东北大学秦皇岛分校

计算机与通信工程学院

Java 程序设计大作业

设计题目 基于 socket 的网络类加载器

专业名称	计算机科学与技术
班级学号	计科 02-20188068
班级序号	35
学生姓名	孔天欣
指导教师	王聪

目 录

Java 程序设计大作业

——基于 Socket 的网络类加载器

一、题目描述

1.1 问题需求

1.设计一个基于 socket 或者 RMI 的服务器，可以将字节码文件（class 文件）加密后发送出去。加密方法可采用教材例子 3.1 中的异或加密。

2.设计一个接口 displayable，包名任意，接口中定义方法 display。

3.设计一个类 x 去实现 displayable 接口，在 display 中任意输出一段话到控制台。将该类的字节码部署到服务器。

4.设计一个可以通过网络获取字节码并解密的类加载器。

5.设计一个客户端程序，使用网络类加载器，从服务器获取类 x 的字节码，通过接口回调，调用 display 方法。

1.2 解决方案

1. 建立一个服务器类和客户端类，服务器类读取类字节码并进行 RC4 加密，并和该类对应方法名一起封装成另一个实体类，将该实体类的实例进行 socket 传输。

2. 客户端类通过 socket 接收到实例后，解析该实例包含的字节码，将字节码进行解密，交给自定义的类加载器去生成类对象。

3. 客户端解析该实例包含的类方法名，取出目标类的方法对象和方法名一一比对，符合的就使用目标类对象的实例去执行该方法。

4. 由于直接写字节码路径或者类名会导致程序耦合度大，且可复用性低，因此采用注解的方式，对客户端需要调用的类和方法分别加上对应的注解，到时候服务端加载一个新类，该类负责扫描本项目所有类，获取含此注解的类和方法，通过反射机制分别将其加密后的字节码和方法名放入实体类中，回送给服务端类。

二、程序设计

2.1 程序概览

2.1.1 UML 类图

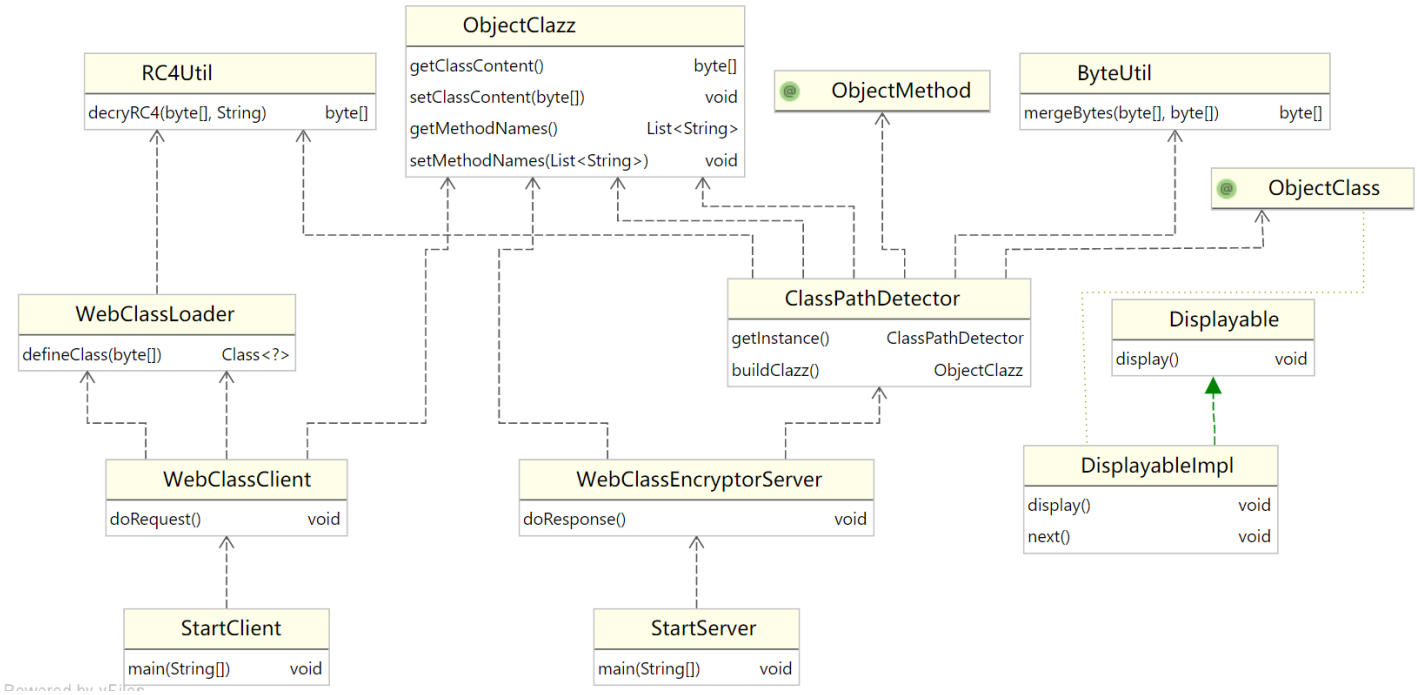

图 2.1.1 UML 类图

2.2 代码文档

2.2.1 类 WebClassEncryptorServer

- com.strutnut.webcloader. WebClassEncryptorServer

WebClassEncryptorServer 作为服务端类，用于通过 socket 接受来自客户端之请求，然后通过 ClassPathDetector 类获取目标类的加密字节码及其方法名数组，封装为 ObjectClazz 类实例，序列化后通过 socket 发送至客户端。

表 2.2.1 WebClassEncryptorServer 的变量

修饰符和类型	字段	说明
private static final int	SERVER_PORT	监听的端口号

表 2.2.2 WebClassEncryptorServer 的方法

修饰符和类型	方法	说明
void	doResponse()	处理客户端请求
private void	doSend(java.net.Socket clientSocket)	发送数据包

2.2.2 类 ClassPathDetector

- com.strutnut.webcloader. ClassPathDetector

该类负责扫描整个项目文件夹，寻找含有@ObjectClass 注解的类中的含@ObjectMethod 注解的方法（在本报告中为 DisplayableImpl 类的 display()和 next()方法，可任意改变），加工得到该类的加密字节码及其方法名数组，并将其封装为 ObjectClazz 类返回。

表 2.2.3 ClassPathDetector 的变量

修饰符和类型	字段	说明
private java.lang.String	classPath	带@ObjectClass 类的全类名（单个）
static ClassPathDetector	classPathDetector	单例对象
private static final java.lang.String	GROUP_ID	项目坐标名
private static final java.lang.String	KEY	加密密匙
private java.util.List<java.lang.String>	methodNames	带@ObjectMehod 的方法集合

表 2.2.4 ClassPathDetector 的方法

修饰符和类型	方法	说明
ObjectClazz	buildClazz()	将 classPath 对应的类字节码加密，并和 methodNames 填入实体类 ObjectClazz 中
static ClassPathDetector	getInstance()	获得单例对象

修饰符和类型	方法	说明
private java.lang.String	searchClassPath()	如果 classPath 已存在，返回之，否则调 searchClassPath(java.lang.String path)方法返回 classPath
private void	searchClassPath (java.lang.String path)	扫描项目所有类，寻找含@ObjectClass 的类以及含有的@ObjectMethod 方法，分别将其全类名和方法名数组放入 classPath 和 methodNames 变量中

2.2.3 类 WebClassClient

- com.strutnut.webcloader. WebClassClient

WebClassClient 作为客户端类，用于向服务端发送请求并接受服务端响应之数据，将数据通过 WebClassLoader 类加载器生成类对象，并该类对象生成实例，以该实例调用返回的方法名数组对应的方法，实现调用该类方法功能。

表 2.2.5 WebClassClient 的变量

修饰符和类型	字段	说明
private static final java.lang.String	IP_ADDRESS	IP 地址
private static final int	OBJECT_PORT	请求端口

表 2.4.2 WebClassClient 的方法

修饰符和类型	方法	说明
void	doRequest()	向服务端发出请求，并处理服务端响应，并于最后完成执行实例方法功能

2.2.4 类 WebClassLoader

- java.lang.ClassLoader

- com.strutnut.webcloader.WebClassLoader

类加载器，继承了抽象类 ClassLoader,负责分析 ObjectClazz 实例中的字节数组 classContent，解密字节码并生成类对象。

表 2.2.6 WebClassLoader 的变量

修饰符和类型	字段	说明
private static final java.lang.String	KEY	密匙

表 2.2.7 WebClassLoader 的方法

修饰符和类型	方法	说明
private byte[]	decrypt(byte[] classContent)	解密字节码
java.lang.Class<?>	defineClass(byte[] classContent)	生成类对象

2.2.5 类 ObjectClazz

- com.strutnut.bean.ObjectClazz implements java.io.Serializable

实体类，内含目标类加密后的字节码数组和该类方法名集合，用于携带类信息，同时被序列化后进行网络传输。

表 2.2.8 ObjectClazz 的变量

修饰符和类型	字段	说明
private byte[]	classContent	加密的字节码
private java.util.List<java.lang.String>	methodNames	方法名

2.2.6 类 ByteUtil

- com.strutnut.utils.ByteUtil

工具类，负责处理字节相关内容。

表 2.2.9 ByteUtil 的方法

修饰符和类型	方法	说明
static byte[]	mergeBytes(byte[] firstBytes,	合并字节数组

修饰符和类型	方法	说明
	byte[] secondBytes)	

2.2.7 类 RC4Util

- com.strutnut.utils. RC4Util

工具类，负责加密和解密字节数组，加密算法采用的是 RC4 异或加密算法。

表 2.2.10 RC4Util 的变量

修饰符和类型	字段	说明
private static int	LENGTH	加密长度

表 2.2.11 RC4Util 的方法

修饰符和类型	方法	说明
static byte[]	decry(byte[] data, java.lang.String key)	检查输入规范，并调用 RC4()方法加密
private static byte[]	RC4(byte[] data, java.lang.String keyWithoutInited)	对字节数组进行加密
private static byte[]	initKey (java.lang.String stringKey)	根据 key 创建向量并交换元素

2.2.8 接口 Displayable 和 实现类 DisplayableImpl

- com.strutnut.webcloader.tar.DisplayableImpl implements Displayable

DisplayableImpl 是本次实验中需要获得字节码内容并被调用其中方法的类，是接口 Displayable 的实现，类带有@ObjectClass 注解。

表 2.2.12 DisplayableImpl 的方法

修饰符和类型	方法	说明
void	display()	@ObjectMethod 被调用的方法 1
void	next()	@ObjectMethod 被调用的方法 2

三、运行结果及测试

3.1 截图和描述

首先通过 StartServer 类的 main()方法加载 WebClassEncryptor 类，也就是启动服务端，然后调用 doResponse()方法，如图 3.1.1 所示：

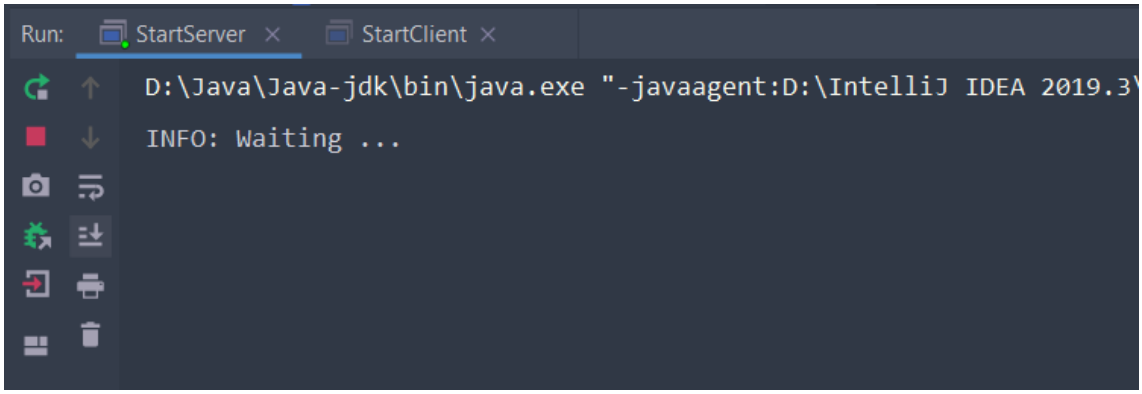

图 3.1.1 服务器日志清单（一）

可见，在得到客户端的请求前，服务器处于等待就绪的状态。

接下来通过 StartClient 类的 main()方法加载 WebClassClient 类，也就是启动客户端，然后调用 doRequset()方法，向服务端发送了请求。服务端在方法 accept()接收到请求后，开辟一条线程为客户端服务，并调用 ClassPathDetector 类的方法集扫描项目所有类，在其中找到客户端需要的带有@ObjectClass 的类对象，将该类字节码数组进行 RC4 加密，并把它和其带有@ObjectMethod 方法的名字集合一块放入 ObjectClazz 类。（在本例中，将 DisplayableImpl 类的字节码和它的所有方法名字放入了 ObjectClazz 类的 classContent 和 methodNames 变量中），将该类序列化后进行 socket 传输作为响应，服务端操作日志清单如图 3.1.2 所示：

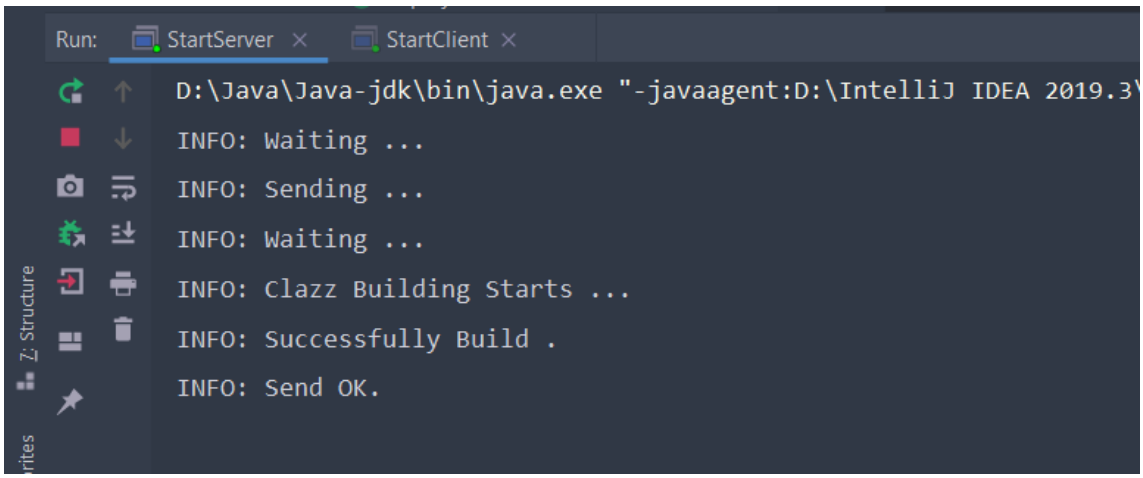

图 3.1.2 服务器日志清单（二）

在服务端成功响应后，客户端从 socket 的输入流中成功反序列化读取到带有目标类一系列消息的 ObjectClazz 类，并调用 WebClassLoader 类的一系列方法集，将其中的加密字节码数组 classContent 进行解密，然后生成 class 类对象，接着，客户端再从 ObjectClazz 类中获取 methodNames，和类对象包含的方法对象一一比对，相同的就执行。

客户端的日志清单如图 3.1.3 所示。

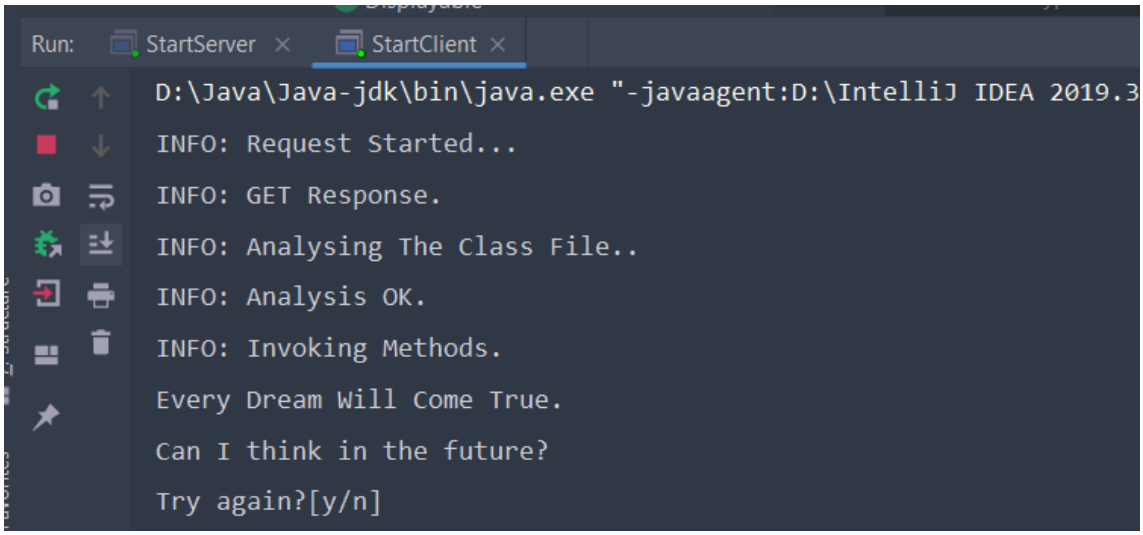

图 3.1.3 客户端日志清单

在本例中，被调用的类和方法定义如下：

```
1. @ObjectClass
2. public class DisplayableImpl implements Displayable {
3.
4.     @Override
5.     @ObjectMethod
6.     public void display() {
```

```
7.          System.out.println("Can I think in the future?");
8.      }
9.
10.     @ObjectMethod
11.     public void next() {
12.         System.out.println("Every Dream Will Come True.");
13.     }
14.}
```

可见，客户端成功生成了目标类对象的实例，并调用了实例的两个方法。

综上所述，这样就实现了两端网络传输指定字节码并执行其指定方法的功能。用注解的方式使得耦合性降低，令代码可以在任何项目运行，而且更加灵活，也能够不拘泥于指定类的方法，可用性高。但本例中的注解仅能够调用不含参的公有方法，且只能指定一个类，因此有一定的局限性，仅仅是一个简易的想法和实现。