1. 模块化

（一）轻量化

为了让Java实现轻量化，Java9正式推出了模块化系统，Java 拆分成多个模块，允许Java程序可以根据需要选择加载模块，这样，Java就能够轻量化地运行。

（二）概述

一个项目Project中包含多个模块，每个模块中又包含有多个包，每个包中包含有多个java文件。模块相互之间是独立的，没有描述文件的设置，互相之间是没有访问权限的。

1. 模块的使用
2. 创建模块

正常创建模块的方法。

1. 描述性文件

在模块的src目录下新建一个名为module-info.java的描述性文件，该文件专门定义模块名，访问权限，模块依赖等信息。描述性文件中使用模块导出和模块依赖来进行配置并使用。

1. 导出包

想让某个模块的某个包被其他模块访问使用，则在描述性文件中导出该包。

导出包格式： exports 包名

如：导出one模块下的xx包：

Module one{

Exports xx;

}

1. 依赖模块

前面one模块将xx包导出，若two模块想使用one模块中的包，则必须在描述性文件中指明依赖one模块。

依赖模块格式： requires 模块名

如：

Module two{

Requires one;

}

1. 模块服务
2. 概述

Java提供了服务机制，允许服务提供者和服务使用者之间完成解耦 ，即使用者只面向接口编程，不清楚提供者的实现类。

1. 模块服务的使用
2. 描述性文件设置

服务接口定义在一个模块中，针对该服务接口提供不同的服务实现类，这些服务实现类可以分布在不同的模块中，服务实现类模块使用provides语句为服务接口指定实现类。而服务的使用者需要声明uses语句来声明要使用该接口，但对于实现类是谁并不清楚，这就将使用者和提供者分开了。

如：

在one模块下定义接口Service，在包xx下定义实现类A，用实现类A实现接口Service，在描述性文件中写入：

Module one{

Exports xx;

Provides Serviice with A;

}

Two模块要使用Service接口，则在two模块的描述性文件中写入uses 接口名。

Module two{

Requires one;

Uses Service;

}

1. 使用服务

ServiceLoader：一种加载服务实现的工具

加载服务的Class对象，并遍历服务。注意参数是服务接口的class对象。

如：

ServiceLoader<Service> mys = ServiceLoader.load(Service.class);

for(Service my : mys){

my.service();

}