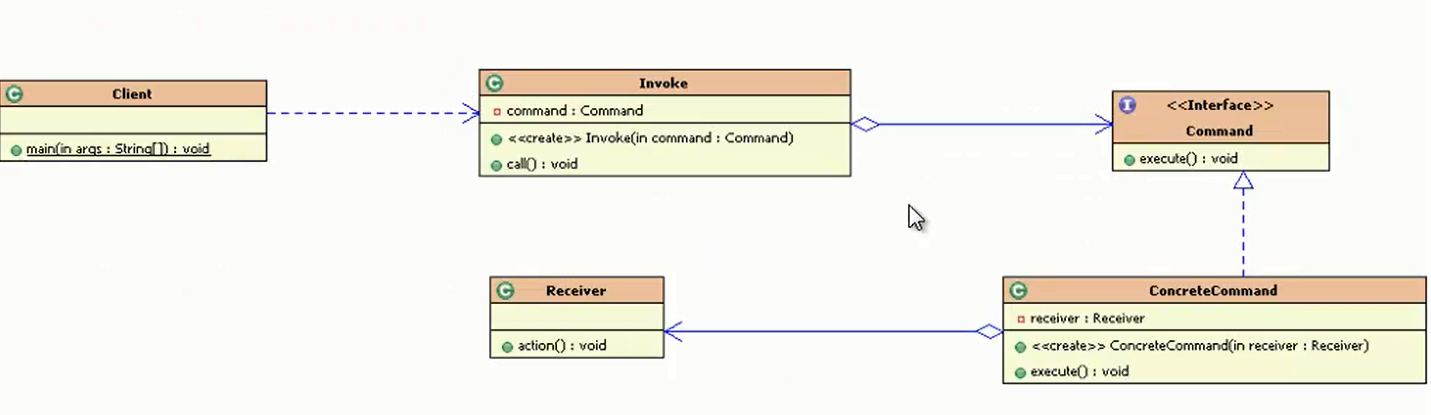
**定义：**  
将一个请求封装成对象，使得请求发送者和请求接受者之间相互隔离，消除两者之间的耦合。引入命令类，使得不同请求对客户参数化，并且可以对命令添加附件操作，**如：排队、撤销、日志、组合等。**

**结构图：**  


* Command：抽象命令类，一般是一个抽象类或者接口。在其中定义了命令的执行execute()、undo()等操作。
* ConcreteCommand：具体命令类，实现了抽象命令类的方法，并且持有命令接受者的引用。具体执行方法时，调用接受者的相关操作。（可以有不同的命令，对应不同的接受者进行操作。用一个list去装各种Command，再对应不同的Reciver即可）
* Invoker(老大，专门调用Command)：请求发送者，通过命令对象执行请求。针对抽象命令类编程，并不需要关心对应的命令如何执行，被谁执行。与抽象命令的关联关系也不固定，可以通过注入或者构造的方法或者采用XML和反射的方法来进行关联。
* Reciver：**请求接受者，执行请求的相关操作，对请求进行详细的业务处理。**

**下面演示单个命令与多个命令：**

**单个：**

**package** command;  
  
**public class** Receiver {  
 **public void** action() {  
 System.***out***.println(**"执行命令"**);  
 }  
}

**package** command;  
  
**public interface** Command {  
 **void** execute();  
}

**package** command;  
  
**public class** ConcreteCommand **implements** Command {  
  
 **private** Receiver **receiver**;  
  
 **public** ConcreteCommand(Receiver receiver) {  
 **this**.**receiver** = receiver;  
 }  
  
 @Override  
 **public void** execute() {  
 **receiver**.action();  
 }  
}

**package** command;  
  
**public class** Invoke {  
  
 **private** Command **command**;  
  
 **public** Invoke(Command command) {  
 **this**.**command** = command;  
 }  
  
 **public void** call() {  
 **command**.execute();  
 }  
}

**package** command;  
  
**public class** test {  
 **public static void** main(String[] args) {  
 Invoke invoke = **new** Invoke(**new** ConcreteCommand( **new** Receiver()));  
 invoke.call();  
 }  
}

多个：

**package** command;  
  
**public class** Receiver {  
 **public void** action() {  
 System.***out***.println(**"执行命令"**);  
 }  
}

**package** command;  
  
**public interface** Command {  
 **void** execute();  
}

多个中间者：

**package** command.ConcreteList;  
  
**import** command.Command;  
**import** command.Receiver;  
  
**public class** Politician **implements** Command {  
  
 **private** Receiver **receiver**;  
  
 **public** Politician() {  
 }  
  
 **public** Politician(Receiver receiver) {  
 **this**.**receiver** = receiver;  
 }  
  
 @Override  
 **public void** execute() {  
 System.***out***.print(**"Politician去"**);  
 **receiver**.action();  
 }  
}

**package** command.ConcreteList;  
  
**import** command.Command;  
**import** command.Receiver;  
  
**public class** Engineer **implements** Command {  
  
 **private** Receiver **receiver**;  
  
 **public** Engineer() {  
 }  
  
 **public** Engineer(Receiver receiver) {  
 **this**.**receiver** = receiver;  
 }  
  
 @Override  
 **public void** execute() {  
 System.***out***.print(**"Engineer去"**);  
 **receiver**.action();  
 }  
}

**package** command.ConcreteList;  
  
**import** command.Command;  
**import** command.Receiver;  
  
**public class** Programmer **implements** Command {  
  
 **private** Receiver **receiver**;  
  
 **public** Programmer() {  
 }  
  
 **public** Programmer(Receiver receiver) {  
 **this**.**receiver** = receiver;  
 }  
  
 @Override  
 **public void** execute() {  
 System.***out***.print(**"Programmer去"**);  
 **receiver**.action();  
 }  
}

**package** command;  
  
**import** command.ConcreteList.Engineer;  
**import** command.ConcreteList.Politician;  
**import** command.ConcreteList.Programmer;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** Invoke {  
  
 **private** Command **command**;  
  
 **private** List<Command> **commandList**;  
  
 **public** Invoke(Command command) {  
 **this**.**command** = command;  
 }  
  
 **public** Invoke(List<Command> commandList) {  
 **this**.**commandList** = commandList;  
 }  
  
 **public void** call( ) {  
 **command**.execute();  
 }  
  
  
 **public void** callList() {  
 **for** (Command command : **commandList**) {  
 command.execute();  
 }  
 }  
  
 *//对于List Command,把所有的加到List里面去* **public** Invoke() {  
  
 }  
}

**package** command;  
  
**import** command.ConcreteList.Engineer;  
**import** command.ConcreteList.Politician;  
**import** command.ConcreteList.Programmer;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** test {  
 **public static void** main(String[] args) {  
*// Invoke invoke = new Invoke(new ConcreteCommand( new Receiver()));  
// invoke.call();* Receiver receiver = **new** Receiver();  
 List<Command> commandList = **new** ArrayList<>();  
 commandList.add(**new** Engineer(receiver));  
 commandList.add(**new** Politician(receiver));  
 commandList.add(**new** Programmer(receiver));  
 Invoke invokeList = **new** Invoke(commandList);  
 invokeList.callList();  
  
 }  
}

输出：

Engineer去执行命令

Politician去执行命令

Programmer去执行命令