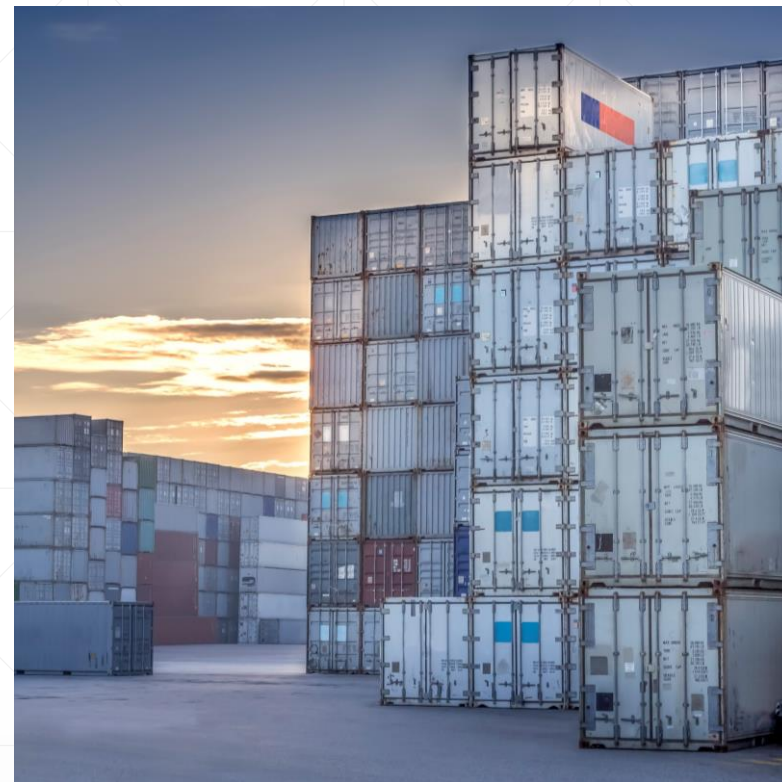


模块化后的JDK

北京理工大学计算机学院
金旭亮



JDK进行模块化的必要性



在 JDK 9之前，Java的运行时库由一个庞大的`rt.jar`所组成，其大小超过60MB，包含了Java大部分运行时类——它们是Java平台的最终载体。

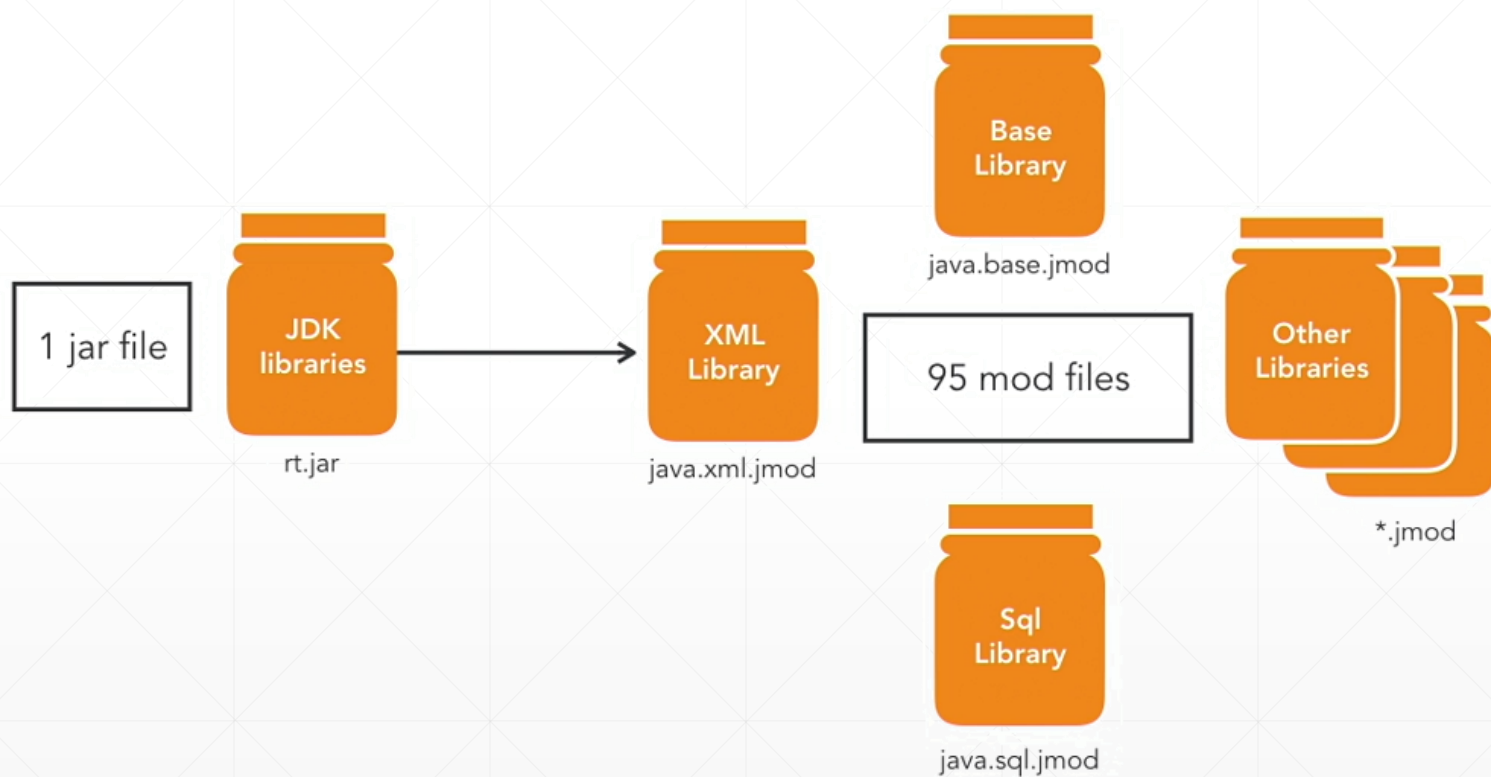


在过去20年里，JDK中增加了许多API，但几乎没有删除任何API，有些类，比如CORBA相关的类，现在已经很少用了，但仍然保留在JRE中，保留它们的唯一目的就是为了保证兼容性。



另一个例子，对于Java Web应用程序，JavaFX就是基本上用不到的，很明显，将JavaFX相关的jar包一并部署到目标机器上是没必要的。

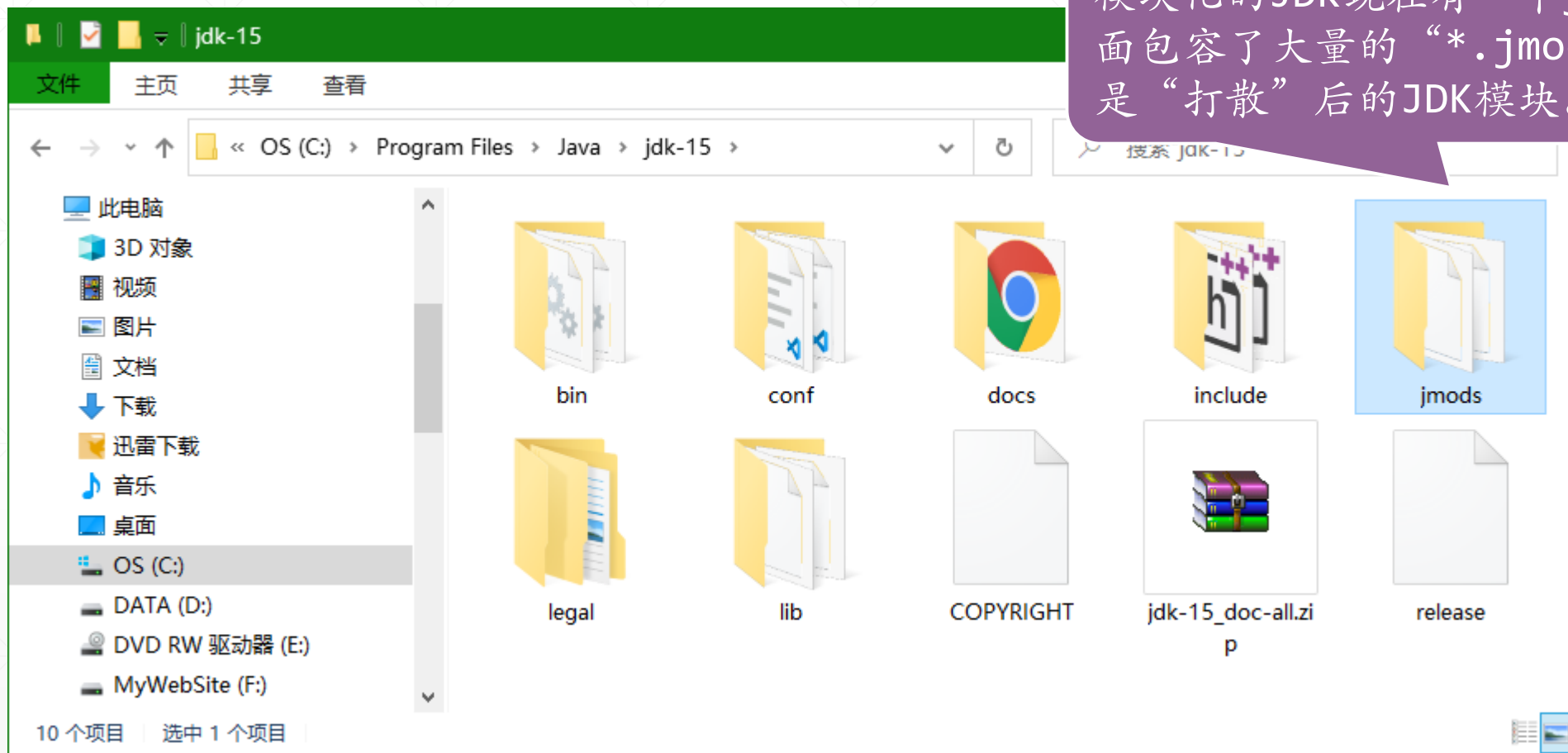
化整为零的JDK 9



从JDK 9开始，`rt.jar`模块被打散为多个模块。

打散后的JDK模块，以`.jmod`作为文件扩展名。

模块化的JDK



模块化的JDK现在有一个jmods文件夹，里面包容了大量的“*.jmod”文件，这其实就是“打散”后的JDK模块。

模块化后的JDK



JDK由许多个单独的平台模块组成，而不再是一个整体库了。



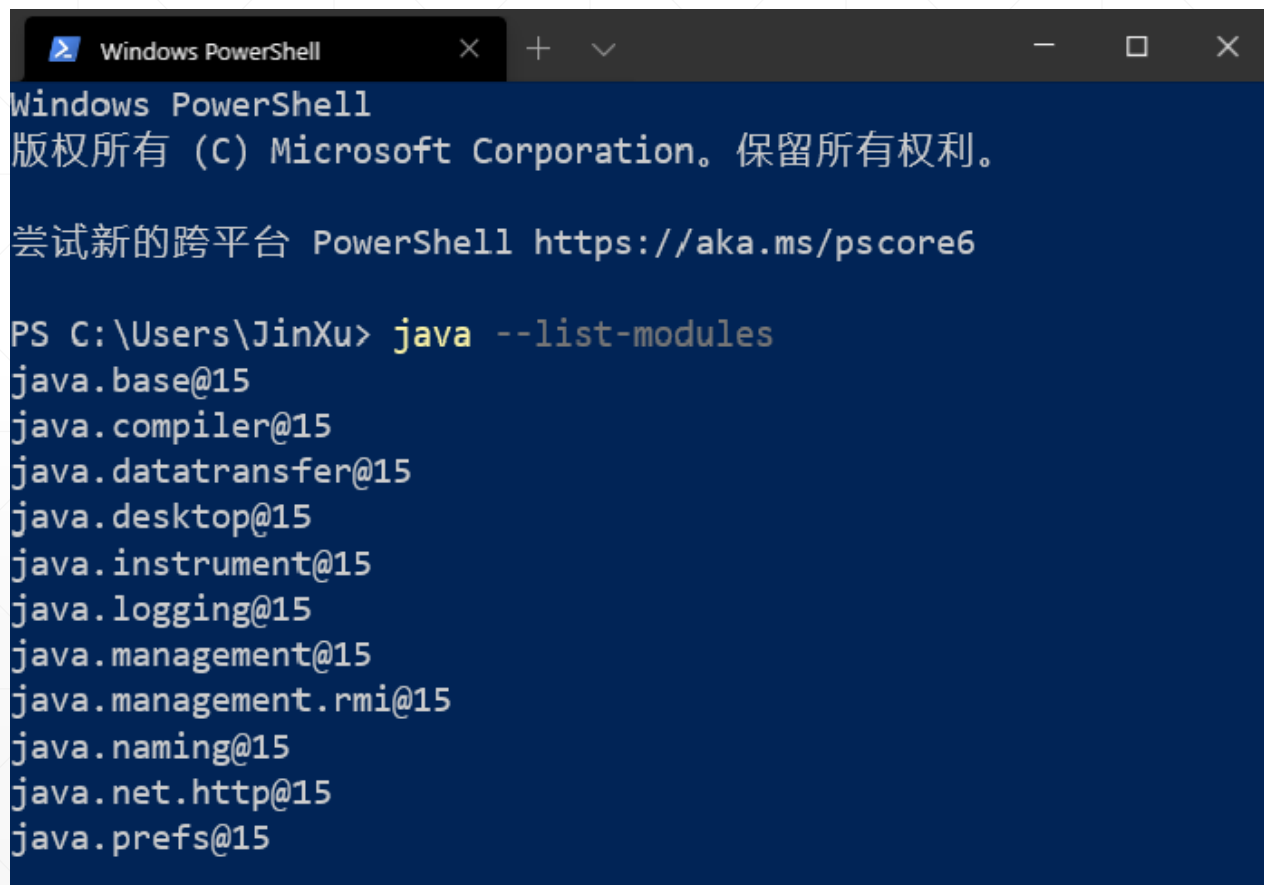
模块之间存在着“单向”依赖关系，Java模块系统不允许模块之间存在编译时的循环依赖。



每个模块都隐式依赖于一个名为“`java.base`”的特殊模块，它是一种“**聚合器模块（aggregator module）**”，这种类型的模块主要用于对其他模块进行逻辑分组，避免在`module-info.java`中导入太多的模块声明。

列出JDK（9以上版本） 的模块清单

```
java --list-modules
```



```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

尝试新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\JinXu> java --list-modules
java.base@15
java.compiler@15
java.datatransfer@15
java.desktop@15
java.instrument@15
java.logging@15
java.management@15
java.management.rmi@15
java.naming@15
java.net.http@15
java.prefs@15
```

以“java.”打头的模块，都是Java SE规范的一部分。

以“jdk.”开头的模块包含了JDK特定的代码，在不同的JDK实现中可能会有所不同。

在JDK 9以上的版本中，java和javac等原有JDK命令行工具都针对模块进行了功能增强，添加了一些新的命令行参数，JDK中还添加了诸如jlink之类的新命令行工具。

jmod与jar在使用上有何区别？

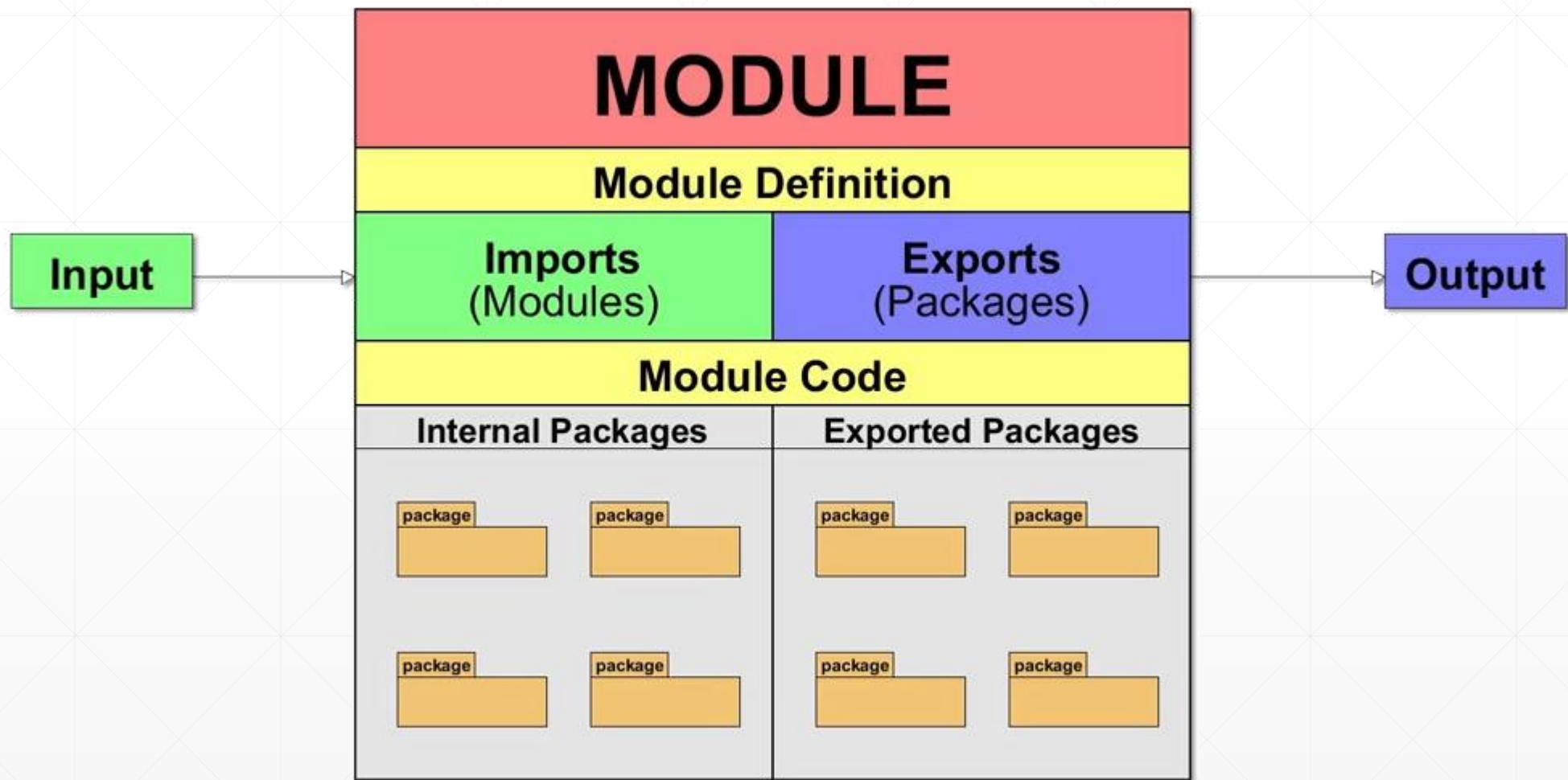


为了保证兼容性，高版本的JDK，始终支持直接运行jar包中的代码，然后，在相应的命令行工具，比如javac和java中，添加对于模块化的支持。

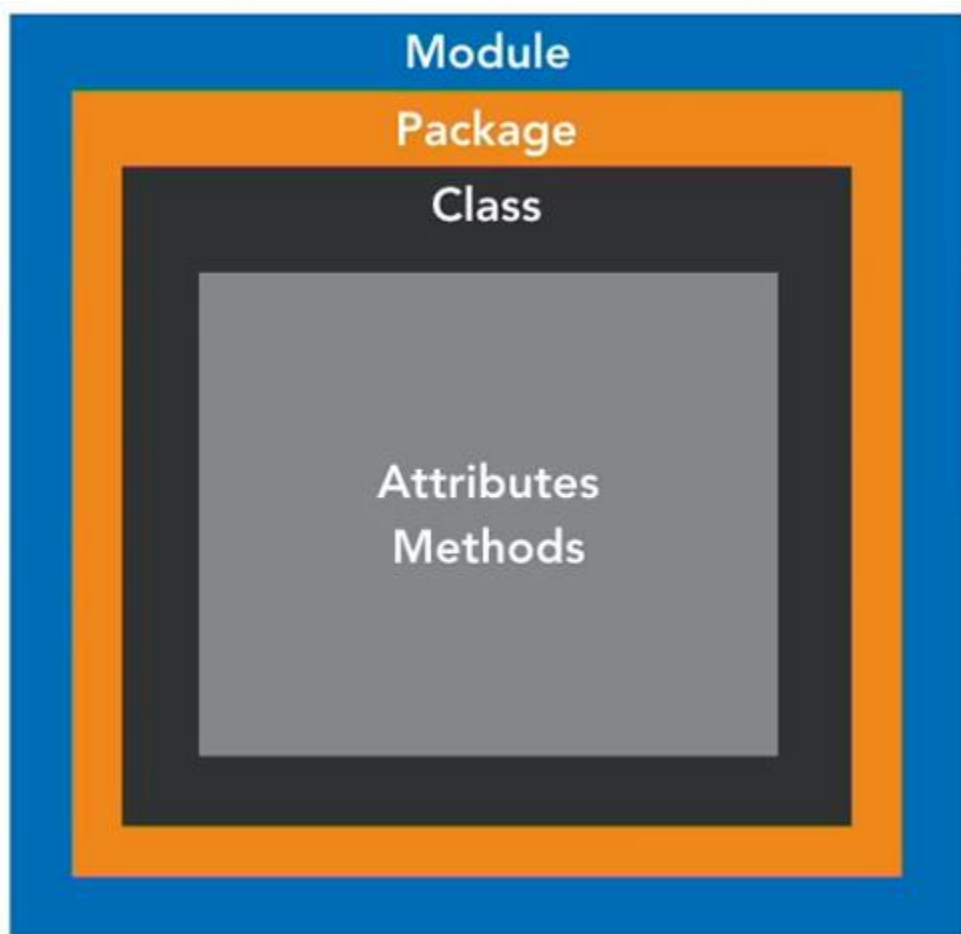


至于jmod文件，则主要用于构建可独立运行的“运行时环境”，有一个jlink工具可以完成这个工作，具体地说，就是jlink会将特定程序中用到的jmod模块抽取出来，构建出一个可独立运行的“文件集合”，这个文件集合仅包容它所用到的模块，移除了无关的模块，因此，比标准的JDK运行时环境要小得多，并且不强制要求目标计算机上预先安装有特定版本的JRE。

Java模块的内部结构图



Java模块代码的“分层”结构



保存模块代码的文件夹结构

```
AppFolder
  src
    ModuleNameFolder
      PackageFolders
        JavaSourceCodeFiles
        module-info.java
```

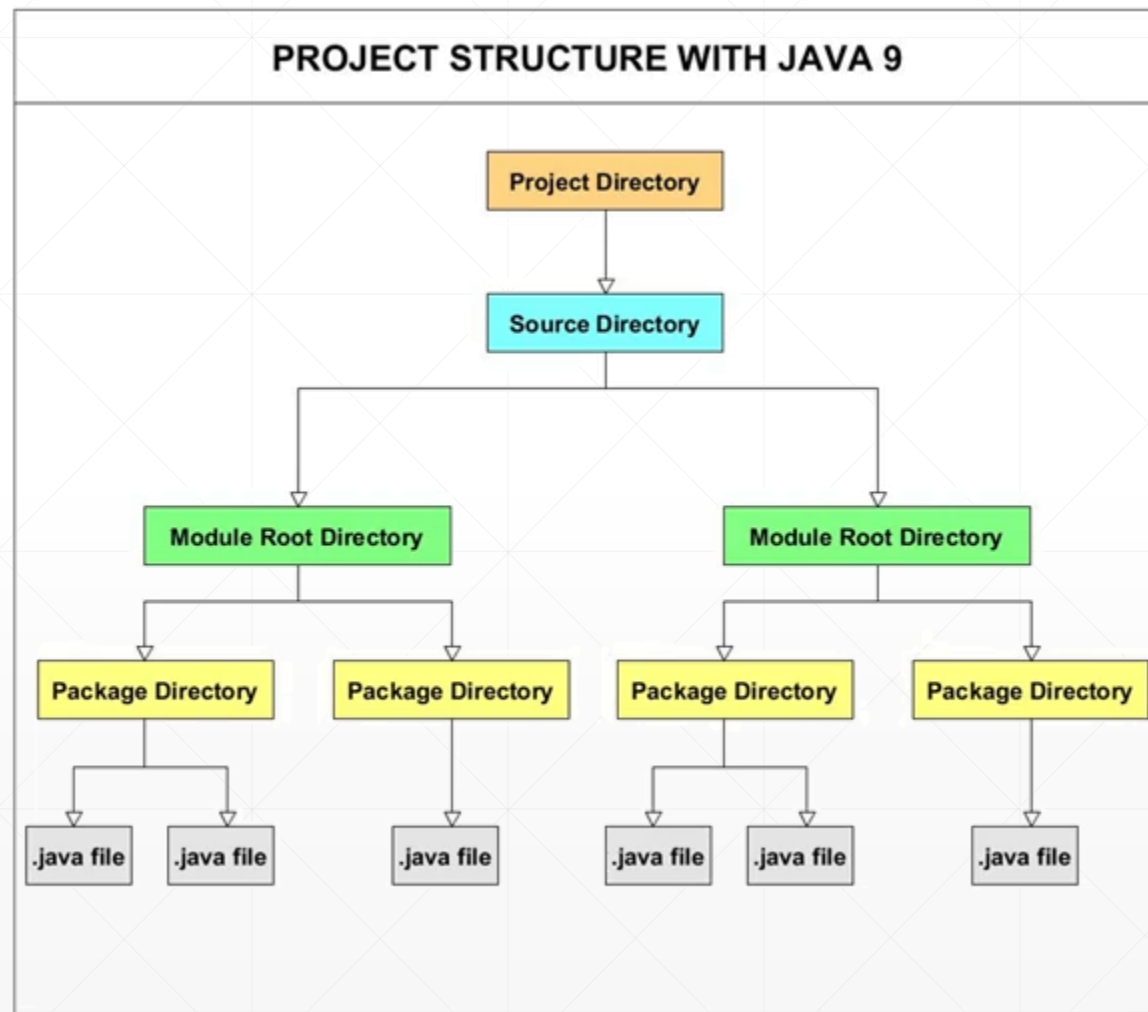
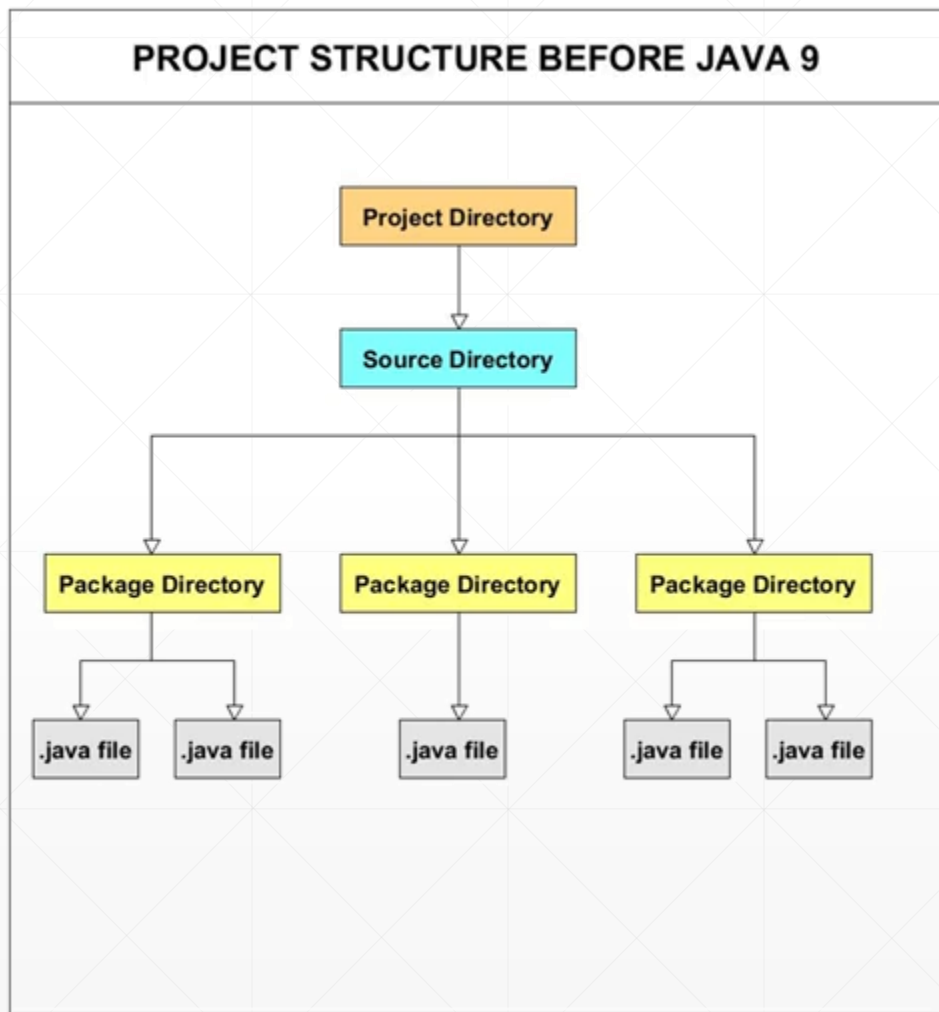
模块描述符 (module descriptor)

每个模块都关联着一个描述符，保存在一个名为 `module-info.java` 的文件中（这个文件名是固定不变的）：

```
module java.prefs {  
    requires java.xml; ❶  
  
    exports java.util.prefs; ❷  
}
```

- ①关键字**requires**表示一个依赖关系，说明本模块需要调用 `java.xml` 模块中的代码，即“依赖于” `java.xml` 模块。
- ②关键字**exports**表明本模块的 `prefs` 包被导出，即此包中的代码可以被其他模块调用。

模块化的Java项目



小结



本讲介绍了JDK模块化后的具体情况，并详细介绍了模块的内部结果及管理方法。



下一讲，将以当前业界广泛使用的IntelliJ IDEA为例，介绍如何使用它来开发一个模块化的Java应用。