1.java重构成go的过程中，解决传递依赖的问题

Java 编译器可以编译存在循环依赖的类。在 Java 中，当两个类相互引用对方时，这并不会造成问题，因为 Java 的编译过程可以正确处理类文件的生成和解析

Go 语言设计时明确禁止了包级别的循环依赖。如果包 `package1` 导入了包 `package2`，那么 `package2` 就不能再导入 `package1`。如果出现这样的循环依赖，Go 编译器将报错并拒绝编译程序

Java 编译器通过多遍扫描源代码来处理循环依赖。在第一遍扫描中，编译器收集所有类的声明信息，但并不立即检查方法体内部的代码。这意味着即使两个类相互引用，只要它们尚未使用对方的成员，编译器就能够成功完成这一遍

Go 编译器并不是传统意义上的单遍编译器，它能够处理复杂的源文件和包依赖关系。但是，在处理这些依赖时，它要求依赖图是无环的，并且它不支持包级别的循环依赖

一个 `.java` 文件通常会包含至少一个类，因为类是 Java 编程语言构建应用程序的基本单位

在 Go 中，一个包对应一个目录，包内所有源文件都属于同一个命名空间，也就是说，同一个包下的所有文件彼此之间是可见的（相当于，Go导包，就把另一个文件的**所有类都导入**进来了，很容易形成循环依赖）

解决方法

从高层次审视应用的架构，理解各个组件之间的关系并寻找解耦的机会

确保每个包都有一个清晰且单一的职责，在逻辑上保持尽可能的独立