**CommitObject类说明 以及 分支回滚等设计文档**

**CommitObject类：**

**属性：**

String path 源文件路径

String versionPath 以KeyValue形式存储下文件路径

String author,String committer,String message 提交者相关的信息

String commitKey,String commitValue

String parent 上一次提交

String branchName 当前分支名

**String GitHub** 仓库

我们发现git add时，只能在根目录下add，不能点进文件夹中再add，因此我们设计时同样需要有一个根目录的思想

**LinkedList<CommitObject> commitList** 以LinkedList链表形式存储每一次commit（长久化存储在.dat文件）

每一次commit以CommitObject对象形式存储，利用LinkedList链表存储对象，每一个链表是一个分支，每一个链表存储在对应.dat文件中

（关于该链表在分支、回滚中操作的应用后续讲）

int numberOfCommits 提交的次数

String head 全局变量

Int numberofcommits 全局变量

**方法：**

**CommitObject()** 无参构造方法

**CommitObject(String path, String author, String committer, String message, String Github, String branchName)** 设置对象的path，author，committer，message，Github，branchName

检查有没有Source file 防止有人乱输入文件路径；

检查分支是否建立，如果没有建立，则我们为用户建立，并提示；

检查以分支名命名的.dat文件存在，来查看该分支上是否有过提交，便于查找parent。如果.dat文件不存在，说明没有上一次提交，提示用户这是第一次commit，没有parent；

访问本机IP地址作为我们的committer

**GetKey()** 获得对象的commitKey

**GetParent()** 获得上一次提交；如果没有上一次提交，输出提示信息"There exists no parent"

**toString()** 方法重写，将对象的commitKey，index，path，versionPath，author，committer，message信息转为String，便于后续打印commit信息

**WriteObject()** 把CommitObject类序列化，以对象组成与链表中，将链表写如dat文件；如果dat文件不存在，说明是第一次commit，直接将对象写入；如果dat文件存在，读取链表，并添加本次序列化对象

**功能需求：**

实现简易版本管理工具中分支、回滚、命令行交互功能

**分支管理要保存的信息**

保存分支信息

保存每个分支最新的commit\_id

当前处于哪个分支上

为每一个分支建立对应的文件夹环境，.dat文件记录该分支上每一个commitobject信息（文件最新的信息即为该分支最新的commit），LinkedList便于回滚时查找。

设置commit管理类进行分支管理，记录当前处于哪个分支。切换分支时，更新Head信息。

**分支切换与回滚**

本质上是把commit对应的根目录Tree对象恢复成一个文件夹

通过commitobject类ReadObject().toString()方法，查看所有commit信息——即git log功能

commitobject信息中记录有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的key和KeyValue文件路径path

根据path，创建文件恢复blob，通过mkdir创建文件夹递归恢复tree

**命令行交互**

通过main函数命令行参数String[] args接收用户指令

输入git branch XXX：创建分支文件夹，.dat文件以及LinkedList链表

输入git log：调用commitobject类ReadObject().toString()方法，打印commit信息

输入git reset XXX：从当前LinkedList链表上获取所需commitobject，根据commitobject对象恢复blob/tree