【Commit设计文档】

//Commit实现：

/\*

#Week2任务有关Commit实现的功能要求

—给定一个工作区目录，生成对应的commit；

—新生成commit前，查找HEAD，

HEAD暂无，本次即为初次commit；生成commit并添加

HEAD存在，将根目录的tree key与HEAD指向的前次commit的tree key进行比较，

不同时（即文件发生了变动）生成commit并添加

—存储指向当前最新commit的HEAD指针，（原指向前次提交的commit；

#Commit文件内容

tree -本次提交的根目录tree key

parent -提交前的前次commit key

author -作者信息

committer -提交commit者信息

自定义备注（供用户自行输入类似示例中“This is the second commit."的句子）

#HEAD文件内容

当前指向的commit所对应的根目录tree key

\*/

/\*

#Commit 类设计

—数据域

private String key

private String value

private String presentcommit

……（若分拆则继续添加）

—构造器：参数是给定工作区目录；

value是前述的Commit文件内容，五个说明；

key根据value生成哈希值；

presentcommit是当前HEAD中所记录的前次commit key；

—成员域

getPresentcommit，即从HEAD中获取前次commit key

HEAD相关方法，createHead()，即新建HEAD文件，updateHead()，即将提交后的commit key写入HEAD等；

return Key、Value、 Presentcommit等；

\*/

【回滚设计文档】

/\*

#回滚功能要求：

—给定一个commit，恢复对应文件夹并替换原先工作区的根目录；

即，查看当前分支的commit历史 ，找到某次历史commit对应的根目录Tree对象，恢复成一个文件夹；

\*Commit value里面存了前一次commit的key（链表遍历）；

\*根目录tree恢复成文件夹后，可以直接替换原先工作区的根目录；

+根据commit key查询得到commit的value；

+从commit value中解析得到根目录tree的key

+恢复(path)：

根据tree的key查询得到value

解析value中的每一条记录，即这个tree对象所代表的文件夹内的子文件与子文件夹名称以及对应的blob/tree key

-对于blob，在path中创建文件，命名为相应的文件名，写入blob的value

-对于tree，在path中创建文件夹，命名为相应的文件夹名，递归调用恢复(path+文件夹名)

+更新HEAD指针

\*/

/\*

#回滚-RollBack类设计：

public class RollBack{

数据域：

Private Tree rootDirTree;

private String rootDirTreeKey;

……

方法域：

构造方法（String theGivenCommitKey,String theGivenPath）{

解析commit对应根目录tree的key；

复原根目录Tree对应的文件夹；

更新HEAD指针；

}

解析commit对应根目录tree的key（String theGivenCommitKey）{

根据commit key查询得到commit的value；//调用Commit类方法；

从commit value中解析得到根目录tree的key;

返回根目录tree的key;

}

复原根目录Tree对应的文件夹（String rootDirTreeKey）throws IOException{

根据tree的key查询得到value;//调用Tree类方法；

解析value,获取子文件、子文件夹名称、对应的blob/tree key；

-对于blob，在theGivenpath中创建文件，

命名为相应的文件名，写入blob的value；//调用Blob类构造方法

-对于tree，在theGivenpath中创建文件夹，

命名为相应的文件夹名，递归调用恢复(path+文件夹名)//调用Tree类构造方法

}

}

\*/

/\*

其他：

定义一个静态方法，打印历史commit日志信息，以供用户查看并选取回滚目标；

\*/

【分支设计文档】

分支要求：

1、获取当前分支：（及显示当前所有分支的文件名）

将所有分支文件存放在专门的文件夹，每次通过遍历该文件夹显示当前分支。

2、显示每个分支的最新commit id

每个分支文件内容为当前的最新commit id，如果更新了提交，直接更改该分支文件的内容。

3、显示当前分支：

通过名为head的文件实现head指针功能，该文件保存当前分支的文件名，当切换分支时，直接更改head文件内容。

分支-Branch类设计：

public class Branch{

数据域：

Private Commit lastcommit;

Private String branchname;

Private String branchaddress；

......

方法域：

//构造方法

Branch(String branchname, Commit lastcommit){

将输入的文件名复制给当前文件名，更新最后一次commit；

若此次无commit，Branch中的commit可暂时为空；

}

//Branch存储方法

branchsave(){

在地址branchaddress的位置创建名为branchname的文件，文件内容为lastcommit；

}

//所有Branch显示方法

branchseek(){

遍历branch文件夹，显示其中所有文件名，及为当前所有分支；

}

//当前Branch显示方法

branchcheck(){

打开head文件，显示head文件内容，即为当前分支名；

}

//Branch交换方法

branchchange(String branchname){

在分支文件夹里查询是否存在名为branchname文件：

若存在，更新head文件；

若不存在，显示给用户，或直接新建一个分支，储存并交换；

}

}

【【注释整合】】

【Blob类】

* 读取文件路径，生成对应的文件（文件名为key，文件内容是value）
* 按照key去寻找value
* 新增对应文件，生成对应的KV存储文件

【Tree类】

* 直接按照路径构造tree

【Hash类】

* 创建程序所需要的储存空间；如果文件夹不存在，创建文件夹；
* 将key与value保存；
* 读取文件内容；
* 按行读取文件中的内容，将ArrayList转化为String类型返回；
* key-value文件读取;
* 将key-value储存为文件;
* 若对应键值不存在，说明文件已更新或未存储，则存储文件
* 依照输入流计算Hash值;
* 创建用于计算hash值的文件缓冲区，使用SHA1哈希/摘要算法，读出numRead字节到buffer中，根据buffer[0:numRead]的内容更新hash值，读完文件退出循环，返回SHA1哈希值
* 计算文件的Hash值；
* 计算文件夹的Hash值；
* 对当前文件夹进行遍历，储存文件及文件夹的key-value值；
  + 对文件夹进行深度优先遍历，返回该文件夹中文件的抽象路径名数组，采用ArrayList对listFiles()的结果进行排序，
  + 若为文件，计算Hash值并储存，将其key-value存储；
    - 若文件Hash值已更新或未存储，则显示Hash值；
  + 若为文件夹，将文件与文件夹的名称、Hash值记录在当前文件夹的ArrayList中，将ArrayList转化为String类型返回，存储key-value文件，
    - 若文件夹Hash值已更新或未存储，则显示Hash值；

【Branch类】

* Branch构造；
* Branch储存；
* 提交commit后更新Branch内容；
* 查找当前分支最新Commit；
* Branch创建；
* 所有Branch显示；
* 当前Branch显示；
* Branch切换；

【Commit类】

* Commit无参构造、仅给定Commit Key的单参构造、给定根目录路径、作者信息、提交者信息、备注信息、前次提交5个参数的构造；
* Commit创建；
  + 根据所给定工作区目录，对根文件夹进行遍历并返回对应key值；查找当前分支；
    - 若分支存在，创建commit并提交新内容；
      * 若不为首次提交，则创建前次Commit对象并进行后续操作，
        + 若与前次Commit储存根目录Key相同，则为重复提交，提示用户；
        + 若根目录Key不同，读入previousCommit，创建Commit，更新分支信息；创建Commit对象写入文件，更新分支信息；
      * 若为首次提交，则创建新的Commit对象写入文件，更新分支信息；
* object对象保存方法
* object对象读取方法
* 访问器：get方法

【RollBack类】

* RollBack无参构造、单参构造；
* RollBack以给定commit key givenCommitKey、给定储存路径givenPath为参数的构造；
  + 检查给定文件夹路径下是否已有，有则删除；
  + 根据给定路径，构成回滚仓库储存路径；
  + 根据给定commit key读取Commit文件内容获得根目录tree的key；
  + 解析根目录Tree的Key，依据该tree对象所代表的文件夹内的子文件与子文件夹名称以及对应的blob/tree key进行恢复；
  + 更新分支信息（更新指针）；
* 根目录Tree key读取方法；
* 根据所给commit key读取对应commit文件内容，获得第一行第二个标记符读取根目录Tree的key；
* 恢复方法；
* 对tree对象所代表的文件夹内的子文件与子文件夹名称以及对应的blob/tree key进行恢复;
* 清除方法；清除已有的文件夹；
* 历史Commit查询方法；展示当前历史Commit的日志；用于回滚前查询；
* 访问器方法；

【Commands类】

* 初始化设置方法；
* 命令行方法；
  + 命令为config，构造others对象；
  + 命令为init，初始化；
  + 命令为commit，提示输入相关信息并提交一次commit；
  + 命令为log，查看提交记录；
  + 命令为rollback，提示输入相关信息并回滚指定commit id对应内容至指定路径；
  + 命令为clone，提示输入相关信息并复制至指定文件夹路径；
  + 命令为branch，分支切换；
* 初始化方法；
* Main方法；