上一讲，我觉得就已经比较透彻的讲清楚，版本是怎么回事了853769620

所以你可以理解为，一次commit操作就是一个版本，因为这个commit对应了本次最新的所有的修改

之前所有的commit做出的修改，在这次commit并没有变化，所以说呢，本次commit包含的最新的所有的修改，加上之前的commimt做出的但是本次没变化的修改，结合起来，不就是最新的一次版本吗？

以此类推呢？

上一个commit对应的变化，加上它之前的所有commit做出的变化，结合起来，就是相当于是上一个commit对应的整个项目完整的一个版本

来做一个实验，版本控制

第一讲，首先git第一个功能，就是通过提交这个事情，保存下来每个文件完整的历史版本

然后呢，我们如果要做版本控制，比如说，发现这次这个版本不ok啊，希望回退到上一个版本，不要这次最新的这个版本的代码了，打算重新写

这个时候怎么办？

就涉及到了版本的控制

git reset --hard HEAD^，就可以回退到上一个版本

HEAD^，代表了什么？

HEAD -> master -> commit（add methods for classes）

HEAD^，代表的是commit（add methods for classes）的上一个commit（add two files）

git reset --hard HEAD^，就是将仓库、暂存区、工作区，全部恢复到上一个commit（add two files）对应的状态

git reset --hard HEAD~5，退回到HEAD之前的倒数第5个commit的状态

git reset --hard d324644，指定一个commit的hash值，回退到很老的版本

git reset这个命令，可以任意穿梭到历史的任何一个版本上去

如果我要回来呢？重新回到之前add methods for classes那个commit对应的版本

git reflog

有一个时光机，可以任意穿梭，回到过去，回到未来

实现了我们第一讲所说的，版本管理的困难，被git轻易的解决了

（1）记录下来了每一个版本

（2）可以让我们通过简单的命令，在任何一个版本中任意穿梭，实现了版本控制功能

git最核心的原理

（1）三个区域：工作区、暂存区、仓库

（2）提交历史：在仓库中，blob->tree->commit->master-HEAD，这套东西形成了git的数据结构