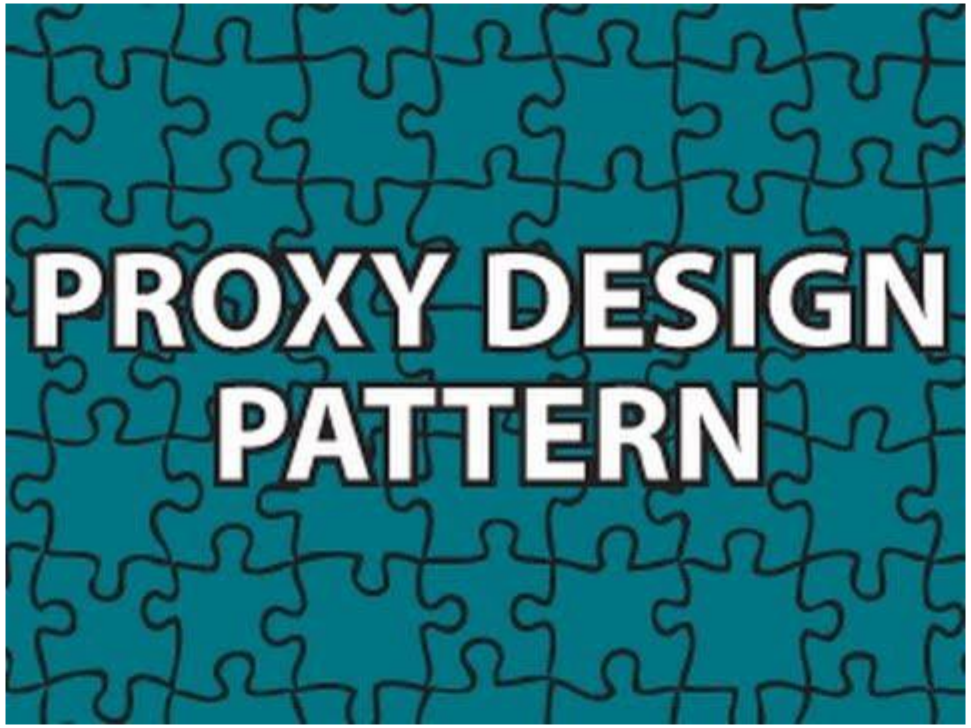
代理模式Proxy

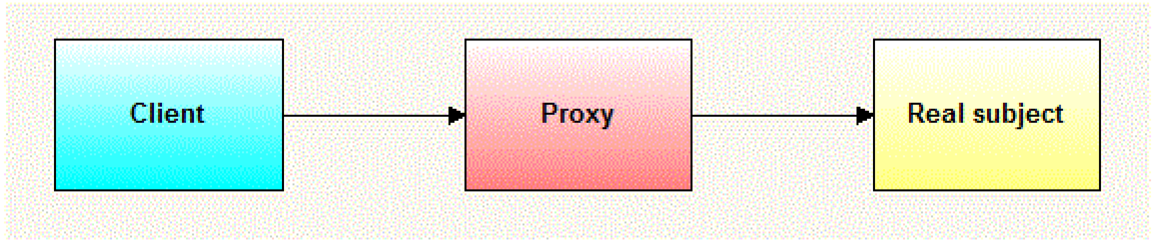
波波老师~研发总监/资深架构师

波波微课
spring2go.com



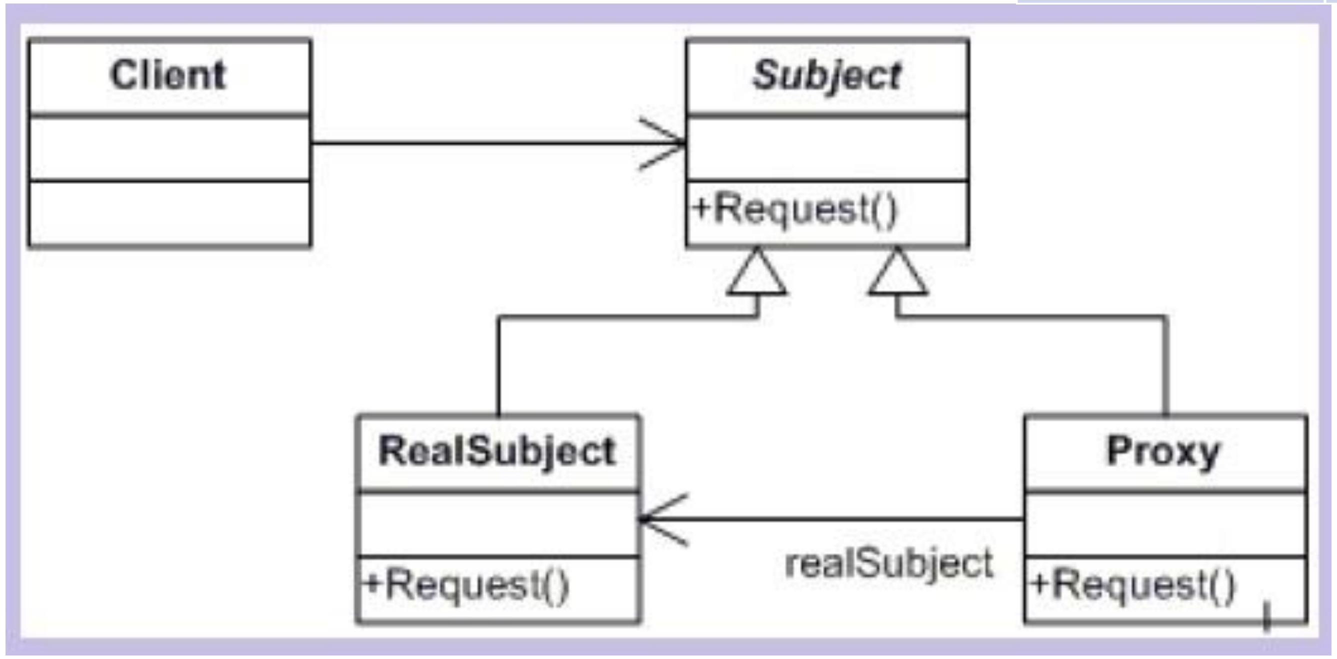
定义

- 提供一个中间代理，以控制对实际对象的访问
  - 封装对目标对象访问的复杂性
  - 提供额外功能



关系图

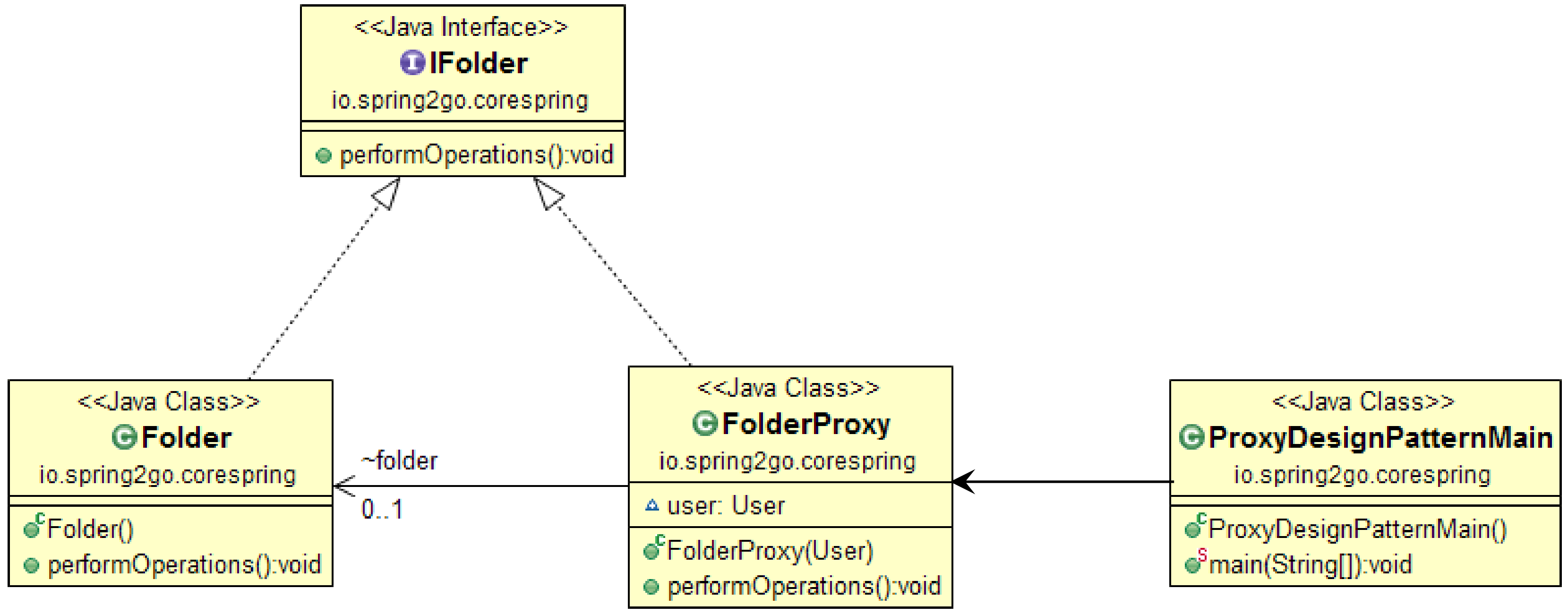
角色	职责
Subject	定义RealSubject和Proxy的公共接口； 让Proxy在任意需要的地方可以替代RealSubject
RealSubject	Proxy所代表的实际对象
Proxy	维护对RealSubject的引用； 同样实现Subject； 控制对RealSubject的访问；



代理种类

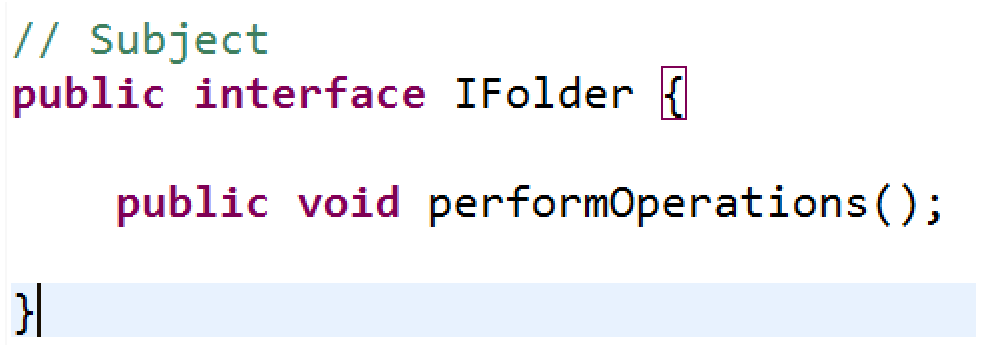
名称	场景
远程代理(remote proxy)	为不同地址空间中的对象提供一个本地表示； 为远程Web Services/REST API提供一个接口
虚代理(virtual proxy)	按需创建耗时耗资源对象
保护代理(protection proxy)	控制对原始对象的访问权限

案例~目录访问控制

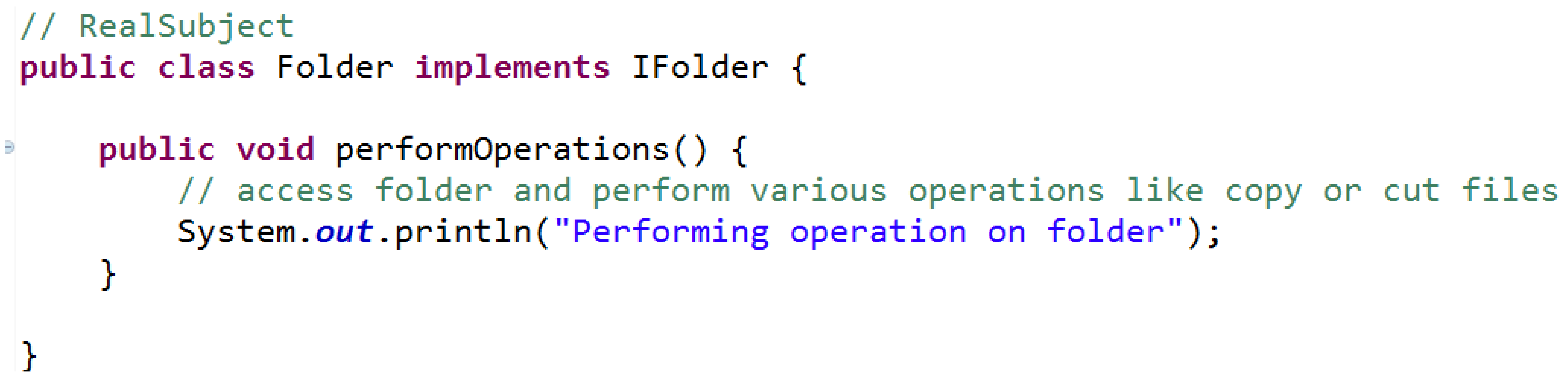


<<Java Interface>>
IFolder
io.spring2go.corespring
performOperations():void
<<Java Class>>
Folder
io.spring2go.corespring
Folder()
performOperations():void
~folder
0..1
<<Java Class>>
FolderProxy
io.spring2go.corespring
user: User
FolderProxy(User)
performOperations():void
<<Java Class>>
ProxyDesignPatternMain
io.spring2go.corespring
ProxyDesignPatternMain()
main(String[]):void

代码~Subject & RealSubject

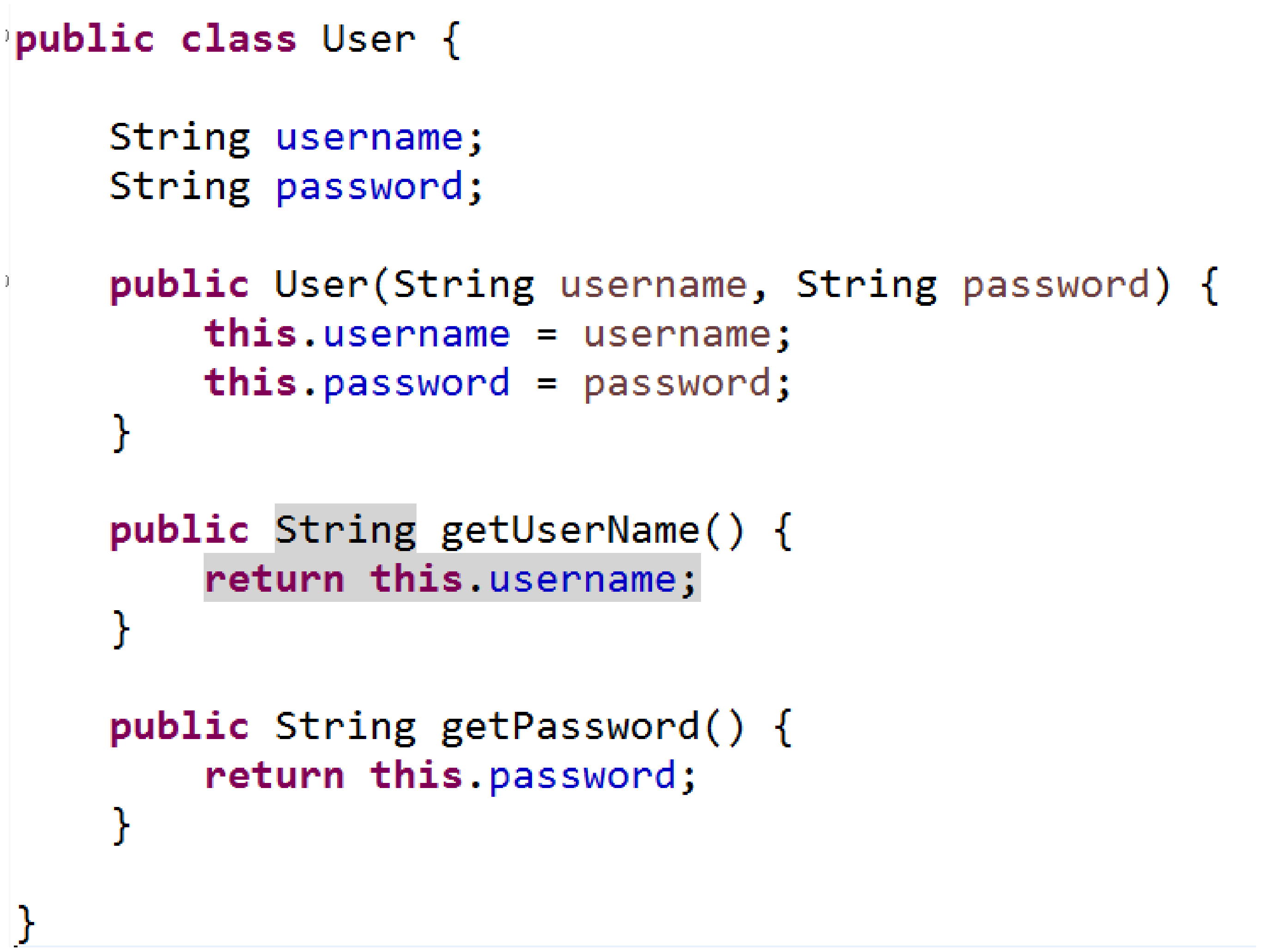


```java
// Subject
public interface IFolder {

    public void performOperations();

}
```



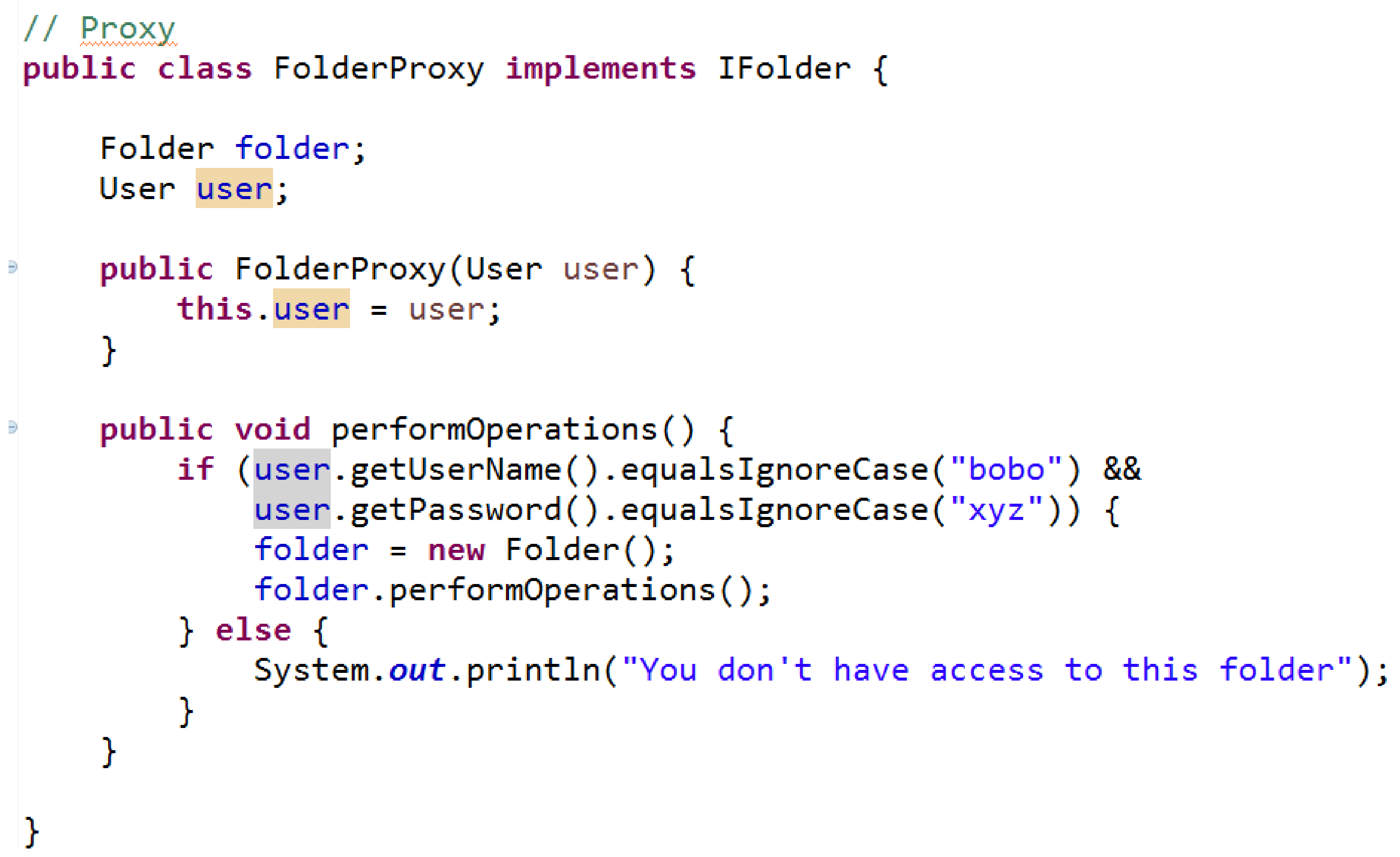
```java
// RealSubject
public class Folder implements IFolder {

    public void performOperations() {
        // access folder and perform various operations like copy or cut files
        System.out.println("Performing operation on folder");
    }

}
```

代码~User



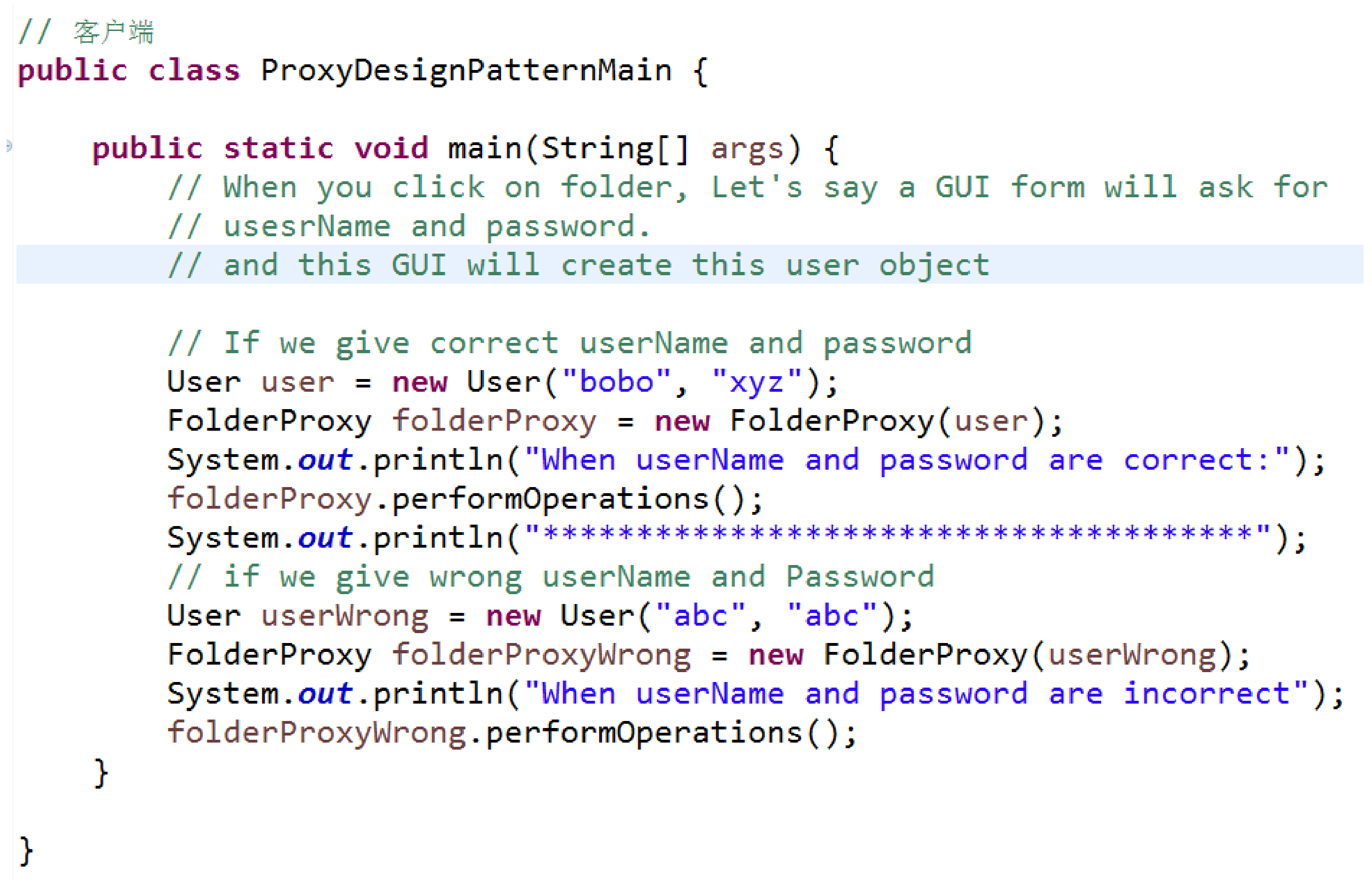
```java
public class User {

    String username;
    String password;

    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    public String getUserName() {
        return this.username;
    }

    public String getPassword() {
        return this.password;
    }

}
```

代码~Proxy

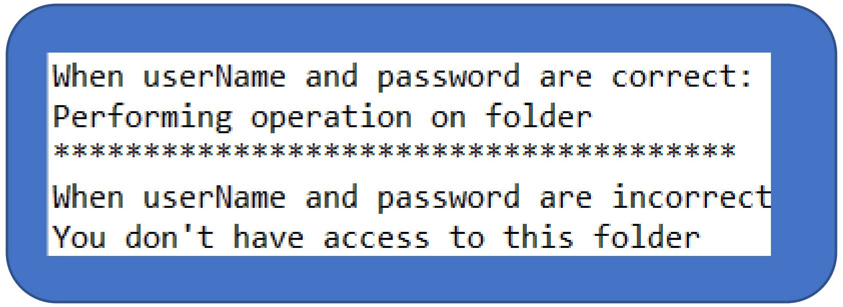


```java
// Proxy
public class FolderProxy implements IFolder {

    Folder folder;
    User user;

    public FolderProxy(User user) {
        this.user = user;
    }

    public void performOperations() {
        if (user.getUserName().equalsIgnoreCase("bobo") &&
            user.getPassword().equalsIgnoreCase("xyz")) {
            folder = new Folder();
            folder.performOperations();
        } else {
            System.out.println("You don't have access to this folder");
        }
    }

}
```

客户端代码



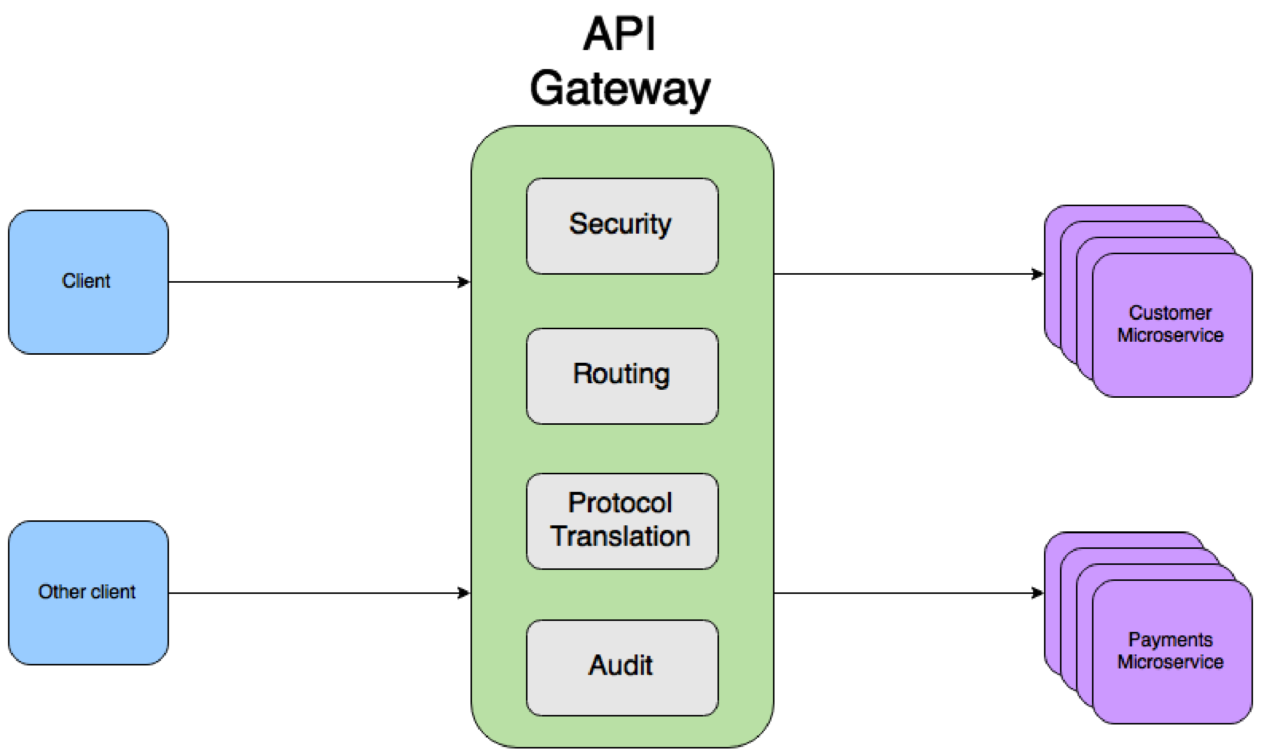
```java
// 客户端
public class ProxyDesignPatternMain {

    public static void main(String[] args) {
        // When you click on folder, Let's say a GUI form will ask for
        // usesrName and password.
        // and this GUI will create this user object

        // If we give correct userName and password
        User user = new User("bobo", "xyz");
        FolderProxy folderProxy = new FolderProxy(user);
        System.out.println("When userName and password are correct:");
        folderProxy.performOperations();
        System.out.println("**************************************");
        // if we give wrong userName and Password
        User userWrong = new User("abc", "abc");
        FolderProxy folderProxyWrong = new FolderProxy(userWrong);
        System.out.println("When userName and password are incorrect");
        folderProxyWrong.performOperations();
    }

}
```



```
When userName and password are correct:
Performing operation on folder
**************************************
When userName and password are incorrect
You don't have access to this folder
```

应用

- Spring
  - AOP
  - RMI
  - HTTP Invoker
- 微服务网关



问题

- 代理模式和适配器模式/装饰模式的差异？

参考

- Proxy design pattern in java
  - https://java2blog.com/proxy-design-pattern-in-java/

代码

- https://github.com/spring2go/core-spring-patterns

波波微课
spring2go.com

波波微课

spring2go.com