目录

[课程项目： 1](#_Toc8039)

[(1) 目标功能： 1](#_Toc5544)

[(2) 任务安排 2](#_Toc3171)

[(3) 项目类图及描述： 7](#_Toc8398)

[(4) 程序运行操作指南 8](#_Toc5156)

## 课程项目：

使用java语言建立一个简单的git本地版本控制系统。

1. 目标功能**：**

* 实现文件、文件夹的KeyValue存储
* 将文件目录内容保存至指定文件夹，并记录文件保存的相关信息
* 查看文件的保存历史记录（git log）
* 维护文件的提交序列，生成分支实现版本控制（git branch）

## 任务安排

* 第一周任务：

实现key-value存储（封装为class）：

1. 文件的key-value：

Key：文件名（value的hash值）

Value：文件中内容（key根据value计算得出）

功能：

-给定value，向存储中添加对应的key-value

-给定key，查找得到对应的value

1. 文件夹的key-value存储：

实现思路：遍历文件夹目录， 如果遇到子文件则转化为blob并保存。如果遇到子文件夹则递归调用内部的文件夹或文件，转化为blob与tree并保存

Key：tree文件夹value的hash值

tree文件夹的value：内部blob文件的key、每个子文件夹tree的key、子文件以及子文件夹的名称

1. 单元测试（unittest）

通过代码自动化的检测key-value存储实现是否正确，确保大量文件进行存储时的效率与正确率。

1. 实现设计：

* hash类：计算文件blob、文件夹tree的hash值。

-计算字符串（string型）参数的hash方法

-计算file的hash方法

-返回得到的hash值

* KeyValueObject: blob、tree的父类，子类blob与tree继承自object类

-计算key值

-创建类型为blob类型的文件

-创建类型为tree类型的文件

* blob类：

-计算该blob文件的key

-创建以key命名的blob文件

* tree类：

-计算该tree文件的key值

-以key命名的tree文件

### 第二周任务

实现commit

设计思路：

* commit类：

建立一个commit文件继承自keyvalueObject（以key-value存储。文件名为key，文件内容为value  
（1）Commit的Value：

1.项目根目录tree对象的key；

2. 前驱commit对象的key；

3. 代码author；

4. 代码commiter；

5. commit时间戳；

6. commit备注/注释

（2）commit的key： commit的value 的hash值

* head类 文件内存储最新commit的key
* 进行commit

每次生成的commit，将其根目录的tree与已有的最新commit的tree的key进行比较，文件内容value不发生改动时，hash值（key）不发生变动。对比key值当key发生变化时，添加一个commit对象，更新HEAD文件中存储的内容。

### 第三周任务

1. **分支**

实现设计：

分支部分应该保存：

a所有分支信息

b每个分支的最新commit id

c当前处于的分支

实现过程：

1. 创建一个类branch继承自keyvalueobject。
2. 创建一个名为branch的文件夹，文件夹中的每个文件是各个分支的信息。在初始化情况下，分支文件夹默认存有master分支，且分支内容为空。
3. 每新建一个分支，在branch文件夹中增加一个以该分支名命名的文件，每次commit将commit的key写入对应的分支文件中
4. 每次commit将其对应的分支名写入head文件中
5. **回滚**

1.从commmit记录中找到回滚目标，即对应的commit key（git log）

2.把commit对应的根目录Tree对象恢复成一个文件夹

（a）根据commit key查询得到commit的value

（b）从commit value中解析得到根目录tree的key

3.恢复(path)：

（a）根据tree的key查询得到value

（b）解析这个tree对象value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)

（c）更新head指针

（d）将head文件中的commit的key改为回滚到的commit的key

1. **分支切换**

（a）找到要切换的分支文件，获得该分支最新commit的key

（b）把commit对应的根目录tree对象恢复成文件夹（同回滚）

（c）更新head指针，将head文件中的分支记录改为切换为的分支

1. **命令行交互**
2. 实现的命令：

-git init:初始化仓库。

在当前目录中创建新的git版本控制系统。初始化仓库将在当前工作目录新建一个文件夹，文件夹内逐级新建用于储存commit、branch、head内容的文件。Init将自动生成一个名为master的主分支，生成一个head指向该分支。

-git add:添加文件至暂存区（仓库）

获得想要add的文件路径或文件名，判断其是否已经存在（比较key值），如果不存在则在目录下新建一个文件，将其转换为keyvalue存储形式保存至暂存区。

-git commit:提交文件

将暂存区文件提交至指定的branch中。

-git branch：获得所有的branch，显示每一个branch的key

实现思路：创建一个文件夹存储每一个分支branch的id，遍历读取文件中所有文件的文件名。

——git checkout “branchname” 切换到名为”branchname”分支

-git log: 获得所有的提交记录，显示每一次commit的key

实现思路：创建一个文件，每次提交将生成的commit的key写入文件中，执行git log操作，依次读出该文件中记录的commit的key以及找到对应的commit文件读出其value

git reset:回退版本，可以指定退回某一次提交的版本

## 项目类图及描述：

1.KeyValue类是所有key-value形式的对象的类的泛化，包括Blob，Tree，Commit，Branch，Head类。

2. Copy类、Read类、Hashcode类提供制造工具，涉及文件夹或文件的复制Copy类，所需文件内容的读取Read类，以及依据文件或文件夹生成哈希值的算法Hashcode类。

3.StaticCommand类主要用于提供对本地仓库和工作区的一系列静态操作：

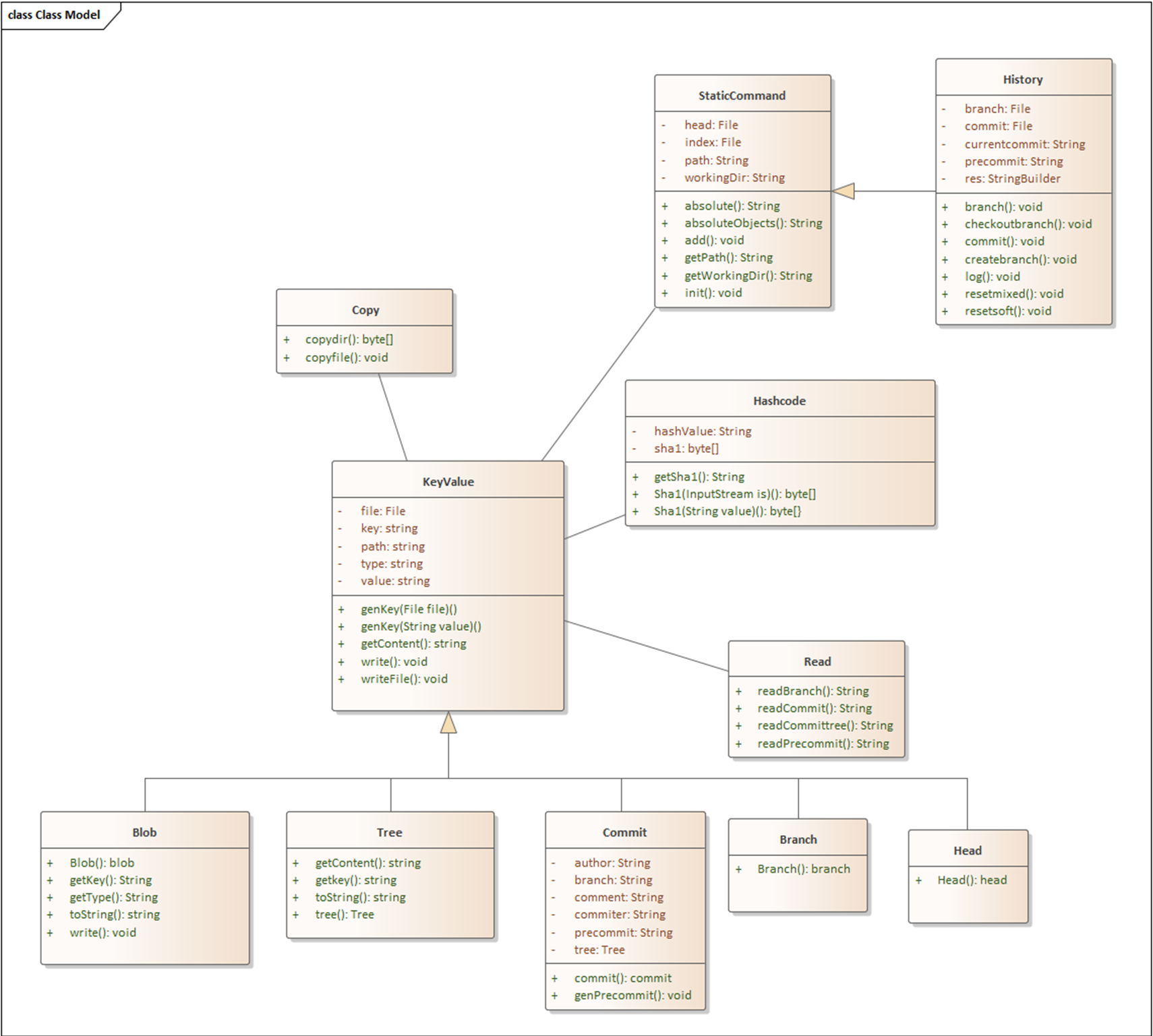
1. init()初试化本地仓库
2. add()添加文件到暂存区

4.History类提供关于commit的历史操作，包括：

1. log()方法查看commit历史
2. resetsoft()方法撤销commit到暂存区
3. resetmixed()方法撤销commit到工作区
4. branch()列出本地仓库中的所有分支信息
5. createbranch(branchName)创建一个新的分支
6. checkoutbranch(branchName)切换分支
7. commit()方法提交一次commit

5.主函数入口为Tgit类

6.单元测试类UnitTest



1. 程序运行操作指南**：**

**具体实现：**

通过main函数命令行参数String[] args接收用户指令

**编译：**

首先在程序文件夹下编译所有的程序：

javac -encoding UTF-8 \*.java

**设置环境变量：**

在系统变量中添加

变量名：CLASSPATH

变量值：.class文件所在的路径

保证.class在任意目录下都能运行

**输入命令：**

建立仓库：

Java Tgit init 初始化仓库

将文件添加进暂存区：

Java Tgit add

将文件提交至本地仓库：

Java Tgit commit

创建分支：

Java Tgit --createbranch

切换分支：

Java Tgit --checkout test（分支名）

查看commit记录

Java Tgit log