# 设计文档

## 功能需求：

1. 实现简易版本管理工具中分支、回滚、命令行交互功能
2. 分支管理要保存的信息：
   1. 保存分支信息
   2. 保存每个分支最新的commit\_id
   3. 当前处于哪个分支上

## 模块详细设计：

#### 分支管理结构：

1. 分支管理可形式化为如下树结构



1. 分支管理结构的具体实现：
   1. 定义一个CommitInfo类，包含commit对象的commit\_id、以及所在的分支
   2. 定义一个分支管理对象CommitManage，包含以下内容
      1. 定义一个head指针，用来指向分支上最新的结点
      2. 定义一个hashmap的属性结构commitPoint，用来存储具体的分支名（Key ：String name）与指向的分支对象（Value: Commit commit），如上图所示的testing与master及其指向对象
      3. 定义一个hashmap的属性结构commitNodePoint, 用来存储具体的commit 对象（Key ：Commit commit）与该对象的具体信息（Value: CommitInfo info）

#### 分支切换：

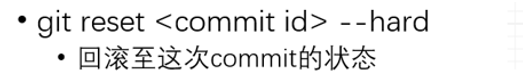
1. 借鉴如下图中git的命令信息：



1. 对于查看已有分支功能，定义一个travelCommit函数遍历CommitManage对象中的commitPoint属性，即可查看所有分支信息
2. 对于创建新的分支，定义一个createBranch（String batchName）函数，向CommitManage的commitPoint属性添加batchName作为key，value（未指定时）用当前CommitManage中head指针所指向的对象作为value
3. 对于切换到相应的分支，定义一个changeBranch（String batchName）函数，将CommitManage中commitPoint（batchName）的值赋给head指针，然会head指针所指向的commit对象

#### 回滚功能：

1. 借鉴如下图中git的命令信息



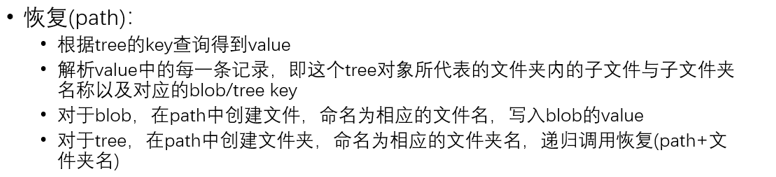
1. 定义一个rollbackCommit（String commit\_id）函数
   1. 遍历commitPoint中所有的key、value
   2. 对每次遍历得到的key、value
      1. If commitNodePoint（value）.commit\_id == commit\_id:

则head指向该key，返回head

* + 1. 不相等，如果该key.parent为null，则跳转到a），将当前结点置为key.paren，跳转到ii重复执行上面流程

#### commit的数据恢复：

1. 定义一个dataRecover()函数，具体思路按照下述图片的思路编写代码



#### 命令行交互功能：

1. 编写一个交互类testFunction(),在main函数中使用scanner接收控制台的命
   1. 启动后先在控制台输出相应的输入命令的格式规范，等待用户输入相应控制命令，进行交互
   2. 大体命令信息：针对分支切换、分支回滚功能：
      1. 用户输入： git branch，调用travelCommit（）
      2. 用户输入：git branch “XXXXX”,调用createBranch（）
      3. 用户输入：git checkout “XXXXX”,调用changeBranch（）
      4. 用户输入：git reset “XXXXX”,调用rollbackCommit（）
      5. 用户输入：git recocer data,调用dataRecover（）