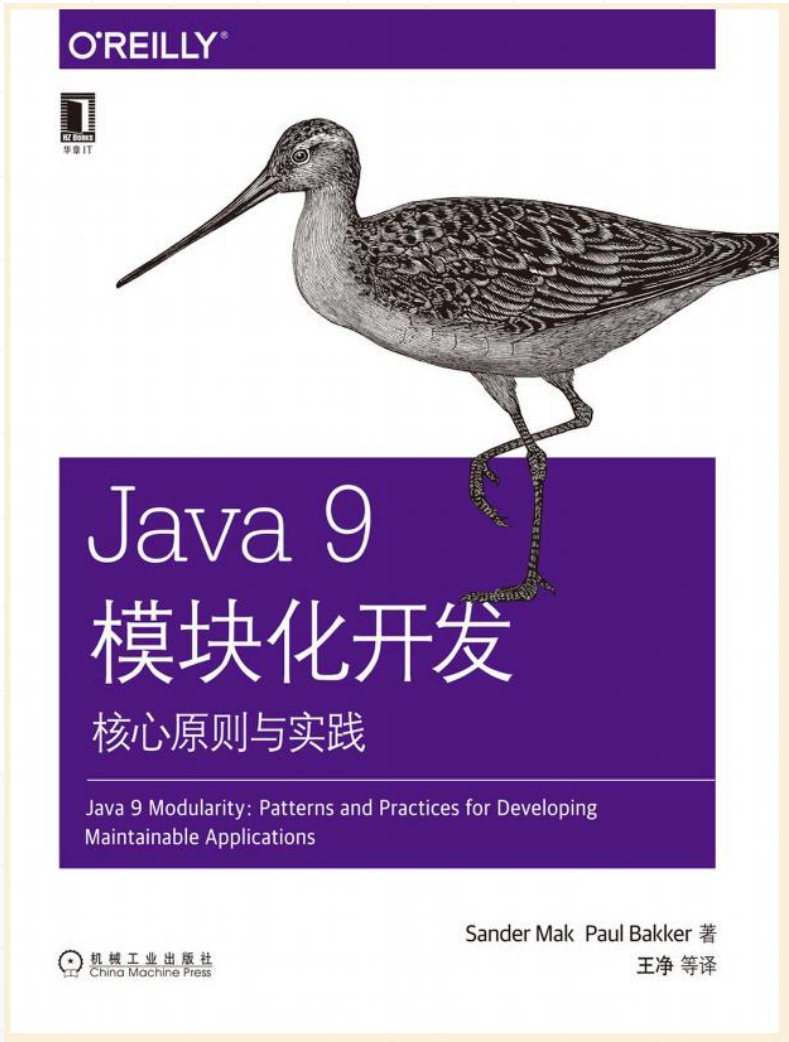
# Java模块化开发概述

北京理工大学计算机学院
金旭亮

# 参考书籍




本PPT之主要参考资料—《Java 9模块开发》

# 学习本部分的前提

- 计算机安装JDK 9以上版本
- 已经掌握Java面向对象编程基础知识（继承、接口、多态）
- 安装有IntelliJ IDEA的最新版本，并且掌握其基本用法。

# 概述

- JPMS（Java Platform Module System），是JDK 9引入的最重要的新特性之一。
- 模块化，对于开发大规模的软件系统，意义重大。

# 早期版本JDK存在的问题

- jar包之间的依赖关系难以管理，出现诸如循环依赖、版本冲突等问题。

- 随着版本的演化，JRE功能越来越多，相应地，也变得越来越庞大，一个Java应用即使只使用JRE中很少一部分的功能，也必须配上一个完整的JRE。

因此，JDK 9中引入了新的模块管理系统，以便解决上述问题。

# 为了实现模块化，早期Java是怎样做的?

package

访问修饰符：public、private等

interface

Maven

OSGi

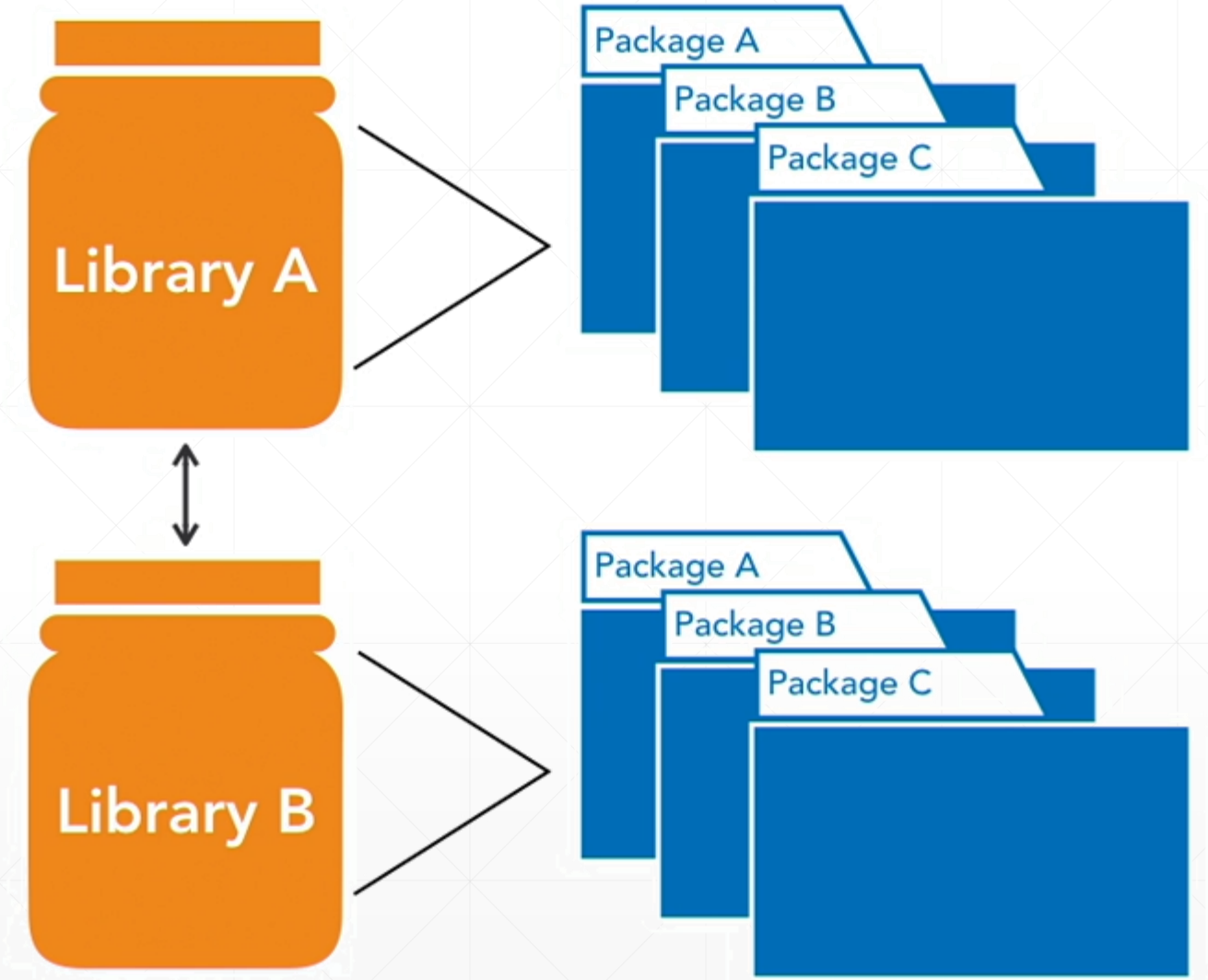
语言特性 ⟷ 构建机制

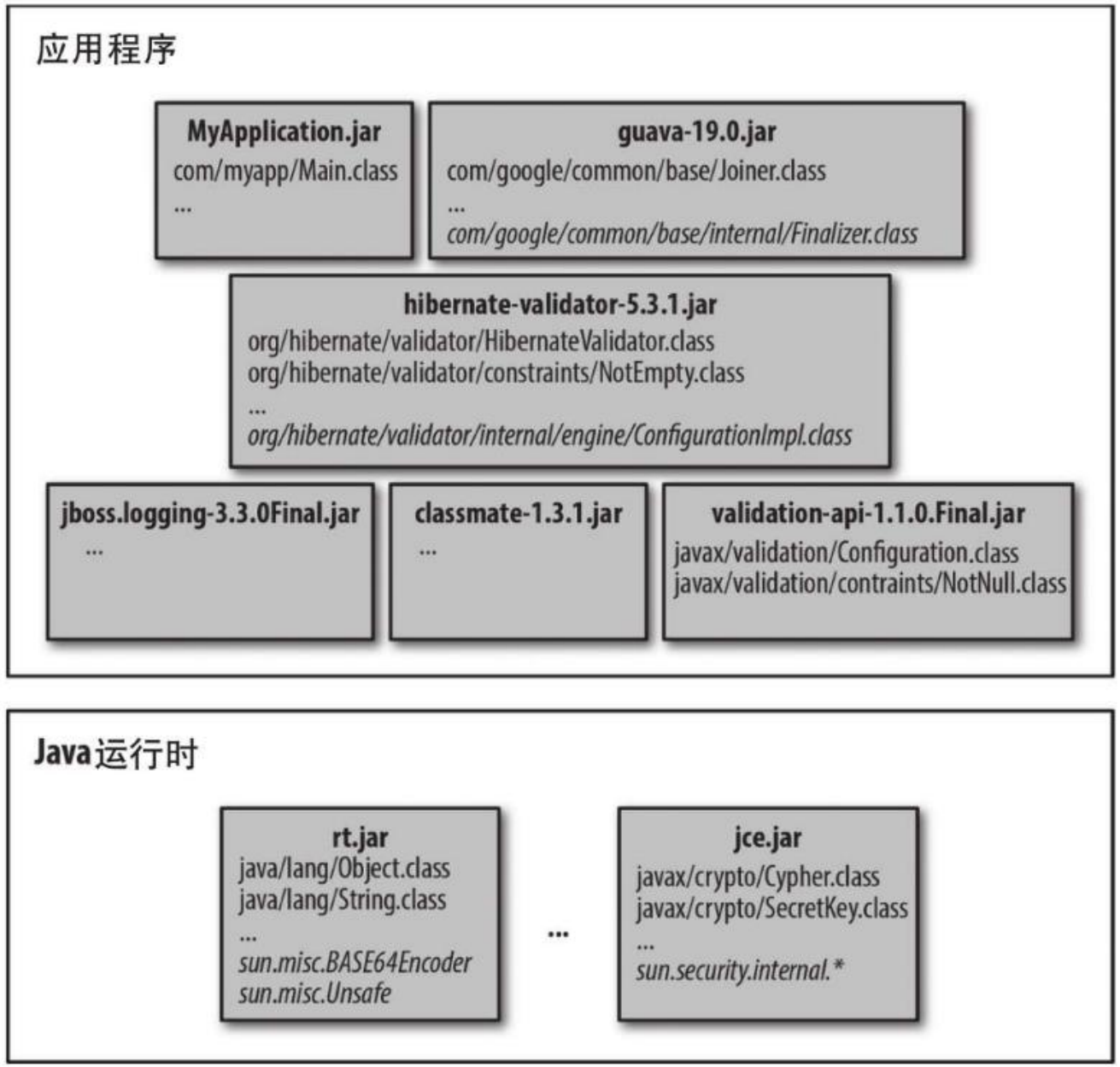
Jar文件

# 通过Jar包实现模块化

JDK9之前Java项目主要使用jar包作为代码重用单元。

jar包内可以包容多个包，包中包容多个类，jar包之间可以相互引用，构成“依赖”关系。






真正由程序员编写的应用程序放在MyApplication.jar中，它依赖于其它的jar包，而这些jar包可能又依赖于另外的jar包，构成一种多层依赖的关系。

jar包中的公有类可以随意访问，无法控制，为数众多的公有类，不仅增加了学习和使用的难度，而且提升了误用的危险。

JRE本身也需要调用jar包的功能，比如rt.jar，其中包含了Java标准库的类。

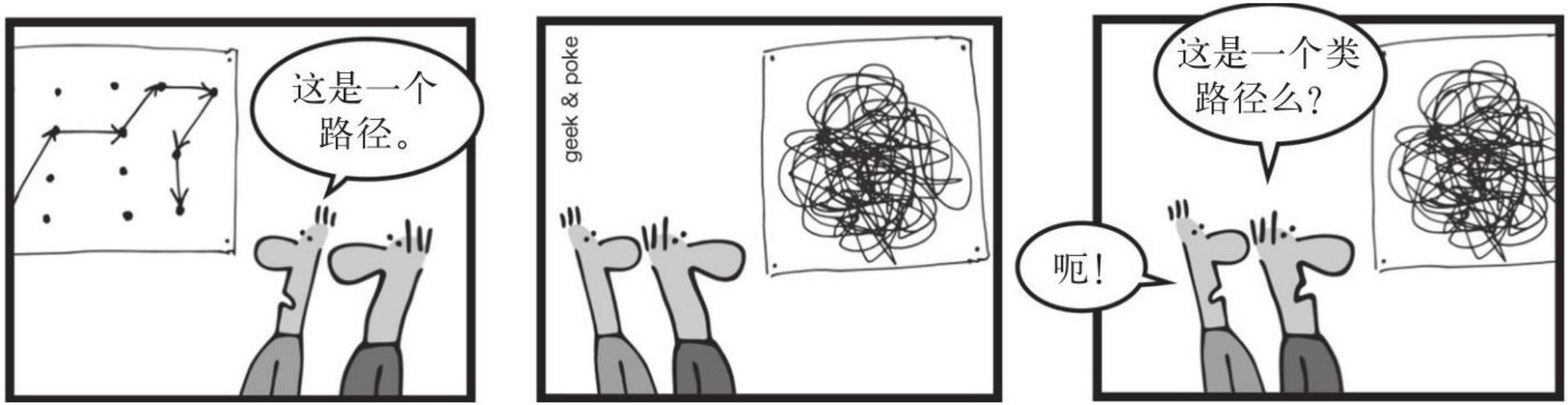
# jar包及其依赖的包必须“放在一起”才能运行

JAR不包含任何依赖信息，因此在运行它时，需要在命令行参数中显式指定所有它所依赖的jar包名，以保证用到的类都可以通过classpath找到：

```
java -classpath lib/guava-19.0.jar:\
                lib/hibernate-validator-5.3.1.jar:\
                lib/jboss-logging-3.3.0Final.jar:\
                lib/classmate-1.3.1.jar:\
                lib/validation-api-1.1.0.Final.jar \
     -jar MyApplication.jar
```

如果在类路径中没有找到所需的类，此时会得到一个运行时异常。JVM无法在应用程序启动时有效地验证类路径的完整性，即无法预先知道类路径是否是完整的，或者是否应该添加另一个JAR。

# 类路径地狱（ classpath hell）



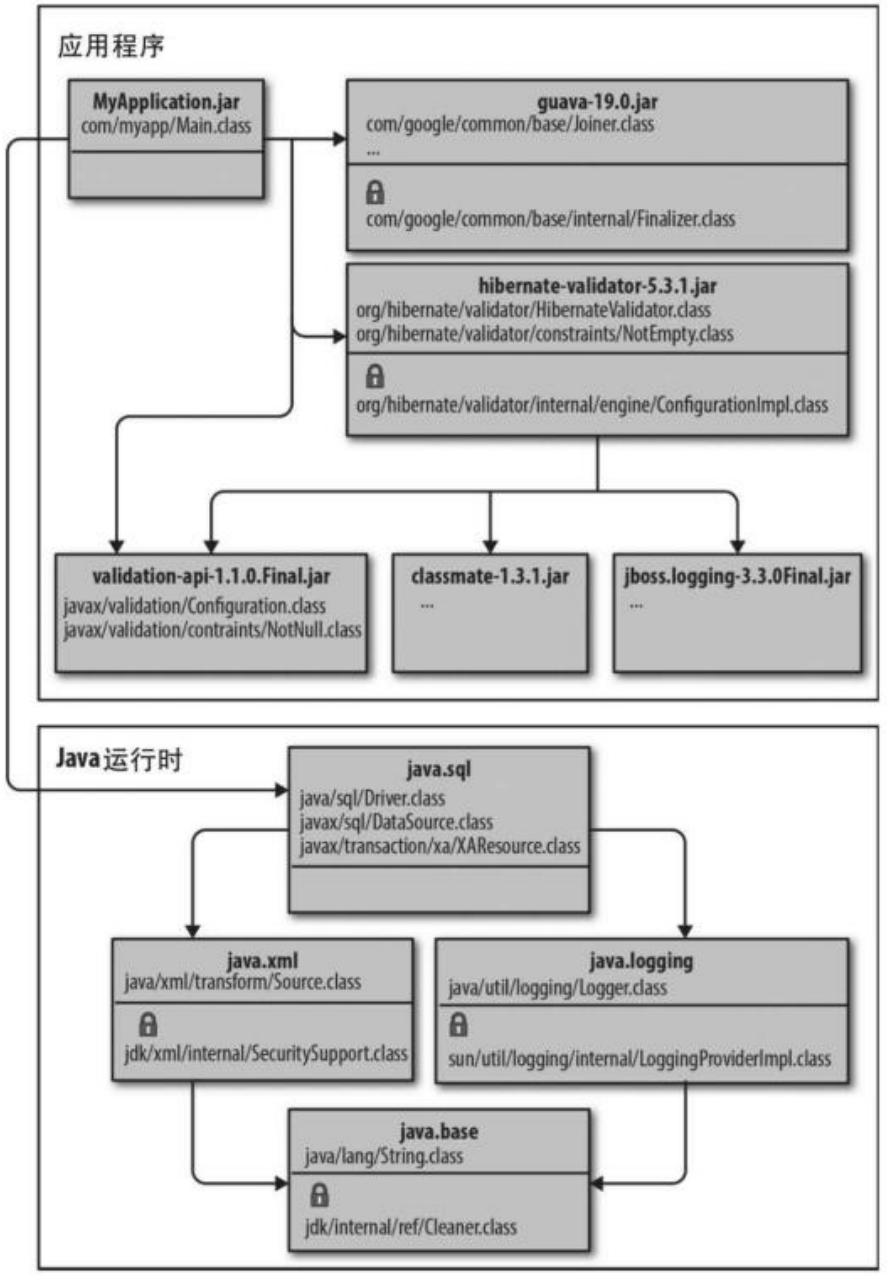

—图摘自《Java 9模块化开发》

# 类路径地狱（ classpath hell）解析

- 在Java程序运行时，JVM会从jar包中抽取所有类构成一张列表，然后顺序查找类。
- 假设由Maven根据POM中的显式依赖信息构建将放到类路径中的JAR集合。由于Maven以传递的方式解决依赖关系问题，因此该集合中出现相同库的两个版本（如Guava 19和Guava 18）是非常常见的。
- 现在，这两个库JAR中的类（可能同名）以一种未定义的顺序出现在类列表中，导致这个类的任一版本都可能会被首先加载。
- 有些类还可能会使用来自（可能不兼容的）其他版本的类。此时就会导致运行时异常。一般来说，当类路径包含两个具有相同（完全限定）名称的类时，只有一个会“获胜”。

# 模块化的Java应用


JPMS: Java Platform Module System




# 模块化的Java应用

模块化之后，整个程序被分解为若干个严格限定依赖关系的模块。

只要不显式声明**导出**，模块中的所有类外界都是不可访问的。

JRE也被模块化了，只有当前应用用到的那些模块被加载。

# 模块化与模块

**模块（module）**是包含代码的可识别软件构件，使用了元数据（metadata）来描述模块及其与其他模块的关系。可以把模块看成是一组用于代码重用的包（package）。

**模块化（ modularization）**是指将系统分解成独立且相互连接的模块的过程。

# 模块设计必须遵循的三个核心原则

强封装性

定义良好的接口

显式依赖

# 小结

本讲介绍了Java模块化开发的技术背景及必要性，了解这些背景知识是有必要的。

下一讲，介绍Java平台（JDK）的模块化现状及相关的基础知识