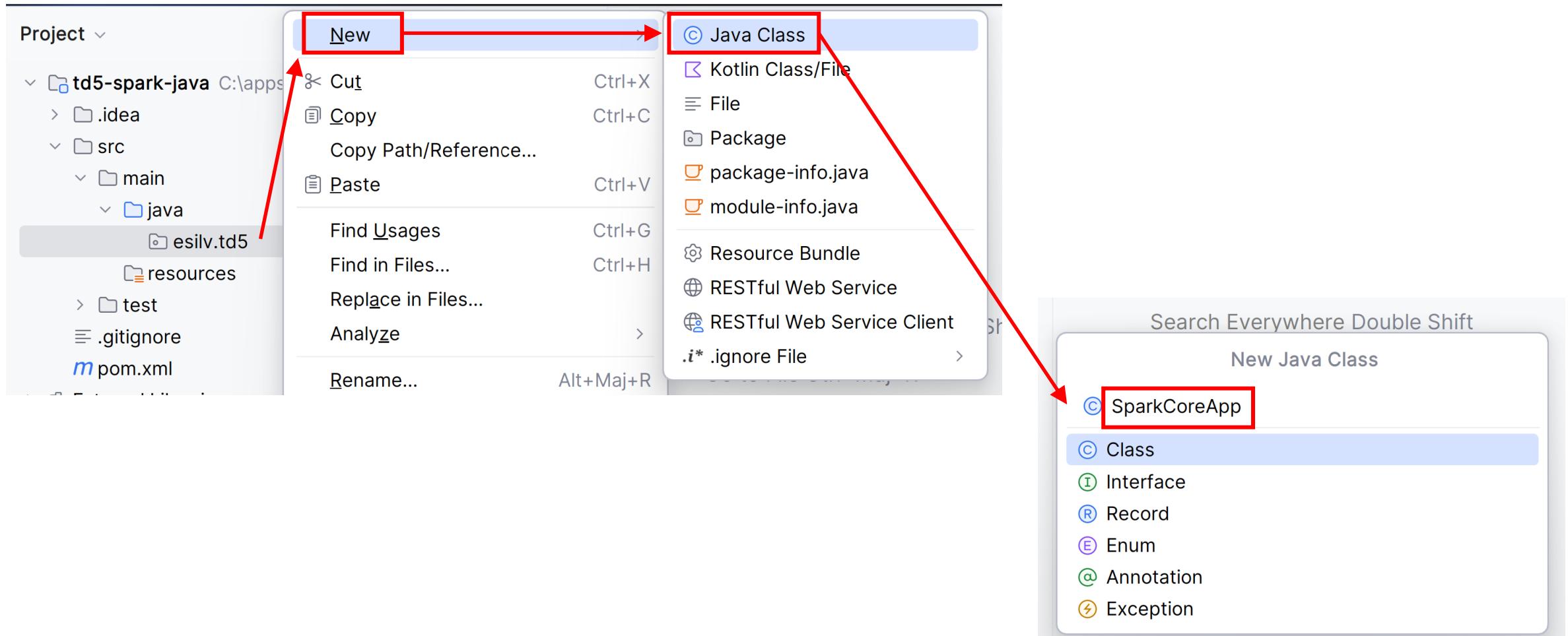
The "Sync" icon will disappear

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

The screenshot shows a Java project named "td5-spark-java" in an IDE. The project structure on the left includes a .idea folder, a src directory containing main and java subfolders, and files like .gitignore and pom.xml. The SparkCoreApp.java file is selected in the code editor on the right. The code editor shows the following:

```
public class SparkCoreApp { no usages
}
```

The code editor has three lines of code. Line 1 starts with "public class" followed by the class name "SparkCoreApp". Line 2 contains a closing brace "}" and line 3 is blank. A tooltip "no usages" is shown next to the closing brace. The IDE interface includes a top bar with tabs for "td5-spark-java" and "Current File".

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

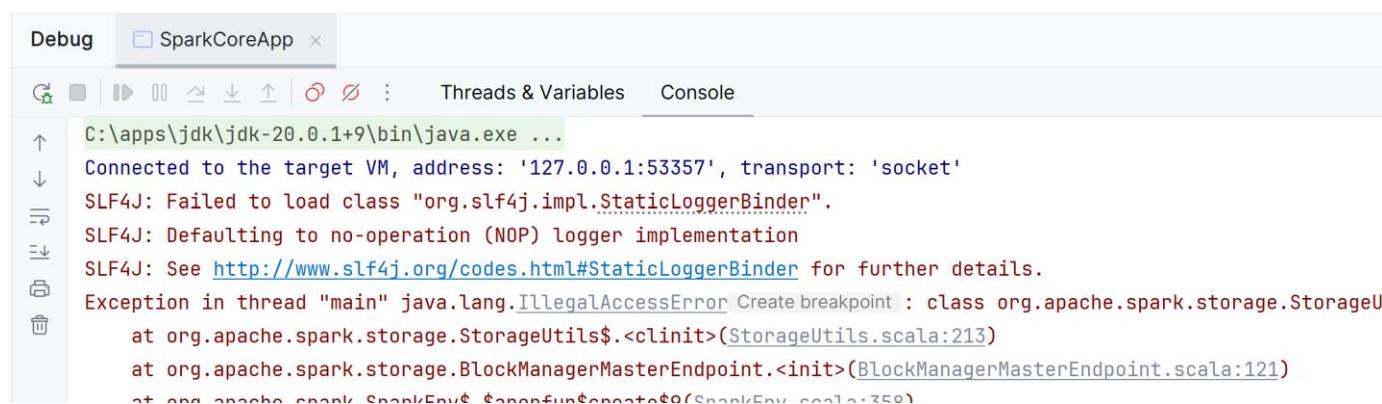
```
import org.apache.spark.SparkConf;
import org.apache.spark.SparkContext;
import org.apache.spark.rdd.RDD;

public class SparkCoreApp {

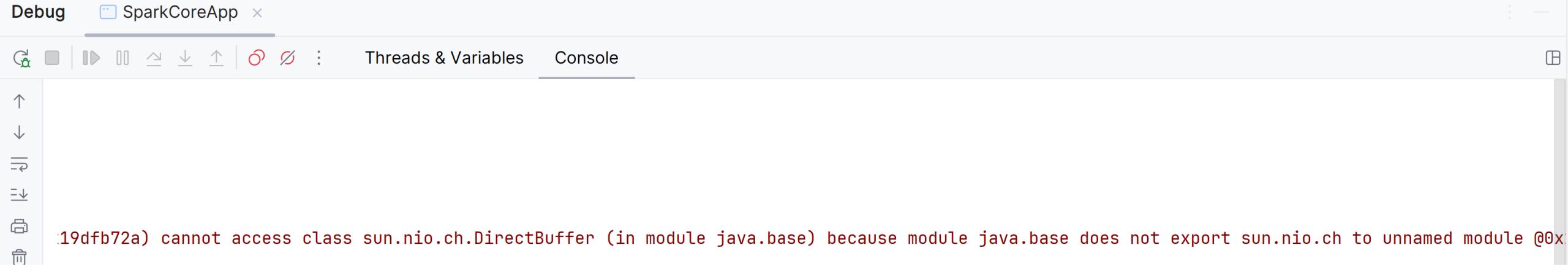
    public static void main(String[] args) {
        SparkConf sparkConf = new SparkConf()
            .setAppName("td5-sparkcore-app")
            .setMaster("local[*]");
        SparkContext sc = SparkContext.getOrCreate(sparkConf);

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

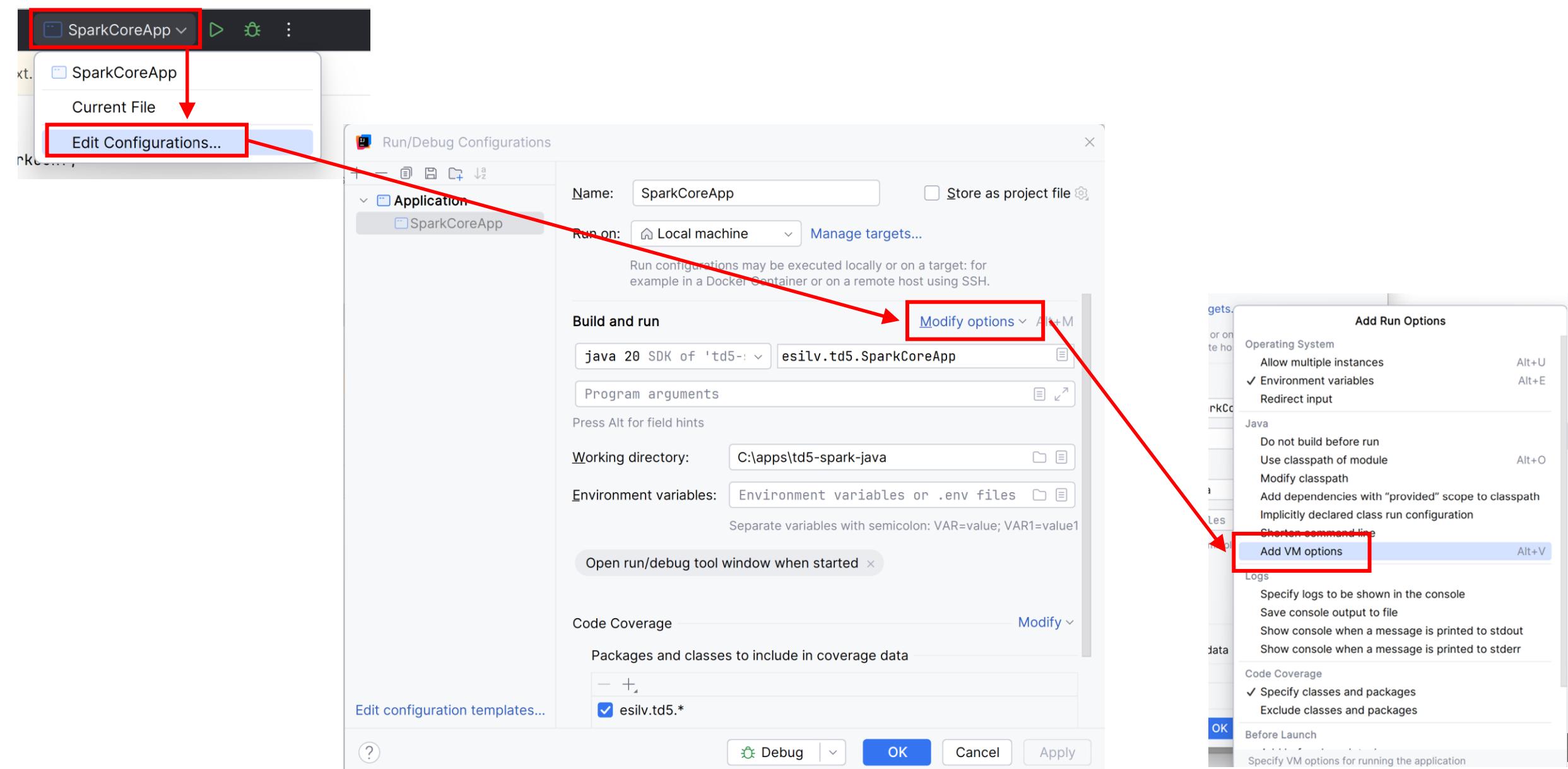
Debug (or Run) will FAIL on jdk >=11 and work on jdk <11



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



The screenshot shows a Java debugger interface with the title bar "Debug SparkCoreApp". The "Console" tab is selected, displaying the following log output:

```
C:\apps\JAK\JAK-20.0.1+Y\bin\java.exe ...
Connected to the target VM, address: '127.0.0.1:53427', transport: 'socket'
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
SLF4J: Failed to load class "org.slf4j.impl.StaticMDCBINDER".
SLF4J: Defaulting to no-operation MDCAdapter implementation.
SLF4J: See http://www.slf4j.org/codes.html#no\_static\_mdc\_binder for further details.
line count:4
Disconnected from the target VM, address: '127.0.0.1:53427', transport: 'socket'

Process finished with exit code 0
```

A red arrow points to the line "line count:4", which is highlighted with a red box. Another red box highlights the text "exit code 0" at the bottom of the log.

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 depends on spark-core_2.13
... you can remove spark-core, it is implicit dependency

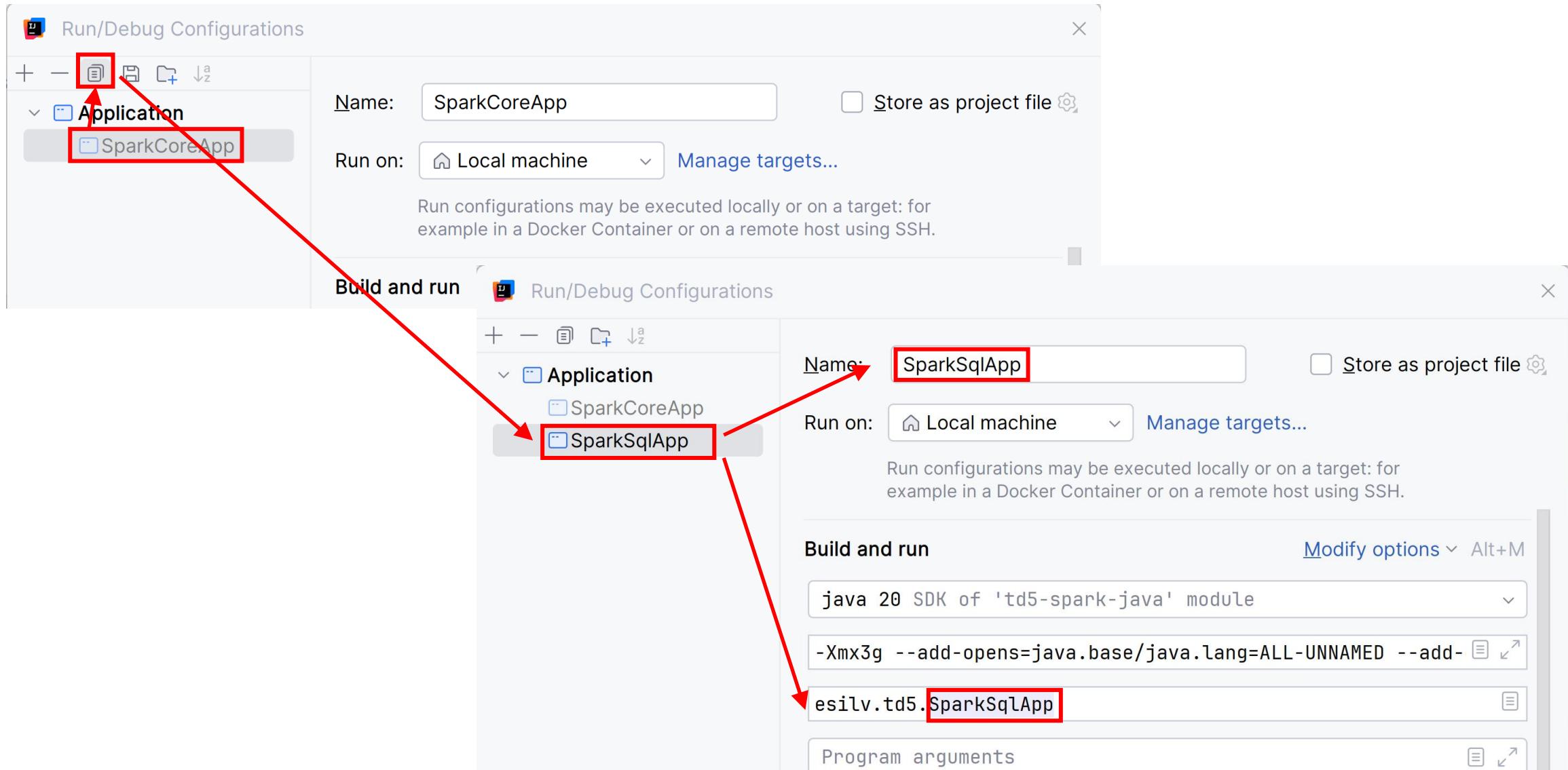
create new class "SparkSqlApp"

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

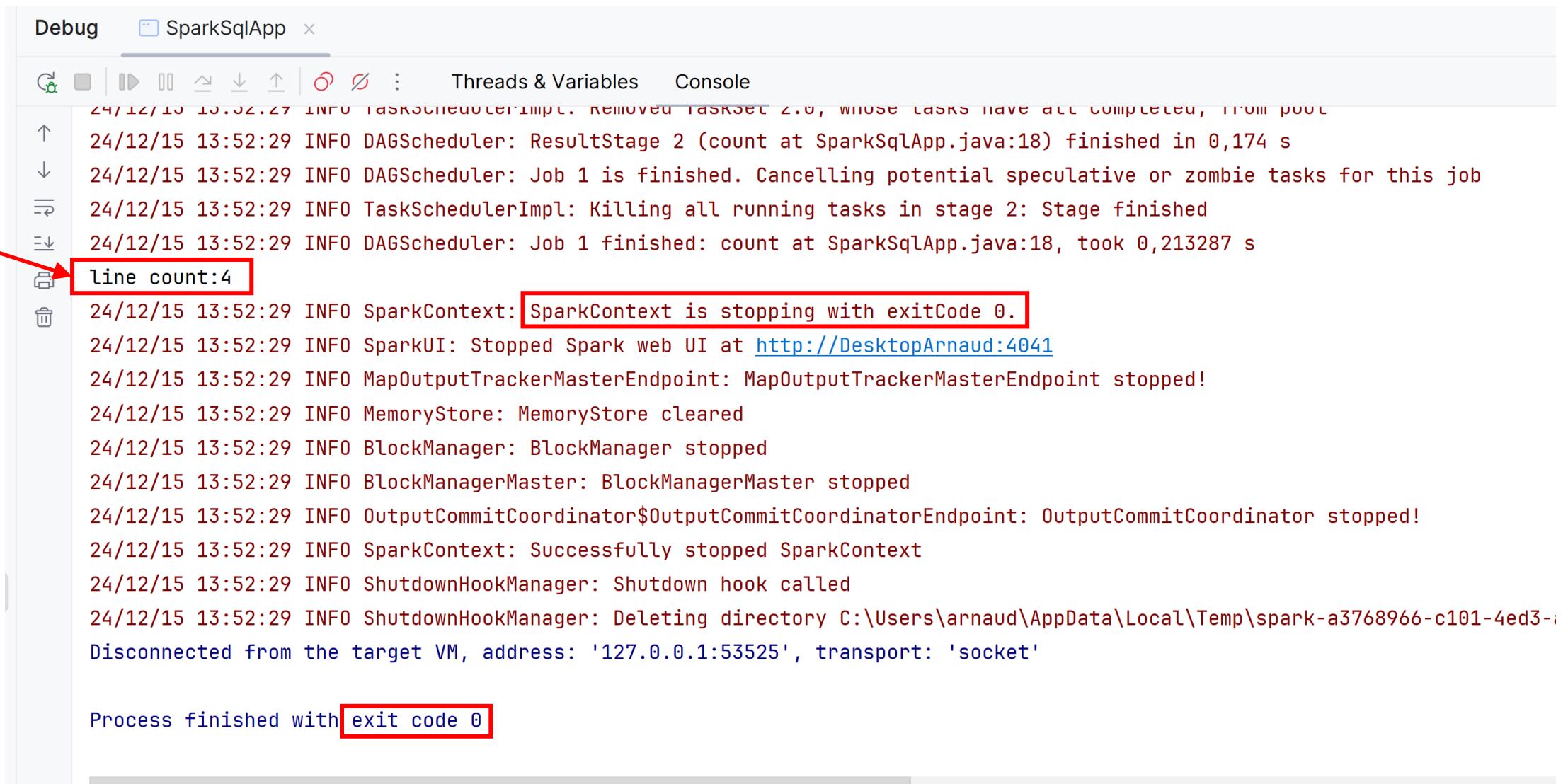
public class SparkSqlApp {
    public static void main(String[] args) {
        SparkConf sparkConf = new SparkConf()
            .setAppName("td5-sparksql-app")
            .setMaster("local[*]");
        SparkSession spark = SparkSession.builder()
            .config(sparkConf)
            .getOrCreate();

        Dataset<String> lineDs = spark.read().textFile("c:/data/lorem ipsum.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



The screenshot shows a Java IDE's debug console for a project named "SparkSqlApp". The console tab is active, displaying log messages from the application's execution. A red arrow points to the "line count:4" message, which is highlighted with a red box. Another red box highlights the "SparkContext is stopping with exitCode 0." message. The bottom of the console shows the process has finished with an exit code of 0.

```
24/12/15 13:52:27 INFO TaskSchedulerImpl: Removed TaskSet 2.0, which tasks have all completed, from pool
↑ 24/12/15 13:52:29 INFO DAGScheduler: ResultStage 2 (count at SparkSqlApp.java:18) finished in 0,174 s
↓ 24/12/15 13:52:29 INFO DAGScheduler: Job 1 is finished. Cancelling potential speculative or zombie tasks for this job
→ 24/12/15 13:52:29 INFO TaskSchedulerImpl: Killing all running tasks in stage 2: Stage finished
☰ 24/12/15 13:52:29 INFO DAGScheduler: Job 1 finished: count at SparkSqlApp.java:18, took 0,213287 s
🔗 line count:4
刪 24/12/15 13:52:29 INFO SparkContext: SparkContext is stopping with exitCode 0.
24/12/15 13:52:29 INFO SparkUI: Stopped Spark web UI at http://DesktopArnaud:4041
24/12/15 13:52:29 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
24/12/15 13:52:29 INFO MemoryStore: MemoryStore cleared
24/12/15 13:52:29 INFO BlockManager: BlockManager stopped
24/12/15 13:52:29 INFO BlockManagerMaster: BlockManagerMaster stopped
24/12/15 13:52:29 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
24/12/15 13:52:29 INFO SparkContext: Successfully stopped SparkContext
24/12/15 13:52:29 INFO ShutdownHookManager: Shutdown hook called
24/12/15 13:52:29 INFO ShutdownHookManager: Deleting directory C:\Users\arnaud\AppData\Local\Temp\spark-a3768966-c101-4ed3-
Disconnected from the target VM, address: '127.0.0.1:53525', transport: 'socket'

Process finished with exit code 0
```

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|  
+---+
```

Exercise : 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> --> 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")

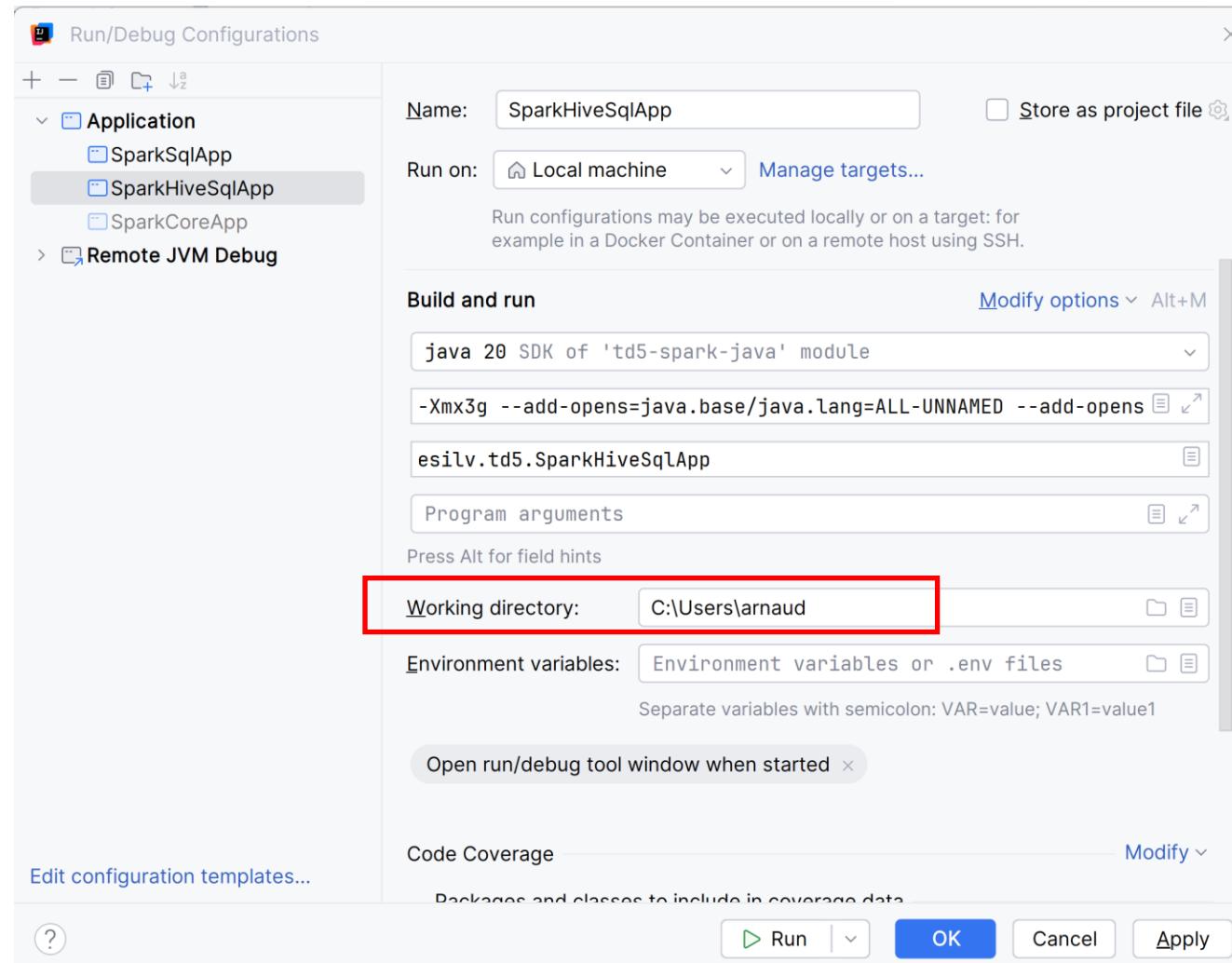
Edit pom.xml again, add dependency

```
<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 ??

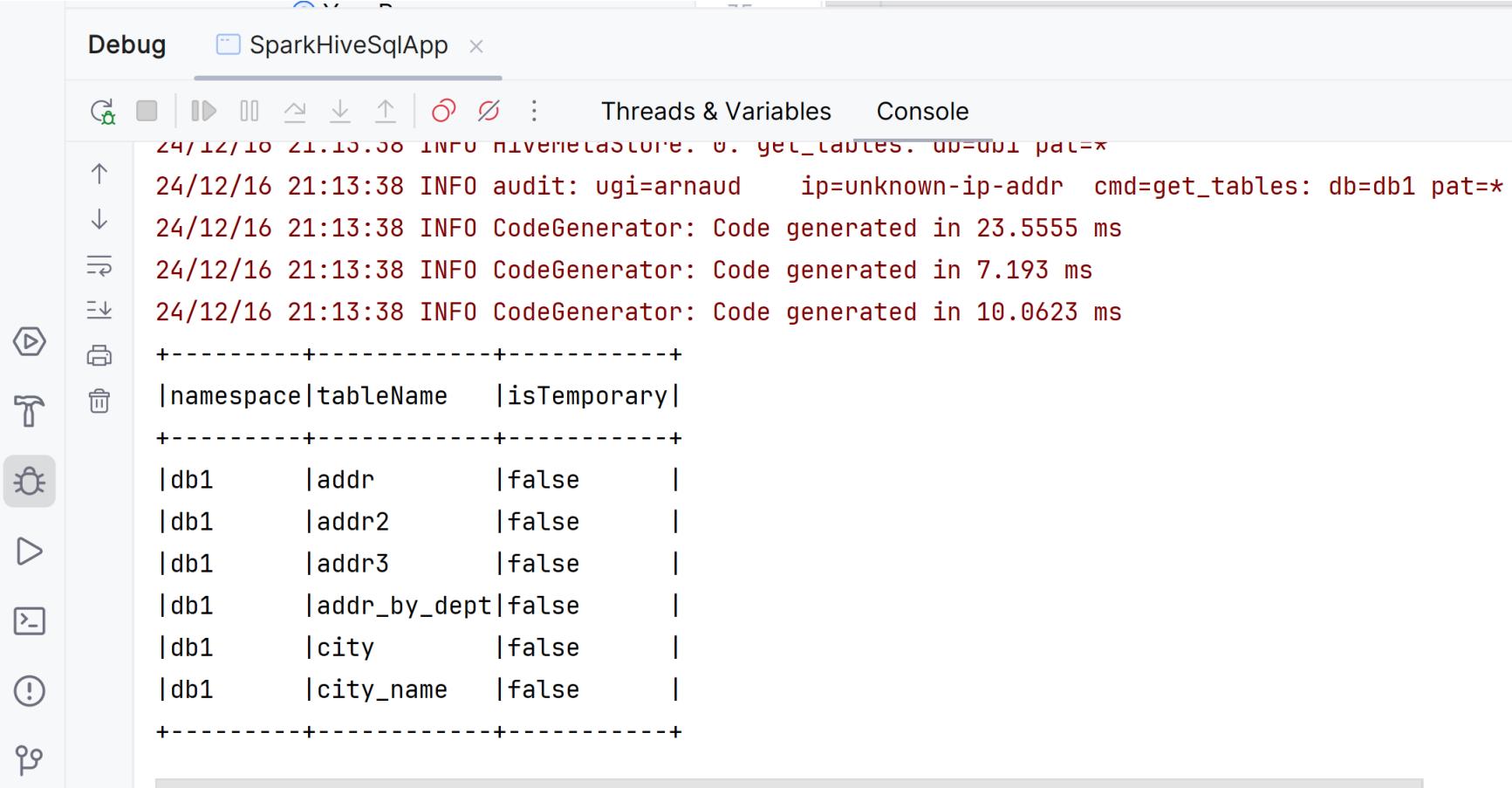
(=> need to stop all others running "spark-shell")

The screenshot shows a Java application named "SparkHiveSqlApp" running in an IDE. The main window displays a stack trace for a database connection error:

```
at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:191)
at com.jolbox.bonecp.BoneCP.obtainRawInternalConnection(BoneCP.java:361)
at com.jolbox.bonecp.BoneCP.<init>(BoneCP.java:416)
... 110 more
Caused by: ERROR XJ040: Failed to start database 'metastore_db' with class loader jdk.internal.loader.ClassLoaders$AppClassLoader
at org.apache.derby.iapi.error.StandardException.newException(Unknown Source)
at org.apache.derby.impl.jdbc.SQLExceptionFactory.wrapArgsForTransportAcrossDRDA(Unknown Source)
... 126 more
Caused by: ERROR XSDB6: Another instance of Derby may have already booted the database C:\Users\arnaud\metastore_db.
at org.apache.derby.iapi.error.StandardException.newException(Unknown Source)
at org.apache.derby.iapi.error.StandardException.newException(Unknown Source)
at org.apache.derby.impl.store.raw.data.BaseDataFileFactory.privGetJBMSLockOnDB(Unknown Source)
at org.apache.derby.impl.store.raw.data.BaseDataFileFactory.run(Unknown Source) <1 internal line>
```

The stack trace indicates that the application is attempting to connect to a database named "metastore_db" but is failing due to a lock held by another instance of Derby. The error messages point to issues with the ClassLoader and the Derby database boot process.

It works..



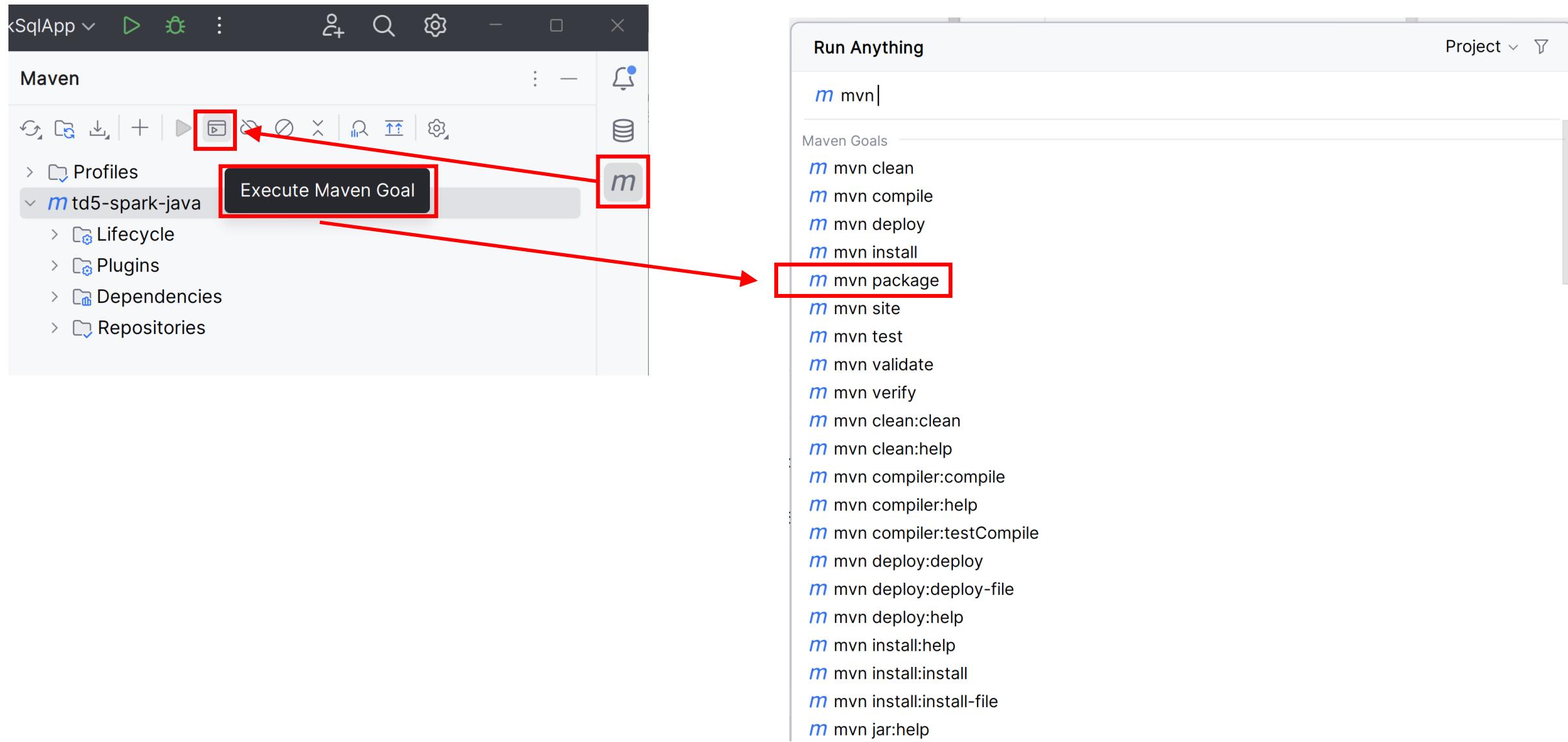
The screenshot shows the IntelliJ IDEA Debug tool window for a project named "SparkHiveSqlApp". The window has tabs for "Debug" and "SparkHiveSqlApp" (which is currently selected). Below the tabs are standard debugging icons: a green play button, a grey square, a red stop button, a double arrow, a left arrow, a right arrow, a magnifying glass, and a refresh symbol. To the right of these are buttons for "Threads & Variables" and "Console". The "Console" tab is active, displaying the following log output:

```
24/12/16 21:13:38 INFO audit: ugi=arnaud ip=unknown-ip-addr cmd=get_tables: db=db1 pat=*
24/12/16 21:13:38 INFO CodeGenerator: Code generated in 23.5555 ms
24/12/16 21:13:38 INFO CodeGenerator: Code generated in 7.193 ms
24/12/16 21:13:38 INFO CodeGenerator: Code generated in 10.0623 ms
+-----+-----+-----+
|namespace|tableName    |isTemporary|
+-----+-----+-----+
|db1      |addr        |false      |
|db1      |addr2       |false      |
|db1      |addr3       |false      |
|db1      |addr_by_dept|false      |
|db1      |city        |false      |
|db1      |city_name   |false      |
+-----+-----+-----+
```

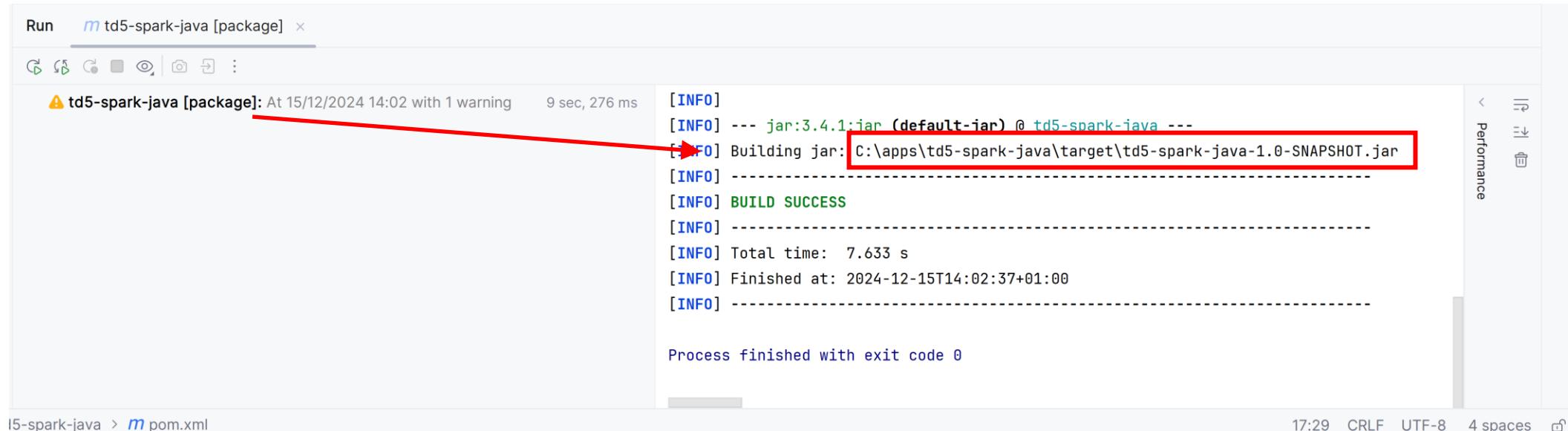
```
24/12/16 21:13:42 INFO TaskSchedulerImpl: Killing all running tasks in stage 2: Stage finished
24/12/16 21:13:42 INFO DAGScheduler: Job 1 finished: count at SparkHiveSqlApp.java:33, took 0,164527 s
addrDs count:26871372
Finished
```

Launching from Terminal
"spark-submit"

Build jar using "mvn package"



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



```
Run m td5-spark-java [package] x
td5-spark-java [package]: At 15/12/2024 14:02 with 1 warning 9 sec, 276 ms
[INFO]
[INFO] --- jar:3.4.1:jar (default-jar) @ td5-spark-java ---
[INFO] Building jar: C:\apps\td5-spark-java\target\td5-spark-java-1.0-SNAPSHOT.jar
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.633 s
[INFO] Finished at: 2024-12-15T14:02:37+01:00
[INFO] -----
Process finished with exit code 0
```

l5-spark-java > m pom.xml 17:29 CRLF UTF-8 4 spaces ↻

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-b82bcfedcd5fe
java.io.IOException: Failed to delete: C:\Users\arnaud\AppData\Local\Temp\spark-a17dec51-9765-4c05-94ba-f96b0bcc4d27\userFiles-a3eb5451-c7
0d-489c-b351-b82bcfedcd5fe\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)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(Native Method)
        at java.base/jdk.internal.reflect.Method.invoke(Method.java:576)
```

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

```
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.$anonfun$stop$25(SparkContext.scala:2216)  
        at org.apache.spark.SparkContext.$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 (!?)

issues.apache.org/jira/browse/SPARK-12216

APACHE SOFTWARE FOUNDATION http://www.apache.org/

Dashboards Projects Issues Search Log In

Public signup for this instance is **disabled**. Go to our [Self serve sign up page](#) to request an account. Report potential security issues privately

Apache Spark / SPARK-12216

Spark failed to delete temp directory

Details

Type:	Bug	Status:	RESOLVED
Priority:	Minor	Resolution:	Invalid
Affects Version/s:	None	Fix Version/s:	None
Component/s:	Spark Shell		
Labels:	None		
Environment:	windows 7 64 bit Spark 1.52 Java 1.8.0_65 PATH includes: C:\Users\Stefan\spark-1.5....		

Description

The mailing list archives have no obvious solution to this:

```
scala> :q
Stopping spark context.
15/12/08 16:24:22 ERROR ShutdownHookManager: Exception while deleting Spark temp dir:
C:\Users\Stefan\AppData\Local\Temp\spark-18f2a418-e02f-458b-8325-60642868fdf
java.io.IOException: Failed to delete: C:\Users\Stefan\AppData\Local\Temp\spark-18f2a418-e02f-458b-8325-
60642868fdf
at org.apache.spark.util.Utils$.deleteRecursively(Utils.scala:884)
*
```

People

Assignee: Unassigned
Reporter: stefan
Votes: 1 Vote for this issue
Watchers: 17 Start watching this issue

Dates

Created:	08/Dec/15 21:56
Updated:	26/Sep/22 13:42
Resolved:	13/Mar/16 12:38

Guram Savinov added a comment - 14/Mar/16 08:08

Why you closed this issue?
You don't care about Spark on Windows?

yaniv oren added a comment - 14/Mar/16 08:18

He gave a workaround, move to linux 😊
Solution perhaps should be in SparkShutdownHookManager.

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 ...

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project:** td5-spark-java
- File:** Executor.scala
- Code Snippet:** A portion of the Executor class is shown, specifically the updateDependencies method. It fetches a file from a URL and adds it to the class loader.
- Debugger:** A debugger session is active, showing the stack trace for the updateDependencies method. The current frame is \$anonfun\$updateDependencies\$14:1170, Executor (org.apache.spark.executor.Executor).
- Variables:** The local variables pane shows the state of variables like \$this, state, renewClassLoader, root, hadoopConf, x, name, timestamp, localName, currentTimeStamp, and url.
- Maven:** The Maven tool window shows the project structure and dependencies.

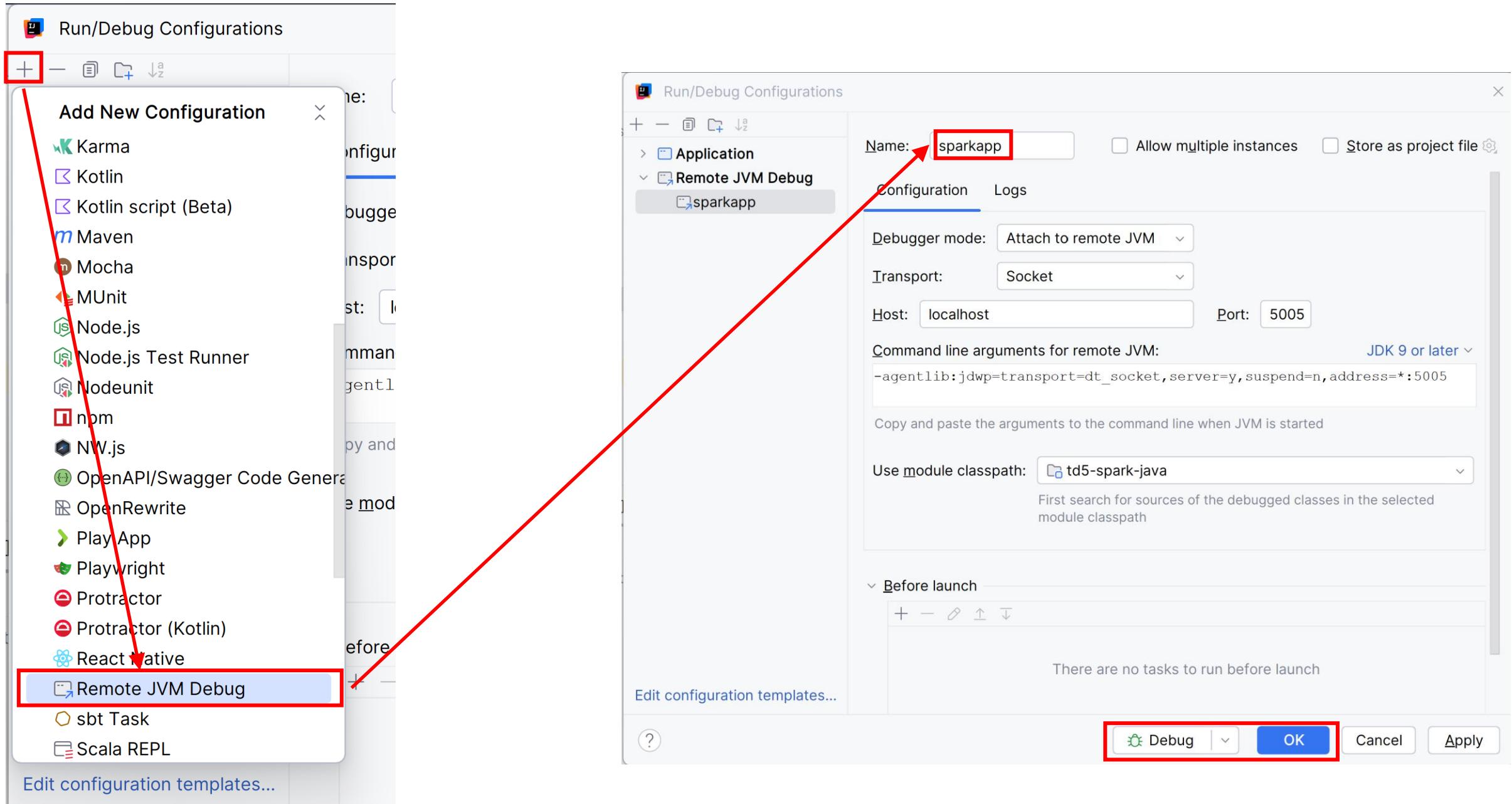
spark-submit with Remote JVM Debugging

```
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,suspend=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"

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "td5-spark-java".
- Code Editor:** Displays the `SparkSqlApp.java` file with the following code:

```
public class SparkSqlApp {
    public static void main(String[] args) {
        Dataset<String> lineDs = spark.read().textFile("c:/data/loremIpsum.txt");
        long lineCount = lineDs.count();
        System.out.println("line count:" + lineCount);
        spark.stop();
    }
}
```
- Debug View:** Shows the "sparkapp" configuration selected in the "Debug" tab.
- Java Main Thread:** The "Threads & Variables" tab is active, showing the current stack trace:

```
main:24, SparkSqlApp (esilv.td5), SparkSqlApp.java
invoke0:-1, NativeMethodAccessorImpl (jdk.internal.reflect), NativeMethodAccessorImpl.ja
invoke:75, NativeMethodAccessorImpl (jdk.internal.reflect), NativeMethodAccessorImpl.ja
invoke:52, DelegatingMethodAccessorImpl (jdk.internal.reflect), DelegatingMethodAccess
invoke:578, Method (java.lang.reflect), Method.java
start:52, JavaMainApplication (org.apache.spark.deploy), SparkApplication.scala
org$apache$spark$deploy$SparkSubmit$$runMain:1029, SparkSubmit (org.apache.spark.de
doRunMain$1:194, SparkSubmit (org.apache.spark.deploy), SparkSubmit.scala
submit:217, SparkSubmit (org.apache.spark.deploy), SparkSubmit.scala
doSubmit:91, SparkSubmit (org.apache.spark.deploy), SparkSubmit.scala
doSubmit:1120, SparkSubmit$$anon$2 (org.apache.spark.deploy), SparkSubmit.scala
main:1129, SparkSubmit$ (org.apache.spark.deploy), SparkSubmit.scala
main:-1, SparkSubmit (org.apache.spark.deploy), SparkSubmit.scala
```
- Variables:** A tooltip for the variable `lineCount` shows its value as `4`.
- Maven:** The Maven tool window is open, showing the project configuration.
- Bottom Status Bar:** Shows the path `d5-spark-ava > src > main > iava > esilv > td5 > SparkSalApp`, and the status `24:1 CRI F UTF-8 4 spaces`.

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

← → ⌂ en.wikipedia.org/wiki/Edit_distance

Contents hide

(Top)

Types of edit distance

Formal definition and properties

Example

Properties

Computation

Common algorithm

Improved algorithms

Applications

Language edit distance

See also

References

Computation [edit]

The first algorithm for computing minimum edit distance between a pair of strings was published by Damerau in 1964.^[6]

Common algorithm [edit]

Main article: *Wagner–Fischer algorithm*

Using Levenshtein's original operations, the (nonsymmetric) edit distance from $a = a_1 \dots a_m$ to $b = b_1 \dots b_n$ is given by d_{mn} , defined by the recurrence^[2]

$$\begin{aligned} d_{i0} &= \sum_{k=1}^i w_{\text{del}}(a_k), && \text{for } 1 \leq i \leq m \\ d_{0j} &= \sum_{k=1}^j w_{\text{ins}}(b_k), && \text{for } 1 \leq j \leq n \\ d_{ij} &= \begin{cases} d_{i-1,j-1} & \text{for } a_i = b_j \\ \min \begin{cases} d_{i-1,j} + w_{\text{del}}(a_i) \\ d_{i,j-1} + w_{\text{ins}}(b_j) \\ d_{i-1,j-1} + w_{\text{sub}}(a_i, b_j) \end{cases} & \text{for } a_i \neq b_j \end{cases} && \text{for } 1 \leq i \leq m, 1 \leq j \end{aligned}$$

This algorithm can be generalized to handle transpositions by adding another term in the recursive clause's minimization.^[3]

The straightforward, *recursive* way of evaluating this recurrence takes *exponential time*.

Therefore, it is usually computed using a *dynamic programming* algorithm that is commonly credited to *Wagner and Fischer*,^[7] although it has a history of multiple invention.^{[2][3]} After completion of the Wagner–Fischer algorithm, a minimal sequence of edit operations can be read off as a backtrace of the operations used during the dynamic programming algorithm starting at d_{mn} .

Levenshtein Distance

← → ⌂ en.wikipedia.org/wiki/Levenshtein_distance



WIKIPEDIA
The Free Encyclopedia

Search Wikipedia

Search

Do

Levenshtein distance

文 36 languages ▾

Article Talk

Read Edit View history Tools ▾

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

e	v	e	n	s	h	t	f	e	r	n	
e	0.5	1	0.5	1.5	2	2.5	3	3.5	4	4.5	5
v	1	1.5	1	0.5	1.5	2	2.5	3	3.5	4	4.5
e	1.5	2	1.5	1	0.5	1	1.5	2	2.5	3	3.5
n	2	2.5	2	1.5	1	0.5	1	1.5	2	2.5	3
s	2.5	3	2.5	2	1.5	1	0.5	1	1.5	2	2.5
h	3	3.5	3	2.5	2	1.5	1	0.5	1	1.5	2
t	3.5	4	3.5	3	2.5	2	1.5	1	0.5	1	1.5
f	4	4.5	4	3.5	3	2.5	2	1.5	1	0.5	1
e	4.5	5	4.5	4	3.5	3	2.5	2	1.5	1	0.5
n	5	5.5	5	4.5	4	3.5	3	2.5	2	1.5	1

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](#).

Contents hide

(Top)

Definition

Example

Upper and lower bounds

Applications

Relationship with other edit distance metrics

Computation

Recursive

Iterative with full matrix

Iterative with two matrix rows

Automata

Approximation

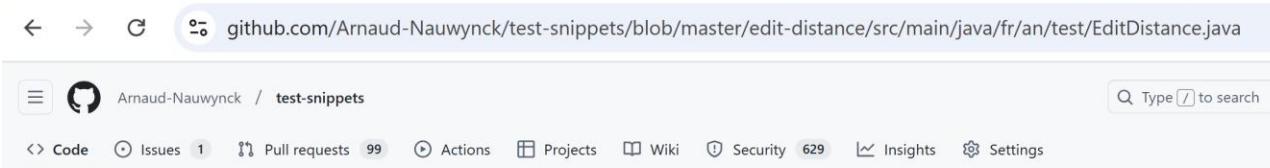
Computational complexity

See also

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>



The screenshot shows a GitHub repository page for 'test-snippets'. The repository has 1 issue, 99 pull requests, 629 security vulnerabilities, and 165 lines of code. The 'Code' tab is selected, displaying the 'EditDistance.java' file. The file contains Java code for calculating edit distance using dynamic programming.

```
1 package fr.an.test;
2
3 import java.util.Arrays;
4
5 public class EditDistance {
6     public static int min(int x, int y, int z) {
7         return Math.min(Math.min(x, y), z);
8     }
9
10    public static int editDistance(String str1, String str2) {
11        int m = str1.length();
12        int n = str2.length();
13
14        int[][] dp = new int[m + 1][n + 1];
15
16        for (int i = 0; i <= m; i++) {
17            for (int j = 0; j <= n; j++) {
18                if (i == 0) {
19                    dp[i][j] = j;
20                } else if (j == 0) {
21                    dp[i][j] = i;
22                } else if (str1.charAt(i - 1) == str2.charAt(j - 1)) {
23                    dp[i][j] = dp[i - 1][j - 1];
24                } else {
25                    dp[i][j] = 1 + min(dp[i][j - 1], // Insert
26                                         dp[i - 1][j], // Remove
27                                         dp[i - 1][j - 1]); // Replace
28                }
29            }
30        }
31
32        return dp[m][n];
33    }
}
```

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", "prises", 10)
// 4
```

```
fr.an.test.EditDistance.editDistanceLimitedTo("paris", "prises", 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', 'prises', 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)

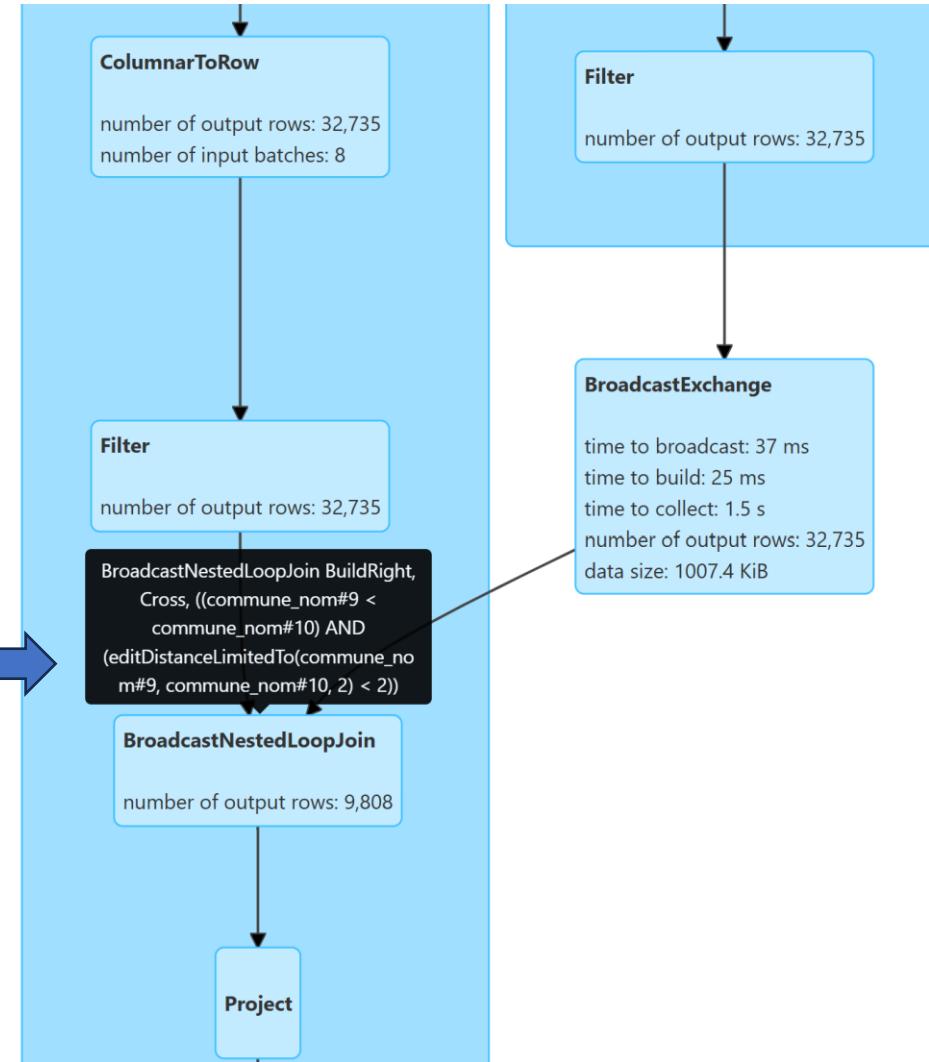
Spill to Disk ...

use --driver-memory=8g

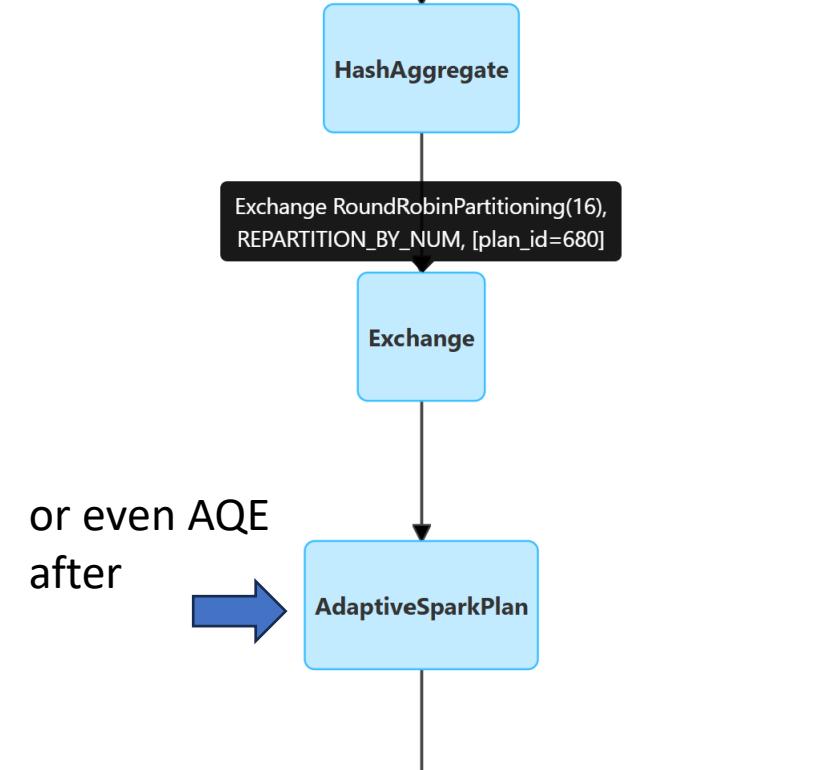
```
scala> crossDist.show(10, false)
24/12/09 01:28:48 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:28:48 WARN RowBasedKeyValueBatch: Calling spill() on RowBasedKeyValueBatch. Will not spill but return 0.
24/12/09 01:29:30 WARN RowBasedKeyValueBatch: 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 RowBasedKevValueBatch: Callina spill() on RowBasedKevValueBatch. Will not spill but return 0.
```

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

push down
in broadcast join !



localhost:4040/SQL/execution/?id=12



or even AQE
after

Cross City Pairs with Distance=1

```
scala> crossDist.show(10, false)
+-----+-----+---+
|c1_nom          |c2_nom          |d |
+-----+-----+---+
|Auge            |Augy             |1 |
|Many             |Mary              |1 |
|Lunan            |Lunay             |1 |
|Caunay           |Launay            |1 |
|Balloons         |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 Arrondissement	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>

The screenshot shows a documentation page for Spark SQL. At the top, there's a navigation bar with links for Overview, Programming Guides, API Docs, Deploying, More, and a search bar. Below the navigation bar, the main content area has a title "Type-Safe User-Defined Aggregate Functions". A sub-section header "User-defined aggregations for strongly typed Datasets revolve around the `Aggregator` abstract class. For example, a type-safe user-defined average can look like:" is followed by code examples for Scala and Java.

Scala

```
import java.io.Serializable;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Encoder;
import org.apache.spark.sql.Encoders;
import org.apache.spark.sql.SparkSession;
import org.apache.spark.sql.TypedColumn;
import org.apache.spark.sql.expressions.Aggregator;

public static class Employee implements Serializable {
    private String name;
    private long salary;

    // Constructors, getters, setters...
}

public static class Average implements Serializable {
    private long sum;
    private long count;

    // Constructors, getters, setters...
}

public static class MyAverage extends Aggregator<Employee, Average, Double> {
    // A zero value for this aggregation. Should satisfy the property that any b + zero = b
    @Override
    public Average zero() {
        return new Average(0L, 0L);
    }
    // Combine two values to produce a new value. For performance, the function may modify `buffer`
}
```

Java

Aggregator<IN,BUFFER,OUT>

IN => named_struct "Point"

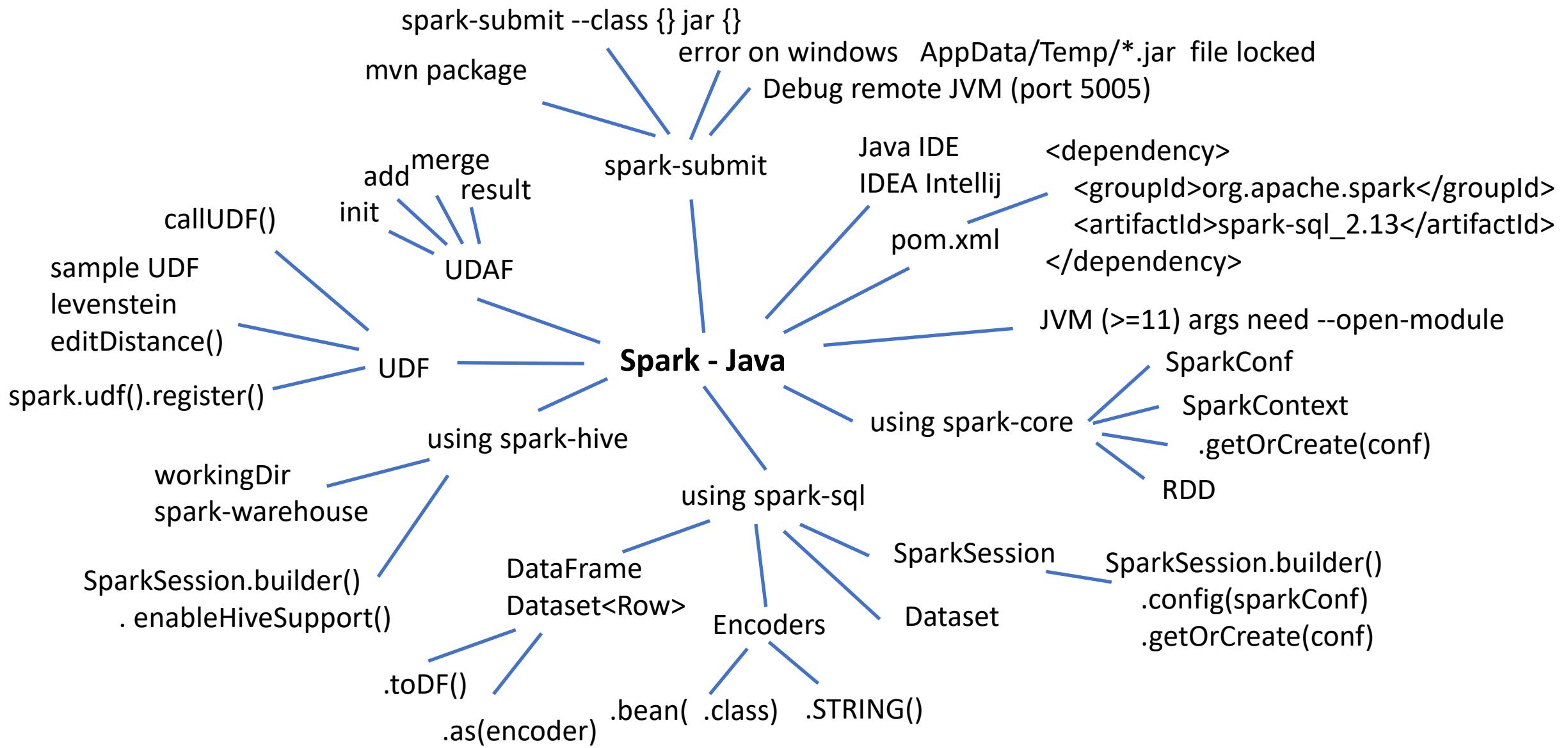
BUFFER => internal sums + number of point

OUT => named_struct for matrix

exercise left to students ...

Exercise :
draw a MindMap of this Hands-on session

MindMap



Questions ?