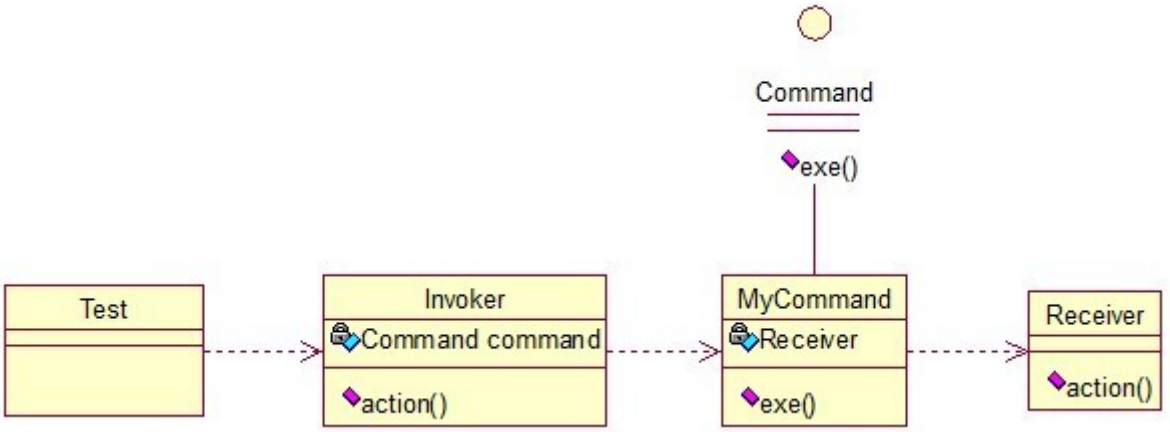
## 命令模式（Command）

命令模式很好理解，举个例子，司令员下令让士兵去干件事情，从整个事情的角度来考虑，司令员的作用是，发出口令，口令经过传递，传到了士兵耳朵里，士兵去执行。这个过程好在，三者相互解耦，任何一方都不用去依赖其他人，只需要做好自己的事儿就行，司令员要的是结果，不会去关注到底士兵是怎么实现的。我们看看关系图：




Invoker是调用者（司令员），Receiver是被调用者（士兵），MyCommand是命令，实现了Command接口，持有接收对象，看实现代码：

[java] view plaincopy

```
1. public interface Command {
2.     public void exe();
3. }
```

[java] view plaincopy

```
1. public class MyCommand implements Command {
2.
3.     private Receiver receiver;
4.
5.     public MyCommand(Receiver receiver) {
6.         this.receiver = receiver;
7.     }
8.
9.     @Override
10.     public void exe() {
11.         receiver.action();
12.     }
13. }
```

[java] view plaincopy

```
1. public class Receiver {
2.     public void action(){
```

```
3.          System.out.println("command received!");
4.      }
5.  }
```

[java] view plaincopy

```
1.  public class Invoker {
2.  
3.      private Command command;
4.  
5.      public Invoker(Command command) {
6.          this.command = command;
7.      }
8.  
9.      public void action(){
10.         command.exe();
11.     }
12. }
```

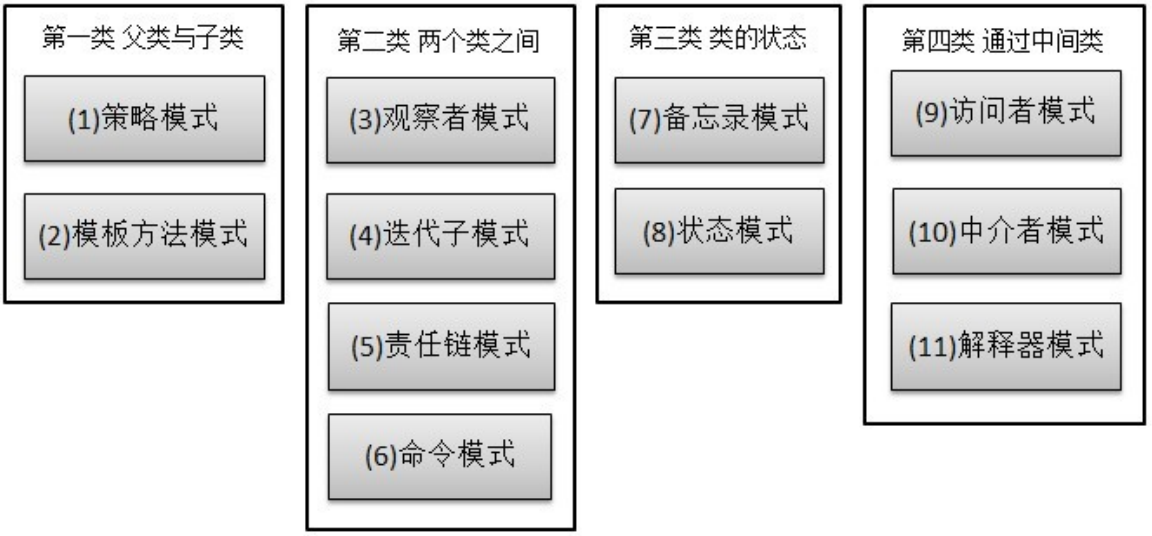
[java] view plaincopy

```
1.  public class Test {
2.  
3.      public static void main(String[] args) {
4.          Receiver receiver = new Receiver();
5.          Command cmd = new MyCommand(receiver);
6.          Invoker invoker = new Invoker(cmd);
7.          invoker.action();
8.      }
9.  }
```

输出：command received!

这个很哈理解，命令模式的目的就是达到命令的发出者和执行者之间解耦，实现请求和执行分开，熟悉Struts的同学应该知道，Struts其实就是一种将请求和呈现分离的技术，其中必然涉及命令模式的思想！

其实每个设计模式都是很重要的一种思想，看上去很熟，其实是因为我们在学到的东西中都有涉及，尽管有时我们并不知道，其实在Java本身的设计之中处处都有体现，像AWT、JDBC、集合类、IO管道或者是Web框架，里面设计模式无处不在。因为我们篇幅有限，很难讲每一个设计模式都讲的很详细，不过我会尽我所能，尽量在有限的空间和篇幅内，把意思写清楚了，更好让大家明白。本章不出意外的话，应该是设计模式最后一讲了，首先还是上一下上篇开头的那个图：


第一类 父类与子类
(1)策略模式
(2)模板方法模式
第二类 两个类之间
(3)观察者模式
(4)迭代子模式
(5)责任链模式
(6)命令模式
第三类 类的状态
(7)备忘录模式
(8)状态模式
第四类 通过中间类
(9)访问者模式
(10)中介者模式
(11)解释器模式