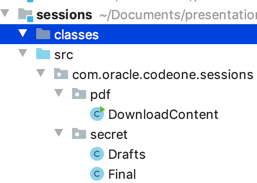
# Hands on Java 11 OCP Certification Prep

# Lab 1 - Modules

Note: It is ok to use an IDE to create the directories/files. We do recommend you use the command line to actually compile/run the programs in the module lab so you can see what is happening behind the scenes of your IDE.

## Step 1 – Create a regular Java project without specifying a module-info file

1. Create this directory structure. You can place the **sessions** folder anyplace on your hard drive that you like. Note:
   1. The classes directory is empty
   2. The three classes are listed immediately below the screenshot



*// Drafts.java*

*package com.oracle.codeone.sessions.secret;*

*import java.util.List;*

*public class Drafts {*

*public static List<String> getDrafts() {*

*return List.of("Draft HOL", "Draft IDEs");*

*}*

*}*

*// Final.java*

*package com.oracle.codeone.sessions.secret;*

*import java.util.List;*

*public class Final {*

*public static List<String> getDecks() {*

*return List.of("Final HOL", "Final IDEs");*

*}*

*}*

*// DownloadContent.java*

*package com.oracle.codeone.sessions.pdf;*

*import com.oracle.codeone.sessions.secret.Drafts;*

*import com.oracle.codeone.sessions.secret.Final;*

*public class DownloadContent {*

*public static void main(String[] args) {*

*System.out.println("Final:");*

*System.out.println(Final.getDecks());*

*System.out.println("Drafts:");*

*System.out.println(Drafts.getDrafts());*

*}*

*}*

1. Create an empty **modules** folder at the same level as the **sessions** folder
2. Compile:

javac -d sessions/classes sessions/src/com/oracle/codeone/sessions/pdf/\*.java sessions/src/com/oracle/codeone/sessions/secret/\*.java

1. Create a jar file:

jar -cvf modules/com.oracle.codeone.sessions.jar -C sessions/classes/ .

Note: There is a space between the final slash and the period.

1. Run the java program:

java -p modules -m com.oracle.codeone.sessions/com.oracle.codeone.sessions.pdf.DownloadContent

1. Confirm the output is:

*Final:*

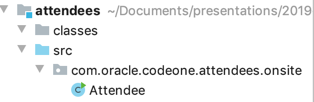
*[Final HOL, Final IDEs]*

*Drafts:*

*[Draft HOL, Draft IDEs]*

## Step 2 – Try to create a second module and observe the compiler error

1. Create an **attendees** folder at the same level as the sessions folder with the following contents. The **Attendee** class follows. (Note that it will not compile)



*// Attendee.java*

*package com.oracle.codeone.attendees.onsite;*

*import com.oracle.codeone.sessions.pdf.DownloadContent;*

*public class Attendee {*

*public static void main(String[] args) {*

*DownloadContent.main(args);*

*}*

*}*

1. If you are working an IDE, you should get a compiler error when trying to create the **Attendee** class because the attendees module doesn’t have a dependency on the sessions module. If you aren’t using an IDE, compile and observe you get an error about the package com.oracle.codeone.sessions.pdf not existing:

javac -p modules -d attendees/classes attendees/src/com/oracle/codeone/attendees/onsite/\*.java

error: package com.oracle.codeone.sessions does not exist

import com.oracle.codeone.sessions.pdf;

^

attendees/src/com/oracle/codeone/attendees/onsite/Attendee.java:9: error: cannot find symbol

System.out.println(Final.getDecks());

^

symbol: variable Final

location: class Attendee

2 errors

## Step 3: Add a module-info to the sessions module

1. Directly under **sessions/src**, create a **module-info.java** file with the contents:

*module com.oracle.codeone.sessions {*

*exports com.oracle.codeone.sessions.pdf;*

*}*

1. Recompile the module

javac -d sessions/classes sessions/src/module-info.java sessions/src/com/oracle/codeone/sessions/pdf/\*.java sessions/src/com/oracle/codeone/sessions/secret/\*.java

1. Re-create the jar

jar -cvf modules/com.oracle.codeone.sessions.jar -C sessions/classes/ .

1. Note: At this point, the com.oracle.codeone.sessions module has “agreed” to make the PDFs package available to other modules. However, the com.oracle.codeone.attendees package hasn’t “declared” that it uses it yet.
2. Directly under **attendees/src**, create a **module-info.java** file with the contents:

*module com.oracle.codeone.attendees {*

*requires com.oracle.codeone.sessions;*

*}*

1. Compile the second module successfully this time

javac -p modules -d attendees/classes attendees/src/module-info.java attendees/src/com/oracle/codeone/attendees/onsite/\*.java

1. Create a jar of this second module

jar -cvf modules/com.oracle.codeone.attendees.jar -C attendees/classes/ .

1. Run

java -p modules -m com.oracle.codeone.attendees/com.oracle.codeone.attendees.onsite.Attendee

1. Confirm the output is:

*Final:*

*[Final HOL, Final IDEs]*

*Drafts:*

*[Draft HOL, Draft IDEs]*

## Bonus Step (4) – Exporting to a specific module

Since this one is a bonus, you get to figure more out instead of us listing each step. Your goal is to export the com.oracle.codeone.sessions.pdf to just the com.oracle.codeone.attendees module.

Hint: the syntax is exports *packageName* to *moduleName*

Once you’ve updated the module-info.java, change the **Attendee** class’ main method to

*public static void main(String[] args) {*

*System.out.println("Final:");*

*System.out.println(Final.getDecks());*

*}*

## Bonus Step (5) – Working with requires transitive

Create a third module named com.oracle.codeone.beer. Have the **module-info.java** file **require** com.oracle.codeone.attendees.

Create one class named **GetBeerReadLater** with a main method:

*public static void main(String[] args) {*

*DownloadContent.main(args);*

*}*

Hint: You need to change the **module-info.java** file in com.oracle.codeone.attendees to get this to compile.