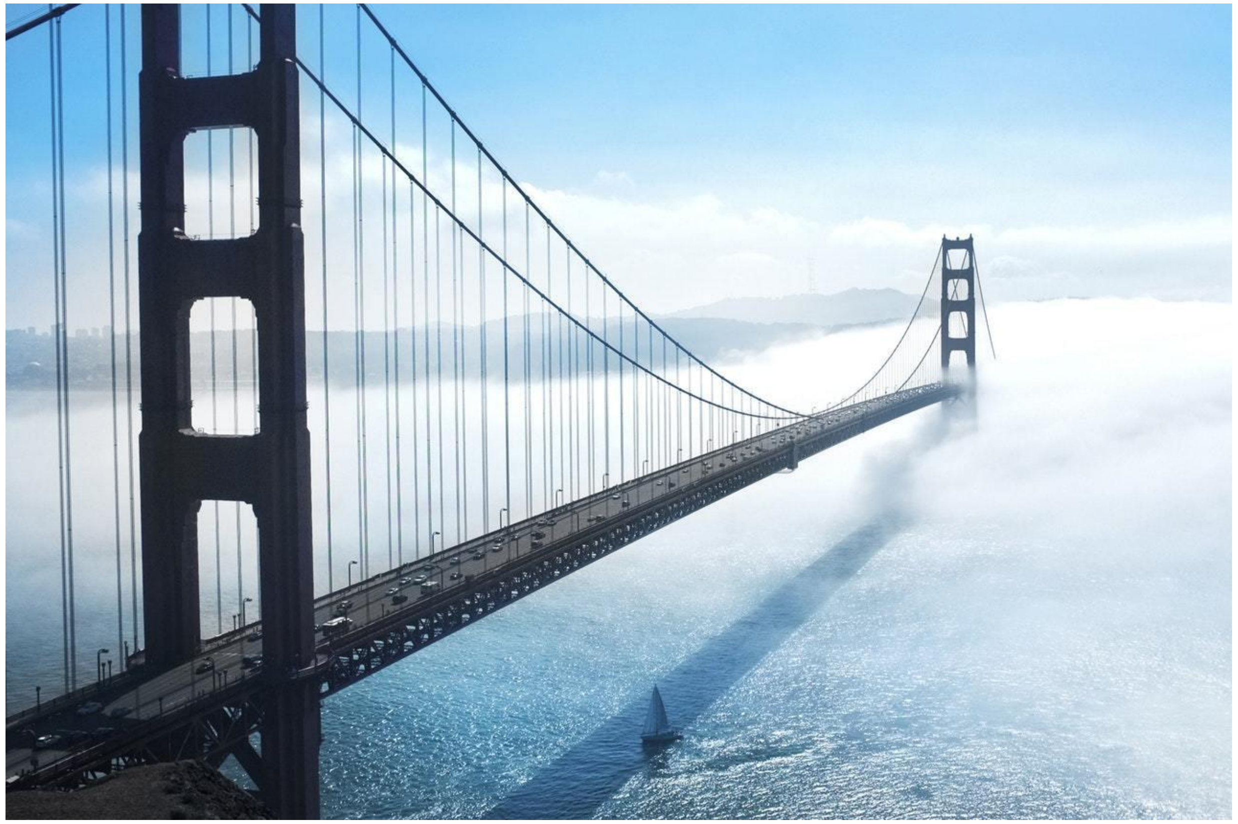
桥接器(Bridge)模式

波波老师~研发总监/资深架构师

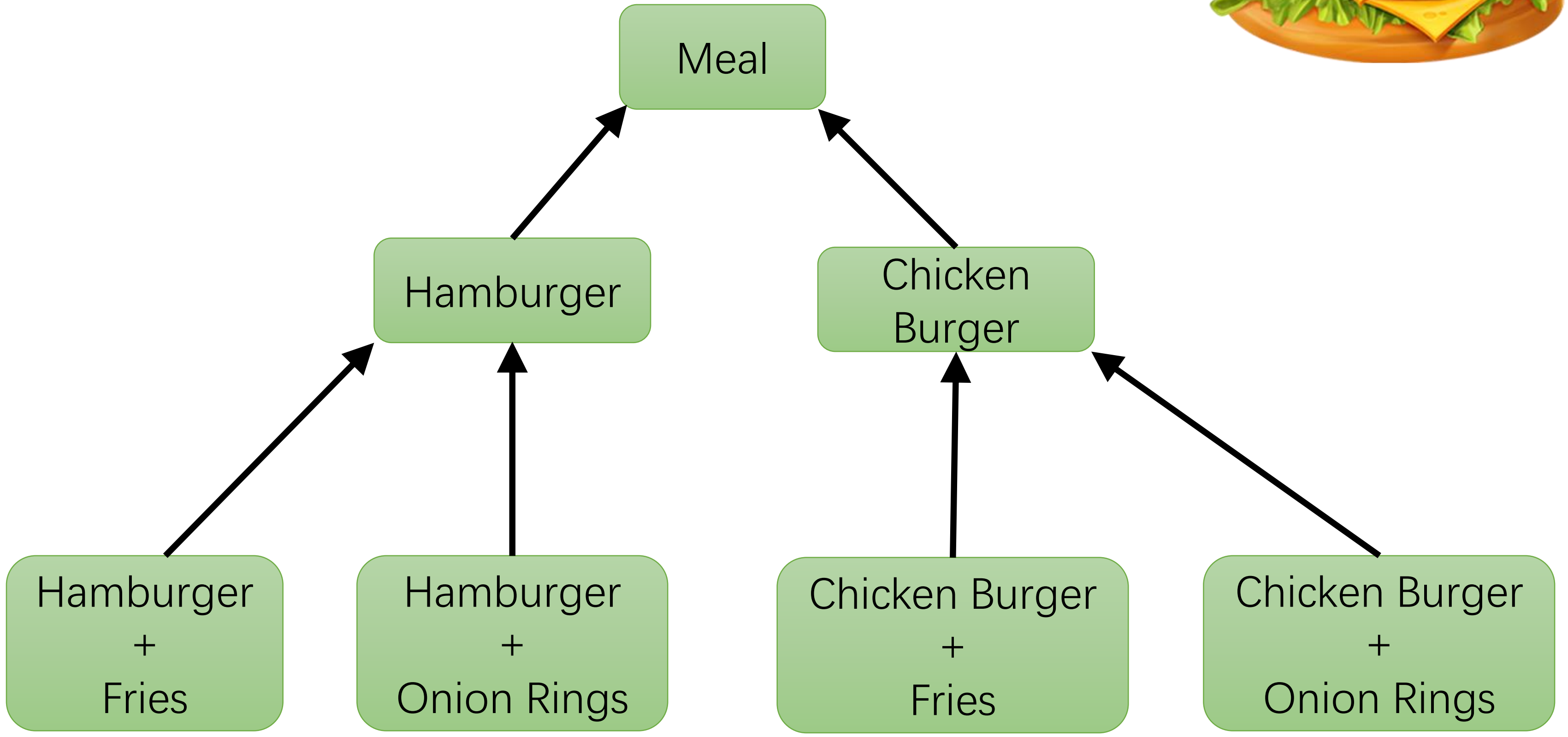
波波微课
spring2go.com



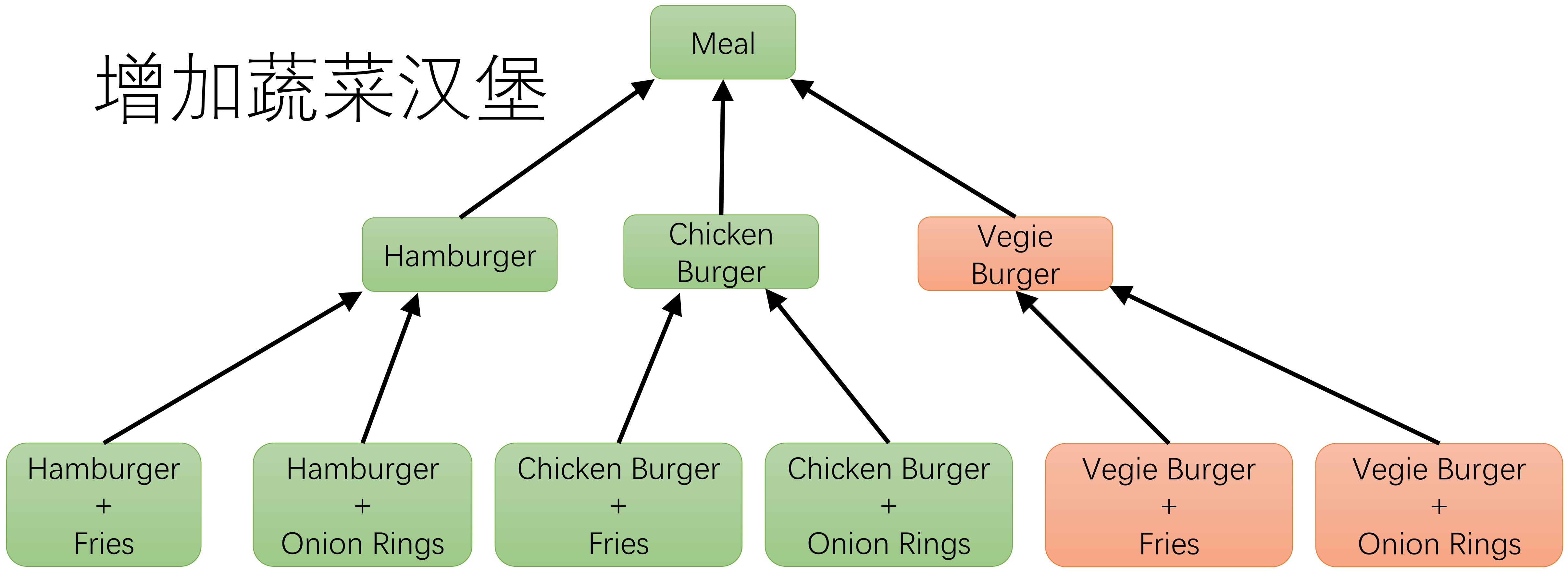
问题域

- 抽象和实现独立变化
- 类层级爆炸（exploding class hierarchy）

汉堡案例


Meal
Hamburger
Chicken Burger
Hamburger + Fries
Hamburger + Onion Rings
Chicken Burger + Fries
Chicken Burger + Onion Rings

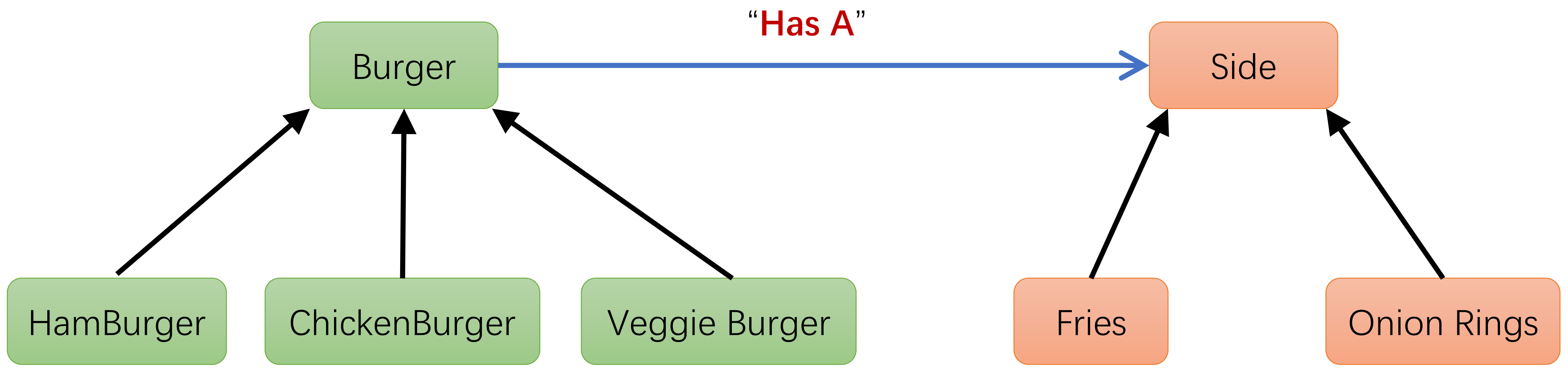
增加蔬菜汉堡


Meal
Hamburger
Chicken Burger
Vegie Burger
Hamburger + Fries
Hamburger + Onion Rings
Chicken Burger + Fries
Chicken Burger + Onion Rings
Vegie Burger + Fries
Vegie Burger + Onion Rings

设计原理

- **组合**优于**继承**(Prefer composition over inheritance)
- **HAS-A**比**IS-A**要好
  - 低耦合
  - 运行期改变行为

两层桥接结构

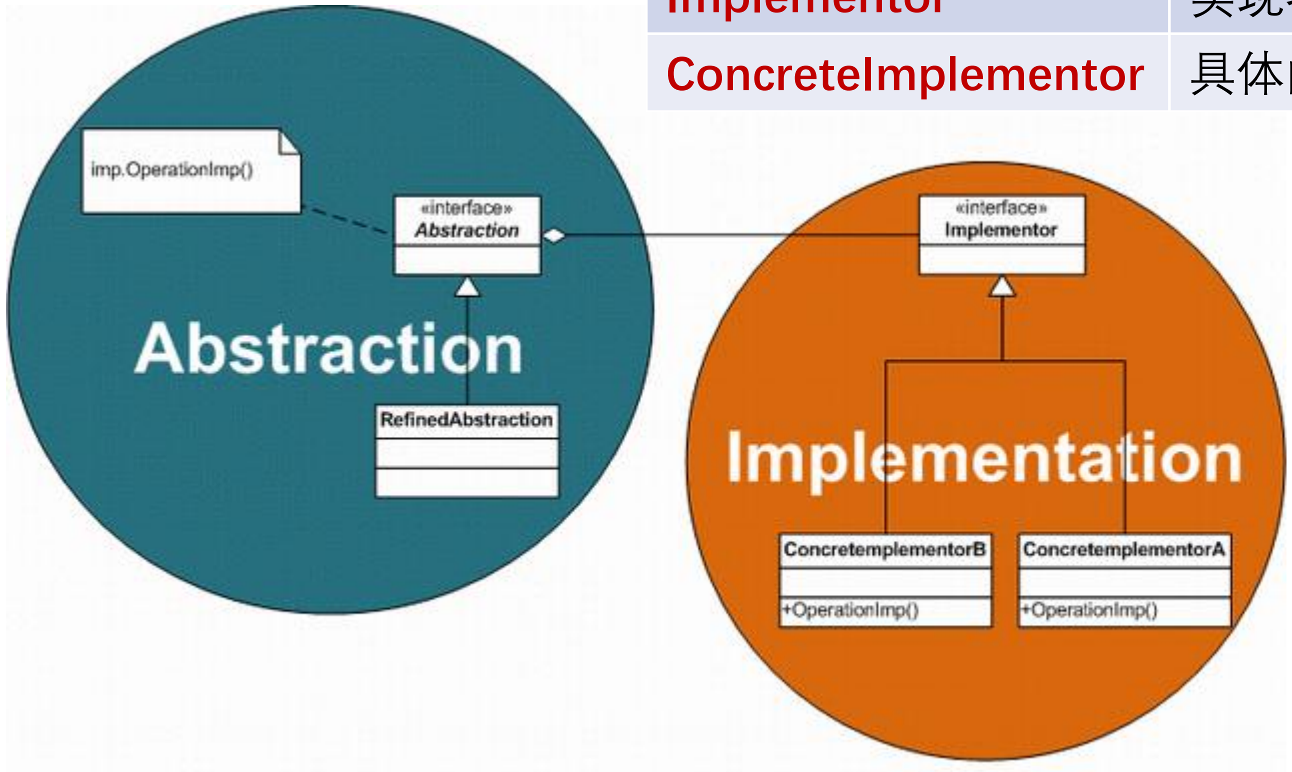

"Has A"
Burger
Side
HamBurger
ChickenBurger
Veggie Burger
Fries
Onion Rings

桥接器Bridge定义

- 将抽象和实现解耦，让两边能独立变化
- 适用
  - 跨平台支持
  - 插件plugin
  - 驱动程序driver

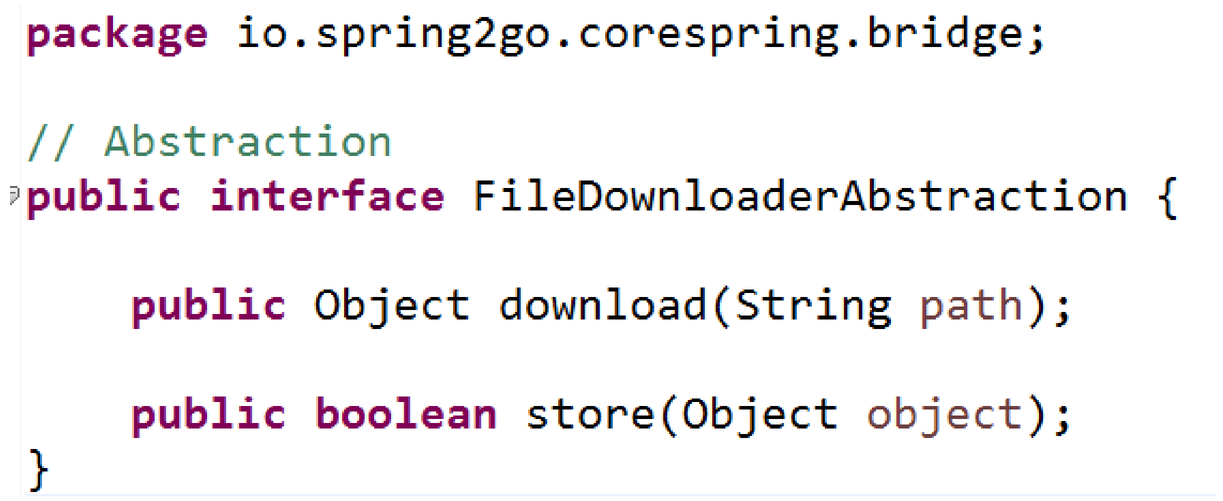
关系图

角色	职责
Abstraction	定义高层操作，使用Implementor提供的原子操作。提供给客户端使用的抽象，引用Implementor。客户使用该抽象，不需要知道实现者细节。
RefinedAbstraction	该类实现Abstraction定义的功能，将实现细节代理到ConcreteImplementor。
Implementor	实现者接口，一般提供原子primitive操作
ConcreteImplementor	具体的实现者

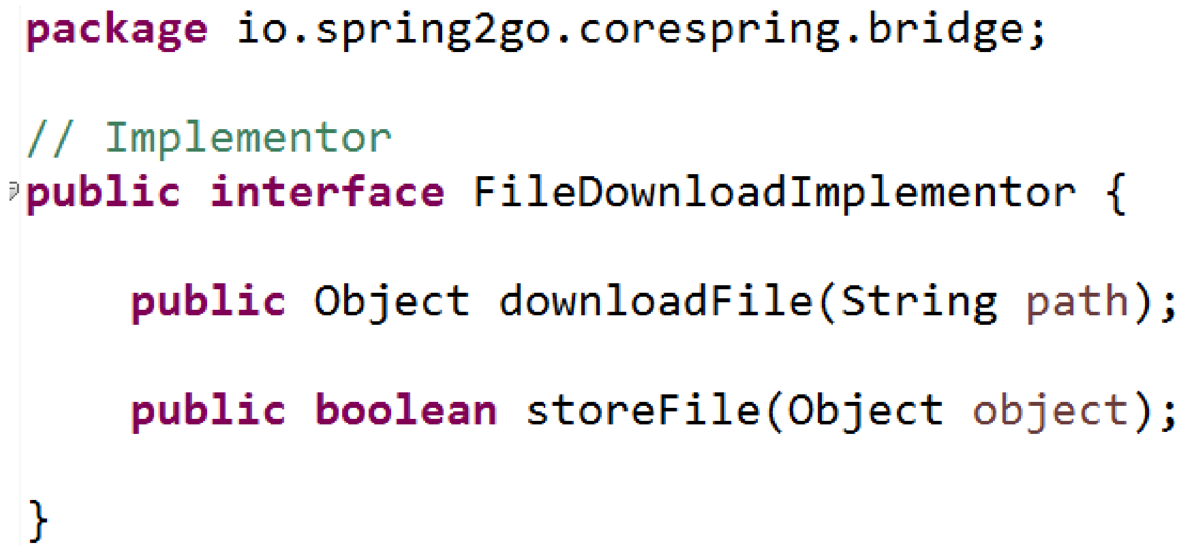


Bridge Pattern - Generic Structure

代码~Abstraction和Implementor者接口

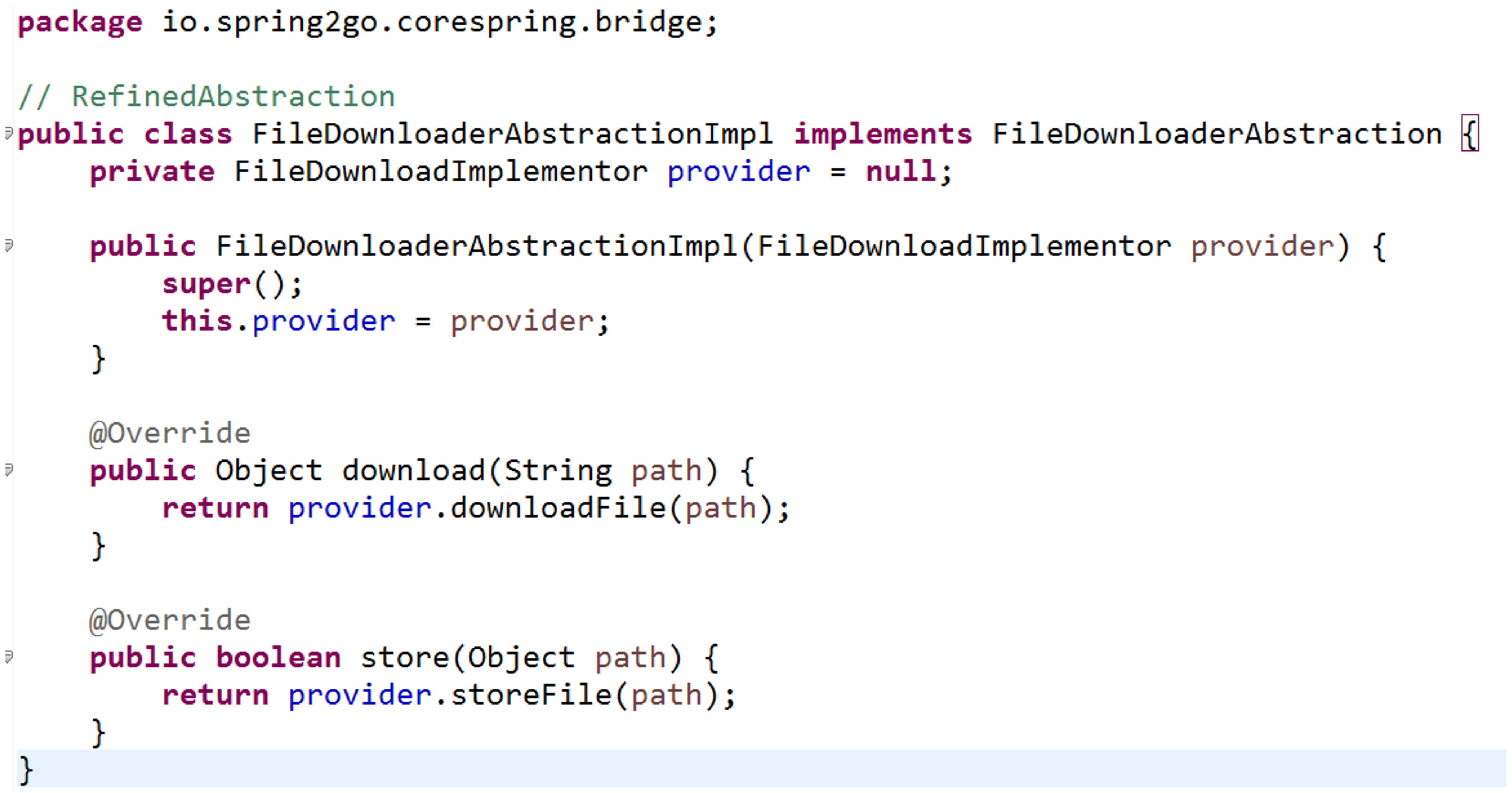


```java
package io.spring2go.corespring.bridge;

// Abstraction
public interface FileDownloaderAbstraction {

    public Object download(String path);

    public boolean store(Object object);
}
```



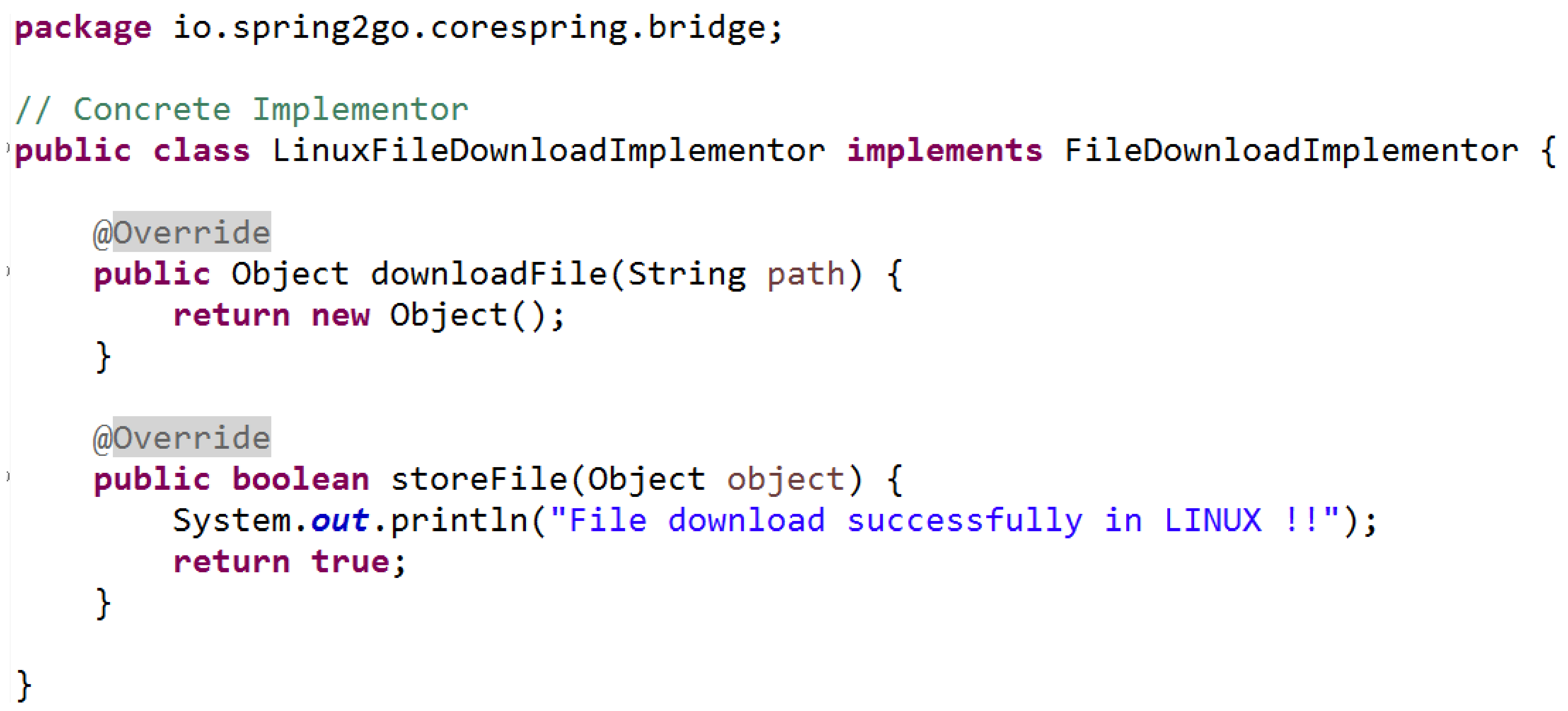
```java
package io.spring2go.corespring.bridge;

// Implementor
public interface FileDownloadImplementor {

    public Object downloadFile(String path);

    public boolean storeFile(Object object);

}
```

代码~RefinedAbstraction实现



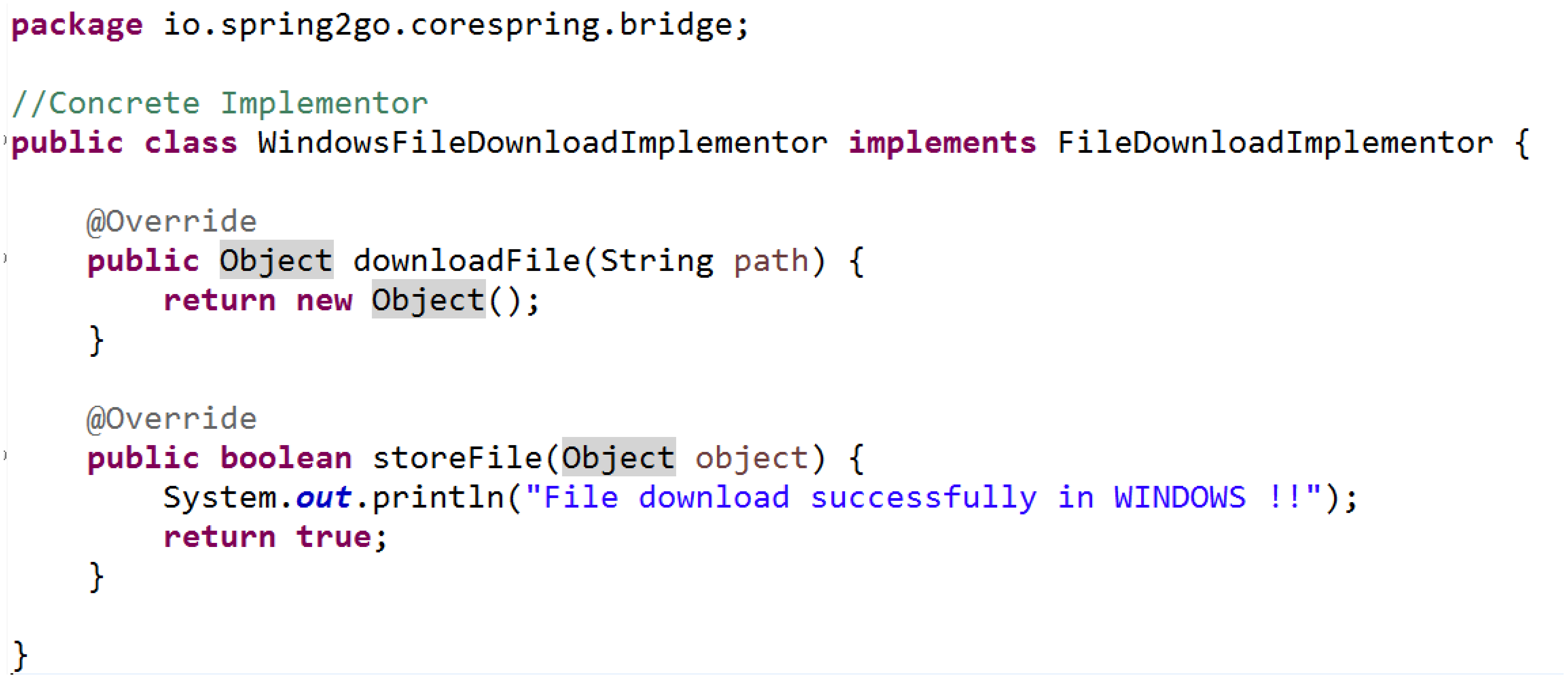
```java
package io.spring2go.corespring.bridge;

// RefinedAbstraction
public class FileDownloaderAbstractionImpl implements FileDownloaderAbstraction {
    private FileDownloadImplementor provider = null;

    public FileDownloaderAbstractionImpl(FileDownloadImplementor provider) {
        super();
        this.provider = provider;
    }

    @Override
    public Object download(String path) {
        return provider.downloadFile(path);
    }

    @Override
    public boolean store(Object path) {
        return provider.storeFile(path);
    }
}
```

代码~具体实现Linux



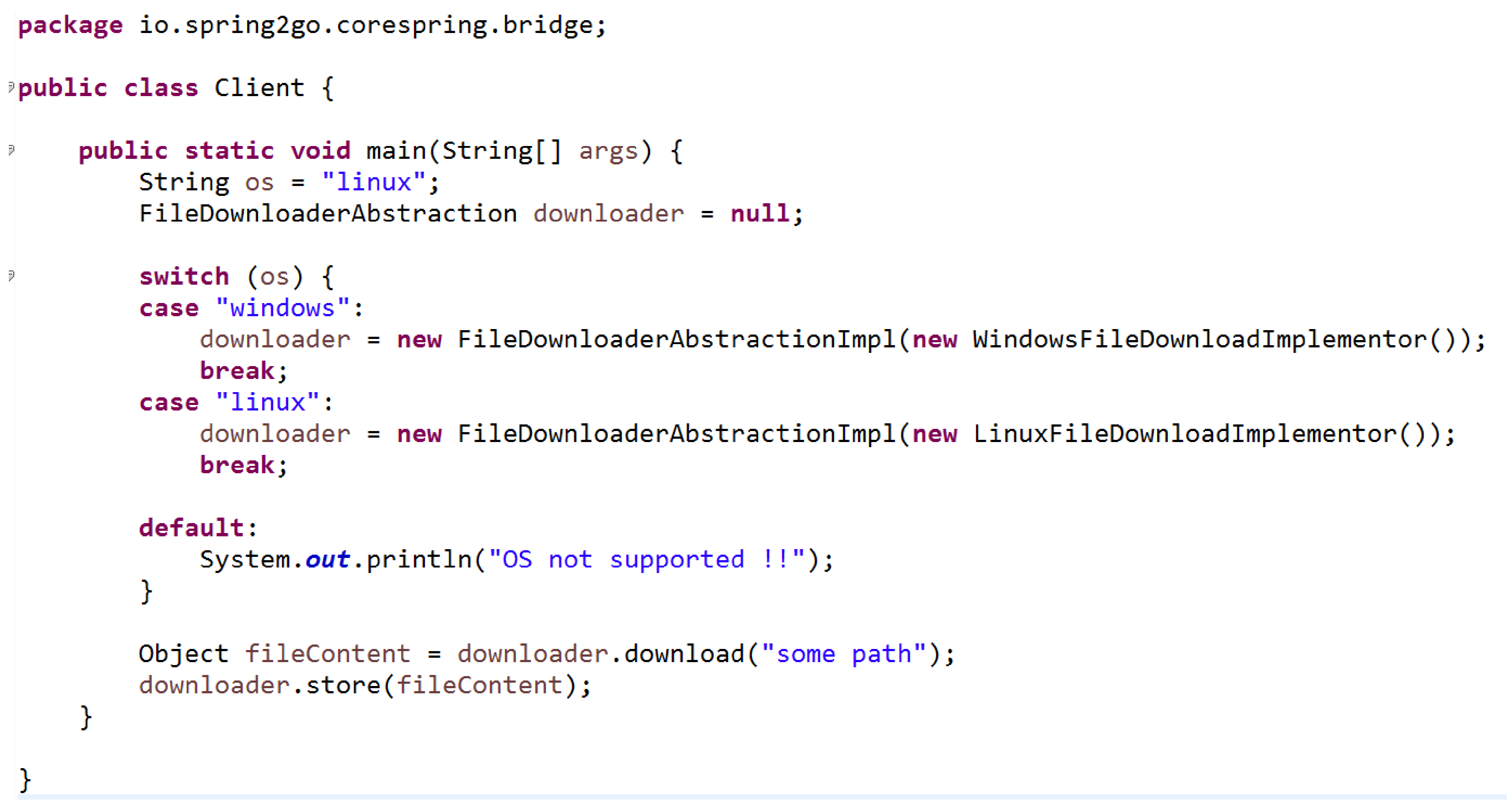
```java
package io.spring2go.corespring.bridge;

// Concrete Implementor
public class LinuxFileDownloadImplementor implements FileDownloadImplementor {

    @Override
    public Object downloadFile(String path) {
        return new Object();
    }

    @Override
    public boolean storeFile(Object object) {
        System.out.println("File download successfully in LINUX !!");
        return true;
    }

}
```

代码~具体实现Windows



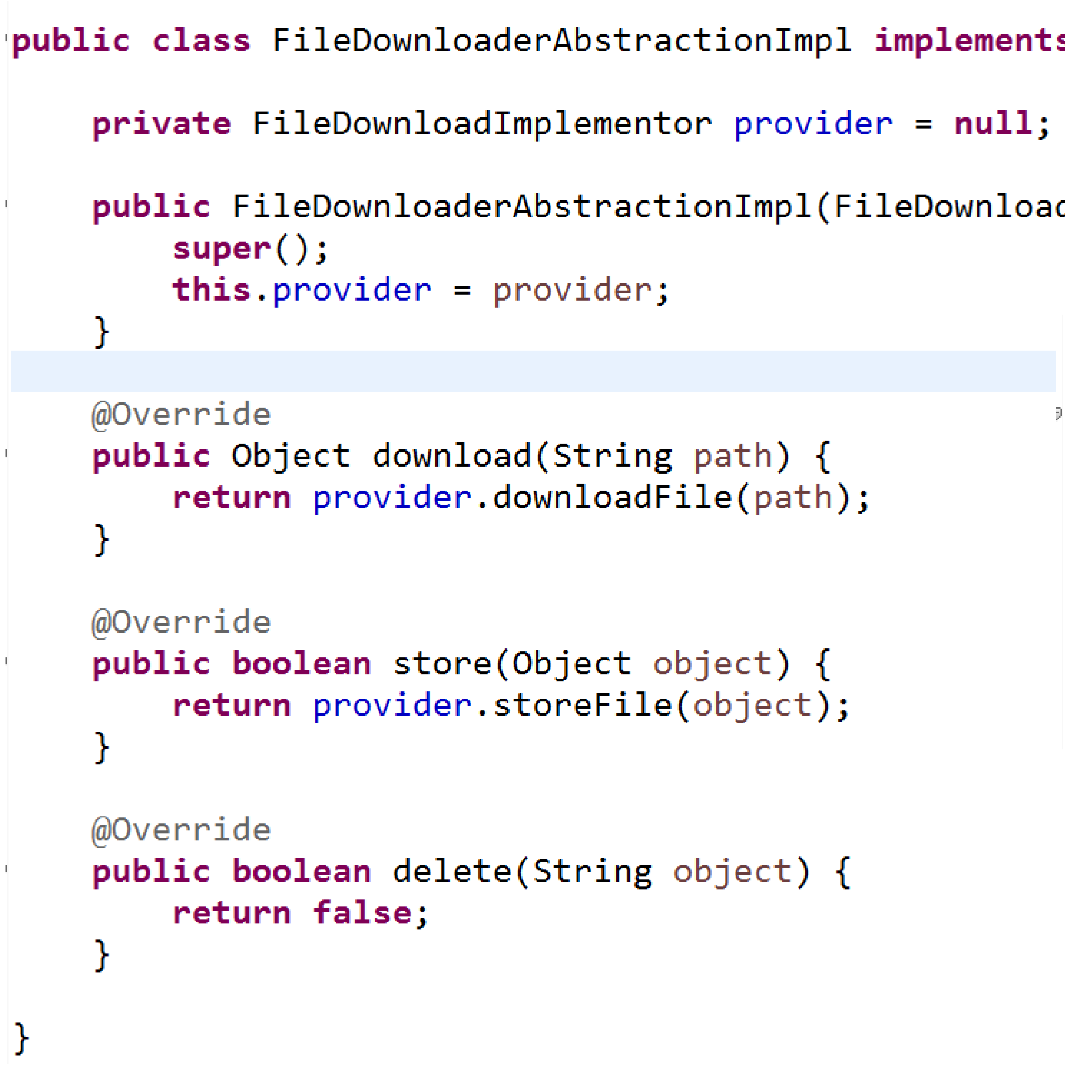
```java
package io.spring2go.corespring.bridge;

//Concrete Implementor
public class WindowsFileDownloadImplementor implements FileDownloadImplementor {

    @Override
    public Object downloadFile(String path) {
        return new Object();
    }

    @Override
    public boolean storeFile(Object object) {
        System.out.println("File download successfully in WINDOWS !!");
        return true;
    }

}
```

代码~客户端

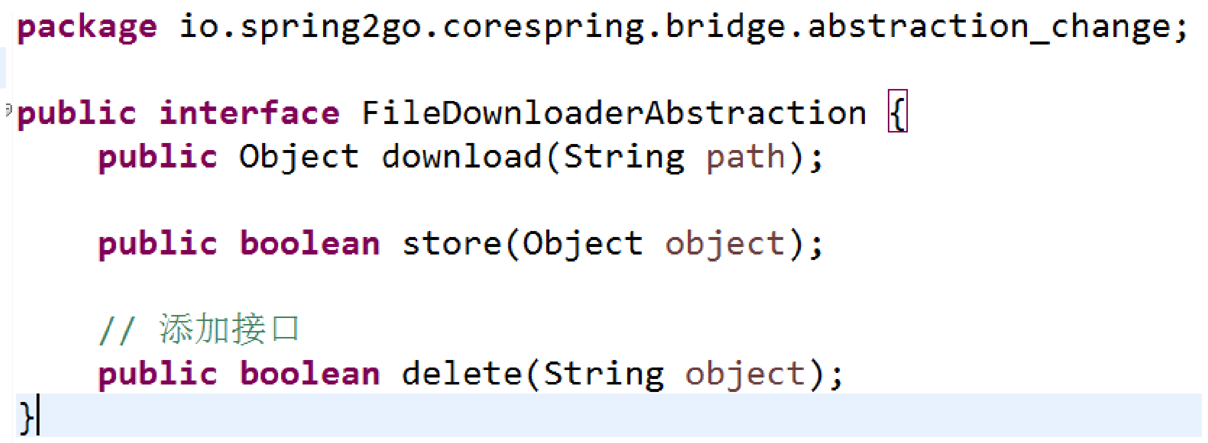


```java
package io.spring2go.corespring.bridge;

public class Client {

    public static void main(String[] args) {
        String os = "linux";
        FileDownloaderAbstraction downloader = null;

        switch (os) {
        case "windows":
            downloader = new FileDownloaderAbstractionImpl(new WindowsFileDownloadImplementor());
            break;
        case "linux":
            downloader = new FileDownloaderAbstractionImpl(new LinuxFileDownloadImplementor());
            break;

        default:
            System.out.println("OS not supported !!");
        }

        Object fileContent = downloader.download("some path");
        downloader.store(fileContent);
    }

}
```

代码~抽象变不影响实现

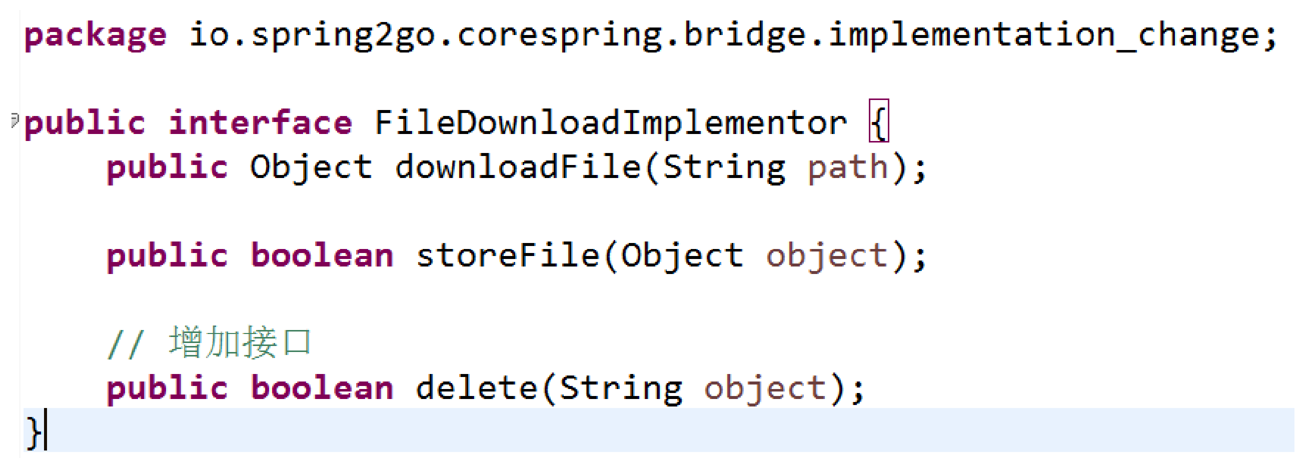


```java
public class FileDownloaderAbstractionImpl implements FileDownloaderAbstraction {

    private FileDownloadImplementor provider = null;

    public FileDownloaderAbstractionImpl(FileDownloadImplementor provider) {
        super();
        this.provider = provider;
    }

    @Override
    public Object download(String path) {
        return provider.downloadFile(path);
    }

    @Override
    public boolean store(Object object) {
        return provider.storeFile(object);
    }

    @Override
    public boolean delete(String object) {
        return false;
    }

}
```

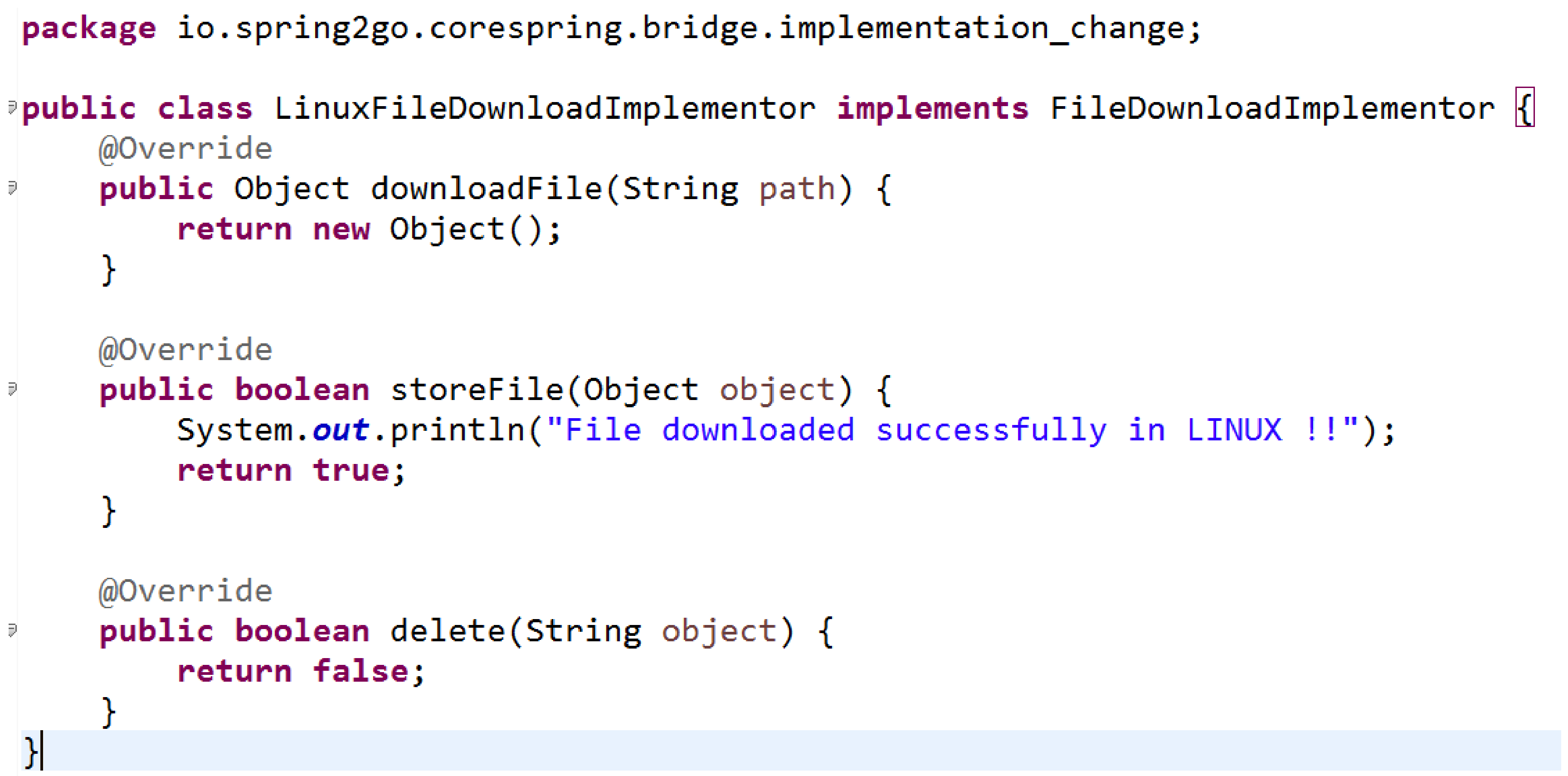


```java
package io.spring2go.corespring.bridge.abstraction_change;

public interface FileDownloaderAbstraction {
    public Object download(String path);

    public boolean store(Object object);

    // 添加接口
    public boolean delete(String object);
}
```

代码~实现变不影响抽象



```java
package io.spring2go.corespring.bridge.implementation_change;

public interface FileDownloadImplementor {
    public Object downloadFile(String path);

    public boolean storeFile(Object object);

    // 增加接口
    public boolean delete(String object);
}
```



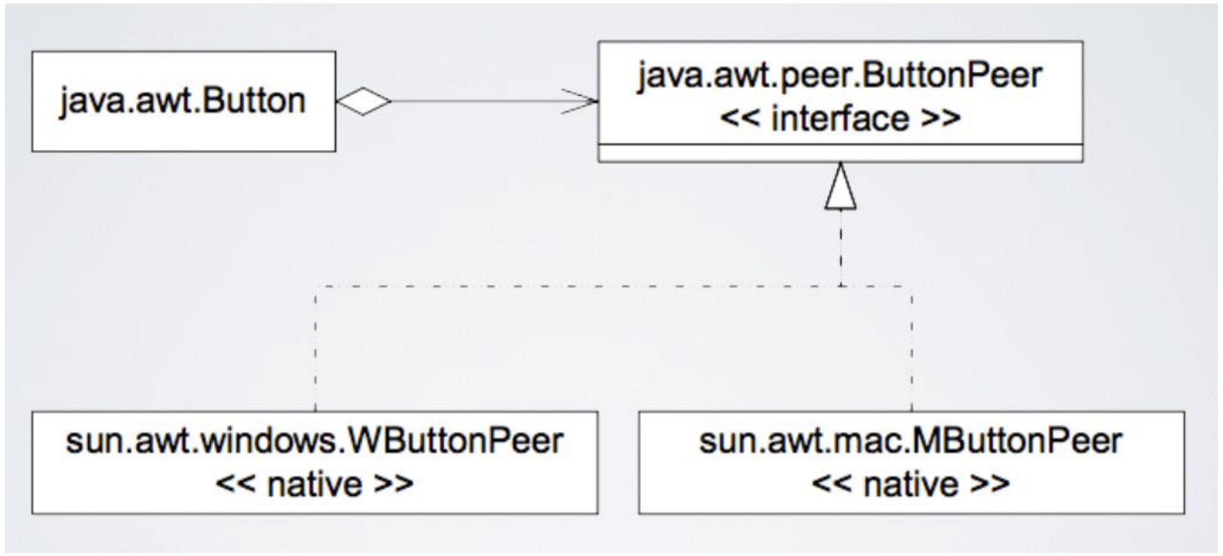
```java
package io.spring2go.corespring.bridge.implementation_change;

public class LinuxFileDownloadImplementor implements FileDownloadImplementor {
    @Override
    public Object downloadFile(String path) {
        return new Object();
    }

    @Override
    public boolean storeFile(Object object) {
        System.out.println("File downloaded successfully in LINUX !!");
        return true;
    }

    @Override
    public boolean delete(String object) {
        return false;
    }
}
```

好处

- 抽象和实现解耦
  - 编译或运行时绑定
- 减少子类数量
- 代码简洁
- 接口和实现可以独立变化
- 易于扩展
- 客户端代码低耦合
  - 依赖于抽象，而非具体实现

应用

- Logging
  - http://spring.io/blog/2009/12/04/logging-dependencies-in-spring/
- Java AWT(Abstract Window Toolkit)



参考

- Bridge Design Pattern
  - https://howtodoinjava.com/design-patterns/structural/bridge-design-pattern/
- Bridge Pattern – Bridging the gap between Interface and Implementation
  - https://www.codeproject.com/Articles/890/Bridge-Pattern-Bridging-the-gap-between-Interface

问题

- 桥接模式，适配器模式和策略模式的结构类似，他们之间的差异？

代码

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

波波微课
spring2go.com

波波微课

spring2go.com