命令模式Command

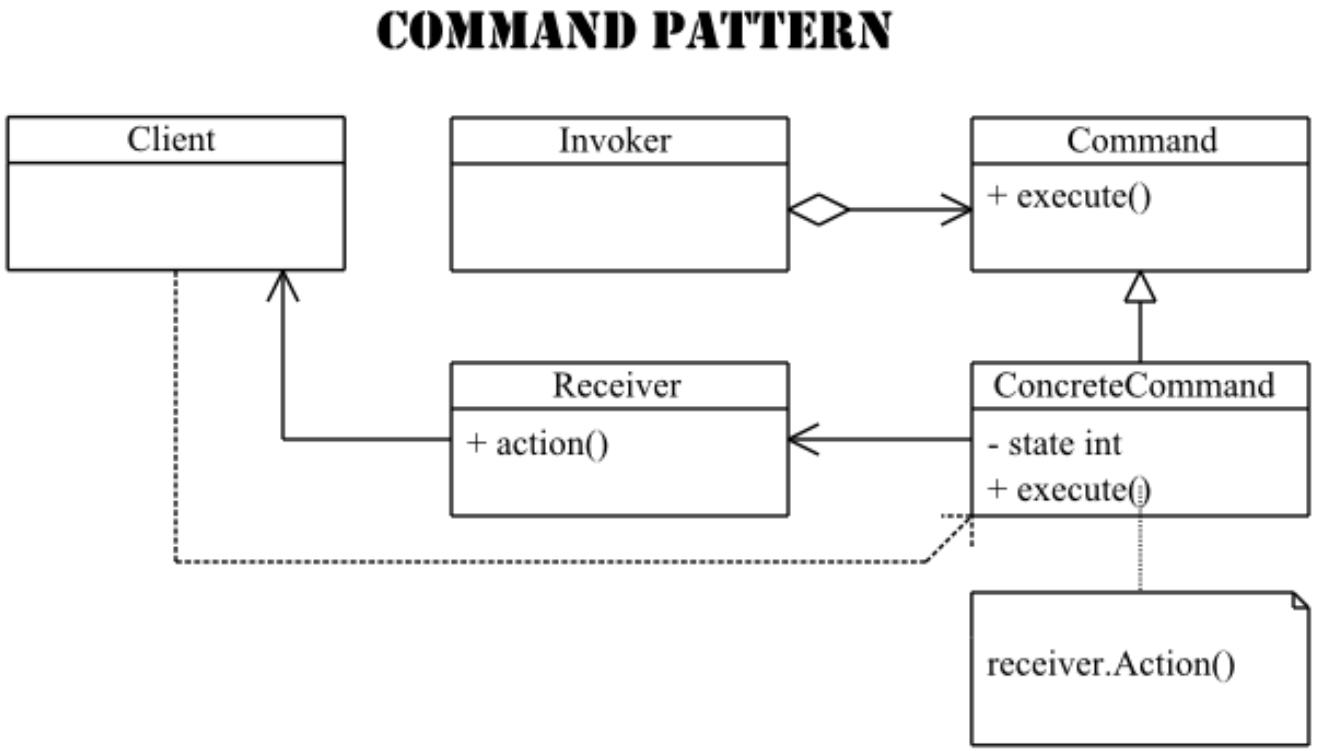
波波老师~研发总监/资深架构师

定义~百度百科

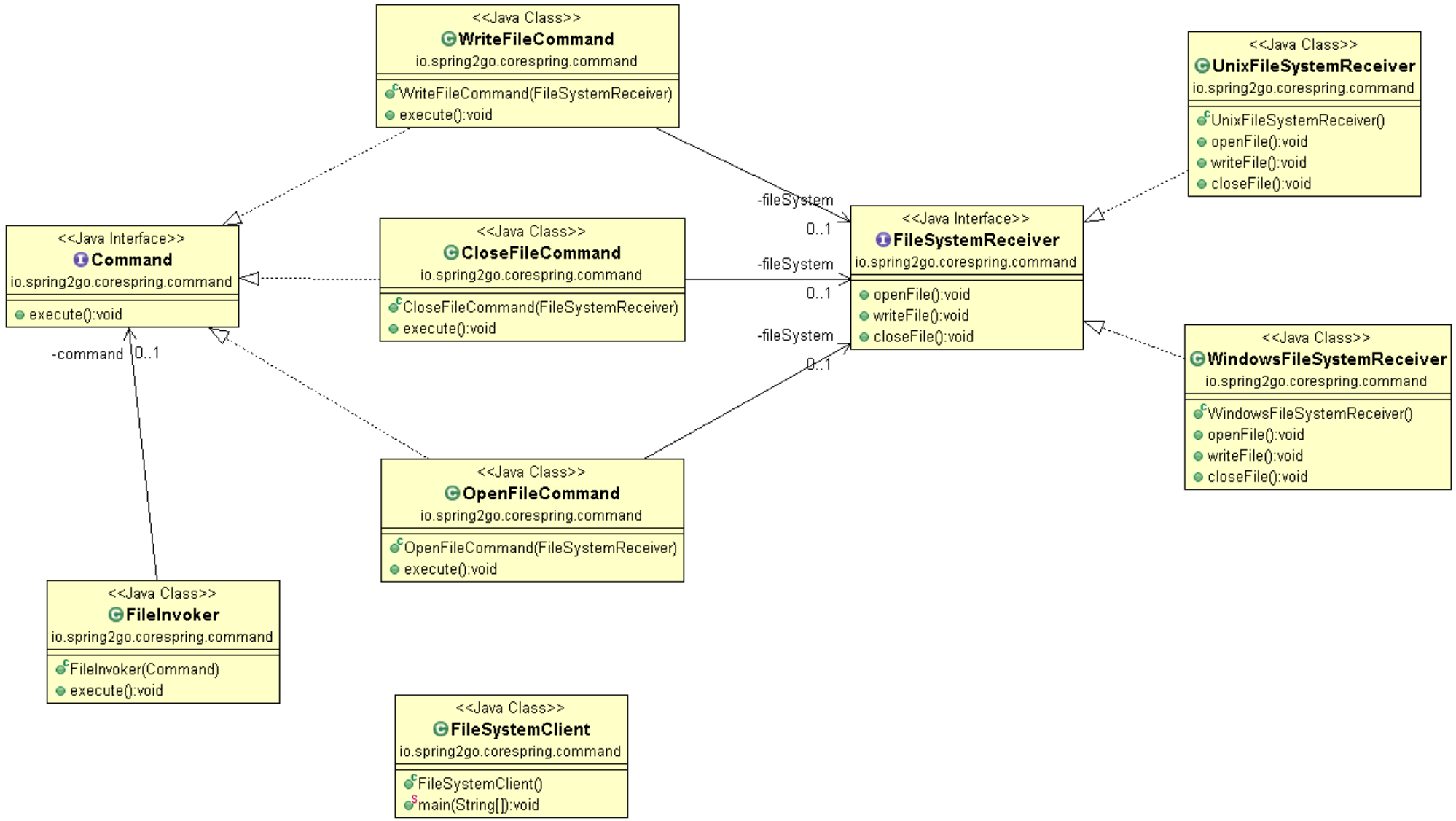
- 在软件系统中，“行为请求者”和“行为实现者”通常呈现一种紧耦合。但在某些场合，比如要对行为进行记录、撤销重做、事务等处理，这种无法抵御变化的紧耦合是不合适的。在这种情况下，如何将行为请求者与行为实现者解耦？
- 将一组行为抽象为对象，实现二者之间的松耦合。这就是命令模式（Command Pattern）

关系图

角色	职责
Command	用于执行一个行为的接口
ConcreteCommand	将行为接收执行者Action Receiver/Taker和行为调用者Action Invoker进行绑定的对象； 该对象负责执行Receiver上的操作。
Client	创建ConcreteCommand对象并设置其Receiver
Invoker	使用Command对象执行请求
Receiver	包含实际的行为操作逻辑



案例~文件操作


<<Java Class>>
WriteFileCommand
io.spring2go.corespring.command
WriteFileCommand(FileSystemReceiver)
execute():void
<<Java Class>>
UnixFileSystemReceiver
io.spring2go.corespring.command
UnixFileSystemReceiver()
openFile():void
writeFile():void
closeFile():void
-fileSystem
0..1
<<Java Interface>>
Command
io.spring2go.corespring.command
execute():void
<<Java Class>>
CloseFileCommand
io.spring2go.corespring.command
CloseFileCommand(FileSystemReceiver)
execute():void
-fileSystem
0..1
<<Java Interface>>
FileSystemReceiver
io.spring2go.corespring.command
openFile():void
writeFile():void
closeFile():void
-fileSystem
0..1
-command
0..1
<<Java Class>>
WindowsFileSystemReceiver
io.spring2go.corespring.command
WindowsFileSystemReceiver()
openFile():void
writeFile():void
closeFile():void
<<Java Class>>
OpenFileCommand
io.spring2go.corespring.command
OpenFileCommand(FileSystemReceiver)
execute():void
<<Java Class>>
FileInvoker
io.spring2go.corespring.command
FileInvoker(Command)
execute():void
<<Java Class>>
FileSystemClient
io.spring2go.corespring.command
FileSystemClient()
main(String[]):void

代码~Receiver接口

```java
// Receiver Interface
public interface FileSystemReceiver {
    void openFile();
    void writeFile();
    void closeFile();
}
```

代码~UnixFileSytemReceiver

```java
// Receiver
public class UnixFileSystemReceiver implements FileSystemReceiver {

    @Override
    public void openFile() {
        System.out.println("Opening file in unix OS");
    }

    @Override
    public void writeFile() {
        System.out.println("Writing file in unix OS");
    }

    @Override
    public void closeFile() {
        System.out.println("Closing file in unix OS");
    }

}
```

代码~WindowsFileSystemReceiver

```java
// Receiver
public class WindowsFileSystemReceiver implements FileSystemReceiver {

    @Override
    public void openFile() {
        System.out.println("Opening file in Windows OS");
    }

    @Override
    public void writeFile() {
        System.out.println("Writing file in Windows OS");
    }

    @Override
    public void closeFile() {
        System.out.println("Closing file in Windows OS");
    }

}
```

代码~Command接口

```java
// Command interface
public interface Command {

    void execute();

}
```

代码~OpenFileCommand

```
// Concrete Command
public class OpenFileCommand implements Command {

    private FileSystemReceiver fileSystem;

    public OpenFileCommand(FileSystemReceiver fs) {
        this.fileSystem = fs;
    }

    @Override
    public void execute() {
        // open command is forwarding request to openFile method
        this.fileSystem.openFile();
    }

}
```

代码~WriteFileCommand

```java
// Concrete Command
public class WriteFileCommand implements Command {

    private FileSystemReceiver fileSystem;

    public WriteFileCommand(FileSystemReceiver fs) {
        this.fileSystem = fs;
    }

    @Override
    public void execute() {
        this.fileSystem.writeFile();
    }

}
```

代码~CloseFileCommand

```java
// Concrete Command
public class CloseFileCommand implements Command {

    private FileSystemReceiver fileSystem;

    public CloseFileCommand(FileSystemReceiver fs) {
        this.fileSystem = fs;
    }

    @Override
    public void execute() {
        this.fileSystem.closeFile();
    }

}
```

代码~Invoker

```
// Invoker
public class FileInvoker {

    private Command command;

    public FileInvoker(Command c) {
        this.command = c;
    }

    public void execute() {
        this.command.execute();
    }
}
```

代码~FileSystemReceiverUtil

```java
public class FileSystemReceiverUtil {

    public static FileSystemReceiver getUnderlyingFileSystem() {
        String osName = System.getProperty("os.name");
        System.out.println("Underlying OS is : " + osName);
        if (osName.contains("Windows")) {
            return new WindowsFileSystemReceiver();
        } else {
            return new UnixFileSystemReceiver();
        }
    }
}
```

代码~客户端

```
// Client
public class FileSystemClient {

    public static void main(String[] args) {
        // creating the receiver object
        FileSystemReceiver fs = FileSystemReceiverUtil.getUnderlyingFileSystem();

        // creating command and associating with receiver
        OpenFileCommand openFileCommand = new OpenFileCommand(fs);

        // creating invoker and associating with Command
        FileInvoker file = new FileInvoker(openFileCommand);

        // perform action on invoker object
        file.execute();

        WriteFileCommand writeFileCommand = new WriteFileCommand(fs);
        file = new FileInvoker(writeFileCommand);
        file.execute();

        CloseFileCommand closeFileCommand = new CloseFileCommand(fs);
        file = new FileInvoker(closeFileCommand);
        file.execute();

    }

}
```

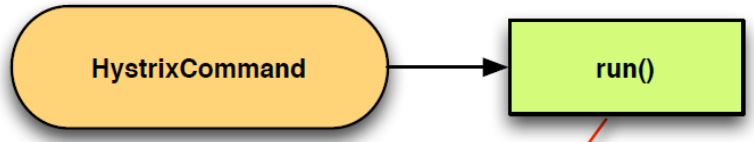
```
Underlying OS is : Windows 7
Opening file in Windows OS
Writing file in Windows OS
Closing file in Windows OS
```

优劣

- 优点
  - 调用者和接收者之间通过Command解耦
  - 增加新Command不需要改现有代码，易于扩展
- 不足
  - 可能会搞出一堆Command

应用

- java.lang.Runable
- javax.swing.Action
- Structs Action
- Netflix Hystrix Command




```
public class CommandHelloWorld extends HystrixCommand<String> {

    ...

    protected String run() {
        return "Hello " + name + "!";
    }
}
```

run() invokes "client" Logic

课后练习

- 命令模式和职责链模式的差异？

参考

- Command Design Pattern
  - https://www.journaldev.com/1624/command-design-pattern
- Command Design Pattern
  - https://howtodoinjava.com/design-patterns/behavioral/command-pattern/

代码

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


波波微课
spring2go.com
春天小课
spring2go.com


C:\