3.1 命令模式

3.1.1 实现描述

1. 命令模式包含以下主要角色：

抽象命令类角色：定义命令的接口，声明执行的方法。

具体命令角色：具体的命令，实现命令接口；通常会持有接收者，并调用接收者的功能来完成命令要执行的操作。

实现者角色：接收者，真正执行命令的对象。任何类都可能成为一个接收者，只要它能够实现命令要求实现的相应功能。

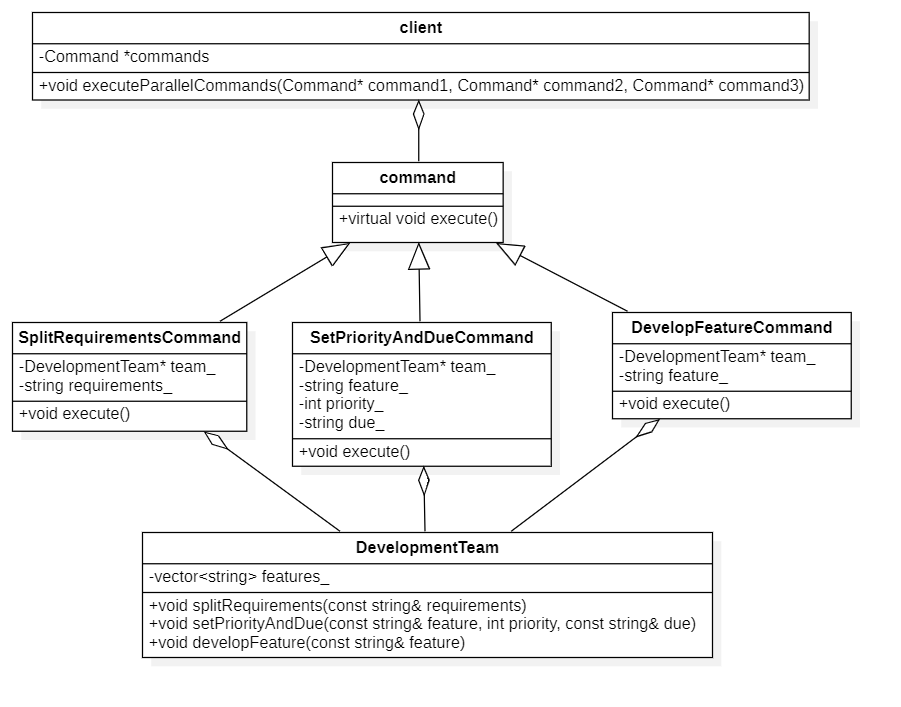
调用者角色：要求命令对象执行请求，通常会持有命令对象，可以持有很多的命令对象。这个是客户端真正触发命令并要求命令执行相应操作的地方，也就是说相当于使用命令对象的入口。

1. 在本项目中，考虑到汽车的研发团队需要市场调研专员提供客户的需求，并根据需求来进行开发。

模拟抽象客户提出需求，项目经理收集需求，研发团队收到需求并拆分为多个Feature，每个Feature设置Priority与Due，并综合排序来进行研发，不同的命令可以拆分为多个小目标完成，不同阶段的命令可以并行执行

分别设置抽象类Command，SetPriorityAndDueCommand类用于实现设置优先级命令DevelopFeatureCommand类用以实现研发特征命令，SplitRequirementsCommand类用以实现拆分需求命令，然后使用DevelopmentTeam类进行命令的具体实现，并对外提供client类作为接口供用户调用

3.1.2 类图等各种UML图，以及流程图



3.1.3 代价分析

降低系统的耦合度。命令模式能将调用操作的对象与实现该操作的对象解耦。

增加或删除命令非常方便。采用命令模式增加与删除命令不会影响其他类，它满足“开闭原则”，对扩展比较灵活。

可以实现宏命令。命令模式可以与组合模式结合，将多个命令装配成一个组合命令，即宏命令。

方便实现 Undo 和 Redo 操作。命令模式可以与后面介绍的备忘录模式结合，实现命令的撤销与恢复。

使用命令模式可能会导致某些系统有过多的具体命令类。

系统结构更加复杂。

3.1.4 其他内容（如果有的话）

总结为以下使用场景：

1. 系统需要将请求调用者和请求接收者解耦，使得调用者和接收者不直接交互。
2. 系统需要在不同的时间指定请求、将请求排队和执行请求。
3. 系统需要支持命令的撤销(Undo)操作和恢复(Redo)操作。