Title: Jdeps Module java.annotation not found

Post Body:

I'm trying to create a minimal jre for Spring Boot microservices using jdeps and jlink, but I'm getting the following error when I get to the using jdeps part

```
Exception in thread "main" java.lang.module.FindException: Module java.annotation not found, required by org.apache.tomcat.emb
```

I already tried the following commands with no effect

```
jdeps --ignore-missing-deps --multi-release 17 --module-path target/lib/* target/errorrr-*.jar jdeps --multi-release 16 --modu
```

I already tried it with java versions 11, 16 and 17 and different versions of Spring Boot.

All dependencies needed for build are copied to target/lib folder by maven-dependency-plugin plugin when I run mvn install

After identifying the responsible dependency I created a new project from scratch with only it to isolate the error, but it remained.

I tried to use gradle at first but as the error remained I changed it to mavem but also no change.

When I add the specified dependency that is being requested the error changes to

```
#13 1.753 Exception in thread "main" java.lang.Error: java.util.concurrent.ExecutionException: com.sun.tools.jdeps.MultiReleas
```

My pom.xml

```
   <?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001
```

If I don't need to use this dependency I can do all the build processes and at the end I have a 76mb jre

Accepted Answer:

I arrived at a solution that is valid, not perfect, but it works correctly. In my case the lib "jackson" was the cause of the problem. It uses java multi release and somehow during jdeps it was causing some error. After some tests I understood that I could remove this lib from the evaluation, and that no module would be missing from deps.info. I needed to add this task for removal:

```
task myDeleteTask(type: Delete) {     delete files("${buildDir}/temp-lib/jackson-databind-{your_version}.jar")     delete file
```

OBS: in the question I used maven because it is easier and I have a wider knowledge, but my real project uses gradle, which ends up making the delete task easier to do

Highest Rated Answer:

I was facing a similar issue, what helped in my case - is specifying both `--class-path` and `--module-path` pointing to the same directory with libs. According to your example, I think it should be `jdeps --ignore-missing-deps --print-module-deps --multi-release 17 --module-path="target/lib/*" --class-path="target/lib/*" target/errorrr-*.jar`.

Also, `jdeps` from JDK 17 (and maybe earlier versions) seems to have a bug where it can throw `com.sun.tools.jdeps.MultiReleaseException`. It seems to have been fixed in JDK 18, at least it works without any issues for me.

With Docker you can do a staged build that will identify the dependencies first using JDK 18, and then build a new JRE image out of JDK 17. Like this:

```
FROM amazoncorretto:18-alpine as deps  COPY ./app.jar /app/app.jar RUN mkdir /app/unpacked && \     cd /app/unpacked && \
```

You can check the full example here: https://github.com/monosoul/jvm-in-docker/blob/main/jre-slim-auto.dockerfile

For anyone interested, here's a blog article about using `jlink` and `jdeps`: https://blog.monosoul.dev/2022/04/25/reduce-java-docker-image-size/