单元测试

BranchManager类

```
  // new branch
  String githubPath ="/Users/apple/Documents/txt 答疑解惑小笔记";
    String branch ="second";


    BranchManager branchOfHub = new BranchManager(githubPath);
    branchOfHub.newBranch(branch);
    //System.out.println( branchOfHub);

    ReadBranch(githubPath);
```

```
// read
  public static void ReadBranch(String github) throws IOException,
ClassNotFoundException{
    File file = new File(github+File.separator+"allBranch.dat");
    try (//create an input stream for file array.dat
          ObjectInputStream input =
          new ObjectInputStream (new FileInputStream(file));
        ){
        LinkedList<Branch> readBranch= new LinkedList<Branch>();
        readBranch =(LinkedList<Branch>) input.readObject();
        System.out.println( readBranch);
        }

  }
```

成功测试代码如上。

测试1 new BranchManager(githubPath)

```
String githubPath ="/Users/apple/Documents/txt 答疑解惑小笔记";
BranchManager branchOfHub = new BranchManager(githubPath);
```

运行错误：InputStream.readObject()无法执行。

错误分析：目标仓库目标文件“allBranch.dat”文件已经写入系统文件夹之中，因此会执行ReadBranch()方法。

应对测试：改写ReadBranch()进行单独测试。

测试2 ReadBranch()

```
public static void ReadBranch(String github) throws IOException,
ClassNotFoundException{
    File file = new File(github+File.separator+"allBranch.dat");
    try (//create an input stream for file array.dat
          ObjectInputStream input =
          new ObjectInputStream (new FileInputStream(file));
        ){
        LinkedList<Branch> readBranch= new LinkedList<Branch>();
        readBranch =(LinkedList<Branch>) input.readObject();
        System.out.println( readBranch);
        }
  }
  ReadBranch(githubPath);
```

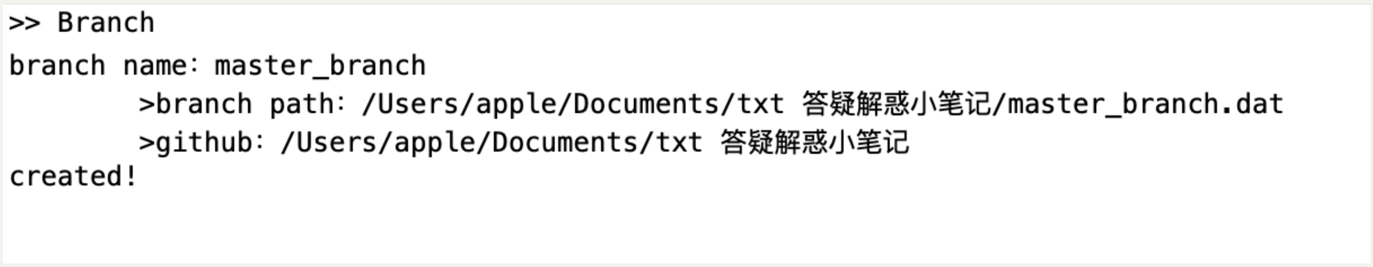
运行错误：InputStream.readObject()无法执行。

错误分析：writeBranch()正常运行，应当检查Branch类的可序列化

应对测试：单独打印Branch的实例化对象。

测试3 Branch

```
    Branch newbranch = new Branch(branch);
    newbranch.setGithub(githubPath);
    System.out.println(">> Branch" + newbranch + "created!");
```



```
>> Branch
branch name: master_branch
        >branch path: /Users/apple/Documents/txt 答疑解惑小笔记/master_branch.dat
        >github: /Users/apple/Documents/txt 答疑解惑小笔记
created!
```

测试分析：Branch可以序列化，并且implements serializable，但仍然无法读取出来，也许是因为Branch是作为BranchManager的一个内部静态类造成的, 将Branch类单独作为一个非静态类。

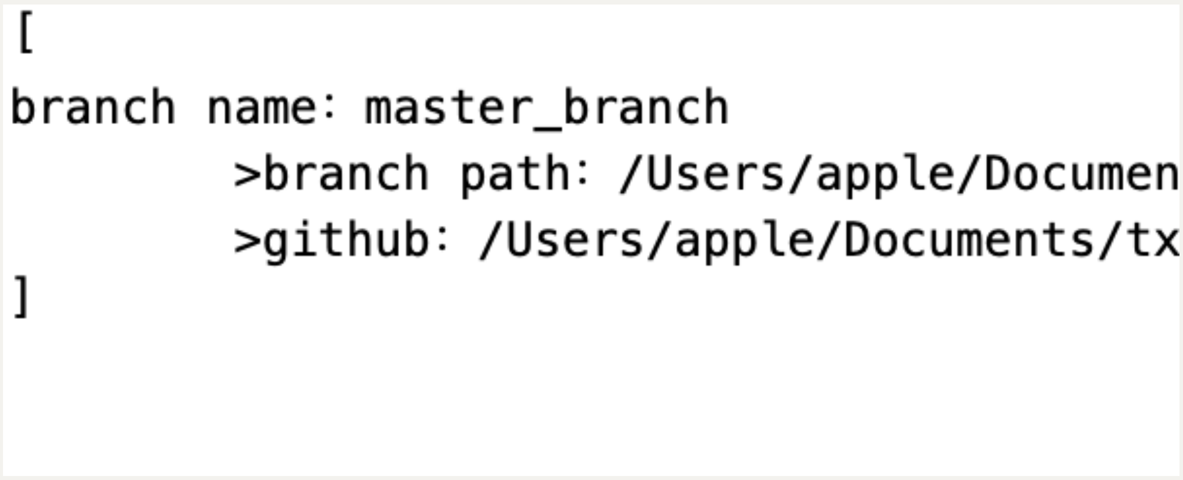
测试4 Branch对象的链表能否写入dat文件

```
  //单独创建链表测试能否写入对象
  Branch newbranch = new Branch(branch);
    newbranch.setGithub(githubPath);
    System.out.println(">> Branch" + newbranch + "created!");

    LinkedList<Branch> allBranch = new LinkedList<Branch>();
    WriteBranch( githubPath, allBranch);

    //write方法
    public static  void WriteBranch(String github, LinkedList<Branch>
allBranch) throws IOException {
    File file = new File(github+File.separator+"allBranch.dat");
    file.createNewFile();

    try ( //Create an output stream for file array.dat
            ObjectOutputStream output = new ObjectOutputStream(new
FileOutputStream(file));
          ){
            output.writeObject(allBranch);
            }
    }
```

测试分析： writeBranch()方法没有问题;

测试5 Branch对象的链表能否从dat文件中读出

```
Branch newbranch = new Branch(branch);
    newbranch.setGithub(githubPath);
    System.out.println(">> Branch" + newbranch + "created!");

    LinkedList<Branch> allBranch = new LinkedList<Branch>();
    allBranch.add(newbranch);
    WriteBranch( githubPath, allBranch);
    ReadBranch(githubPath);
```

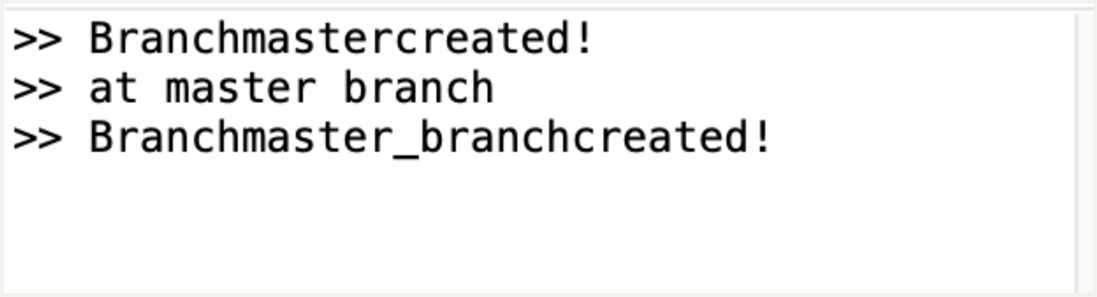


```
[
branch name: master_branch
            >branch path: /Users/apple/Documen
            >github: /Users/apple/Documents/tx
]
```

测试分析：ReadBranch()方法没有问题

测试6 测试BranchManager类的构造与建立新分支

```
BranchManager branchOfHub = new BranchManager(githubPath);
    branchOfHub.newBranch(branch);
```



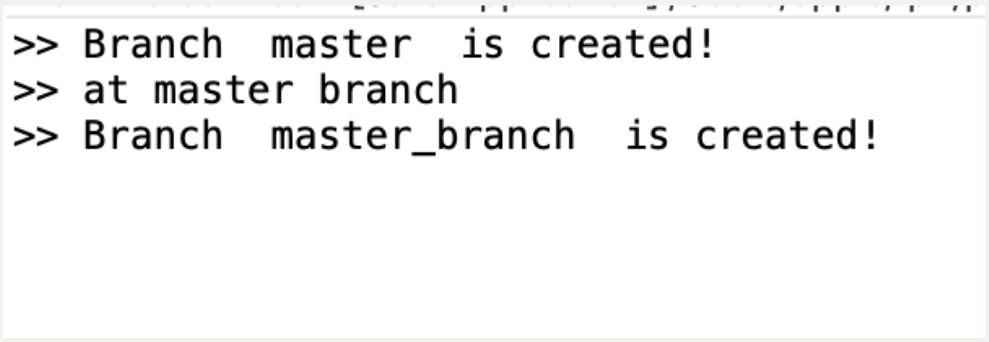
```
>> Branchmastercreated!
>> at master branch
>> Branchmaster_branchcreated!
```

测试分析：提示语句有误会。

测试7 测试newBranch()的default与非default状态

```
//BranchManager的创建中
else {
      newBranch("master");
      System.out.println(">> at master branch" );
      WriteBranch();
      head_ofBranch =allBranch.get(0);
    }

    //.newBranch()
    public void newBranch(String branchName) throws IOException {
    Branch newbranch = new Branch(branchName);
    newbranch.setGithub(this.github);
    allBranch.add(newbranch);
    numOfbranch++;
    WriteBranch();
    System.out.println(">> Branch" + branchName + "created!");
  }
```

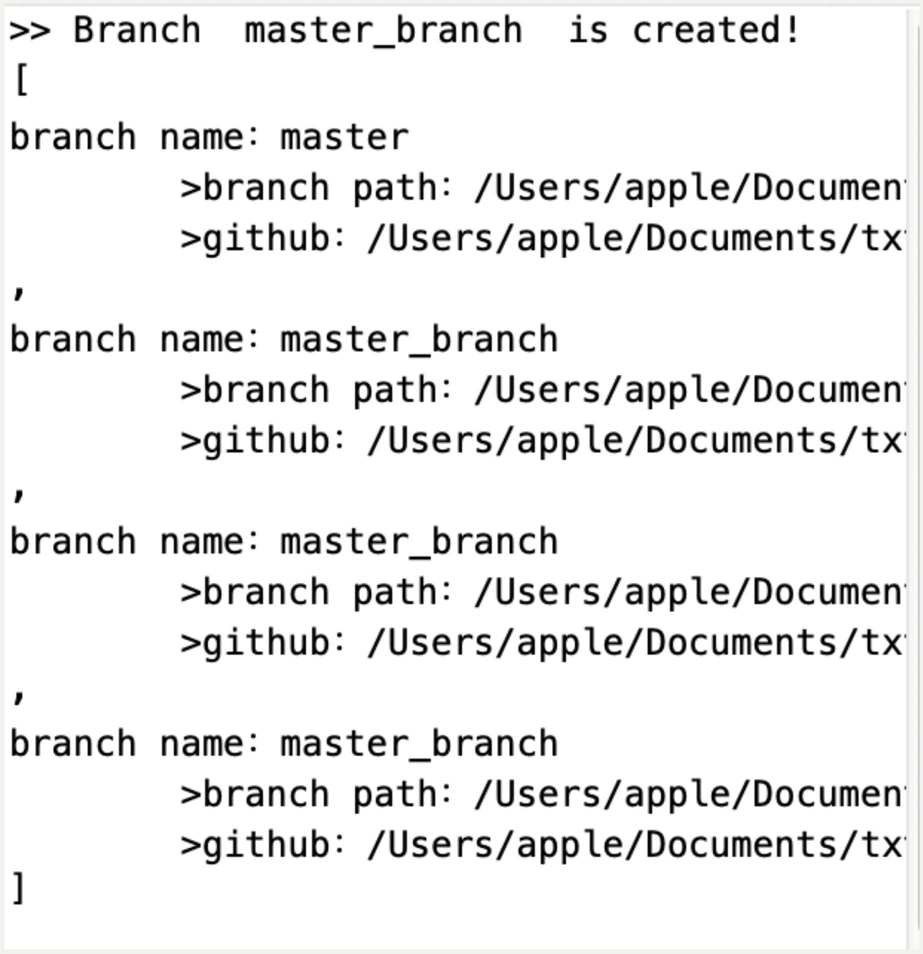
测试分析： 经过调整格式，当从未创建分支的时候建立仓库分支管理将创建新的分支，并打印提醒（因为调用的是同一个函数）。手动添加新的分支提示成功。



```
>> Branch  master  is created!
>> at master branch
>> Branch  master_branch  is created!
```

测试8 newBranch()出现重复创建分支的情况

```
public StringBuilder iterator() {
    StringBuilder names = new StringBuilder();
    if(numOfbranch>0) for (Branch each :allBranch)
names.append(each.branchName);
    return names;
  }
  public void newBranch(String branchName) throws IOException {
    if(numOfbranch>0 && iterator().indexOf(branchName)!=-1) {
      System.out.println(">> Branch  " + branchName + "  already
created! change a name");
      return;
    }

    Branch newbranch = new Branch(branchName);
    newbranch.setGithub(this.github);
    allBranch.add(newbranch);
    numOfbranch++;
    WriteBranch();
    System.out.println(">> Branch  " + branchName + "  is created!");
  }
```



```
>> Branch  master_branch  is created!
[
branch name: master
        >branch path: /Users/apple/Documen
        >github: /Users/apple/Documents/tx
,
branch name: master_branch
        >branch path: /Users/apple/Documen
        >github: /Users/apple/Documents/tx
,
branch name: master_branch
        >branch path: /Users/apple/Documen
        >github: /Users/apple/Documents/tx
,
branch name: master_branch
        >branch path: /Users/apple/Documen
        >github: /Users/apple/Documents/tx
]
```

测试分析：成功修改重复创建分支的问题，且可以成功读取文件。

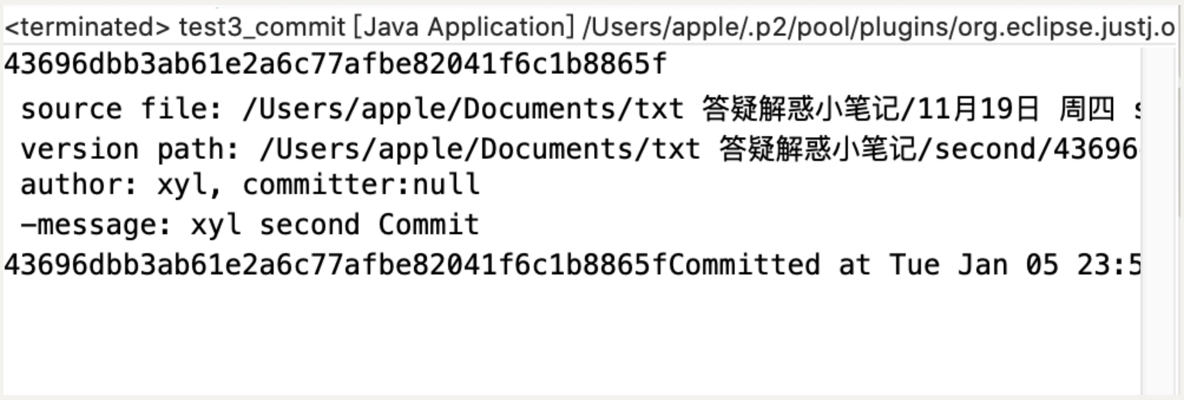
CommitManager类

测试newCommit()

```
String committer = null;
      String author = "xyl";
      String message = "xyl first Commit";
      String message2 = "xyl second Commit";
String githubPath ="/Users/apple/Documents/txt 答疑解惑小笔记";
      String branchName ="second";

      String path = "/Users/apple/Documents/txt 答疑解惑小笔记/11月19日
周四 schedule.rtf";
      CommitManager commitOfBranch= new CommitManager
(githubPath,branchName);
      commitOfBranch.newCommit(path, author, committer, message2);
```

运行错误：Cannot invoke "java.util.LinkedList.add(Object)" because "this.commitList" is null

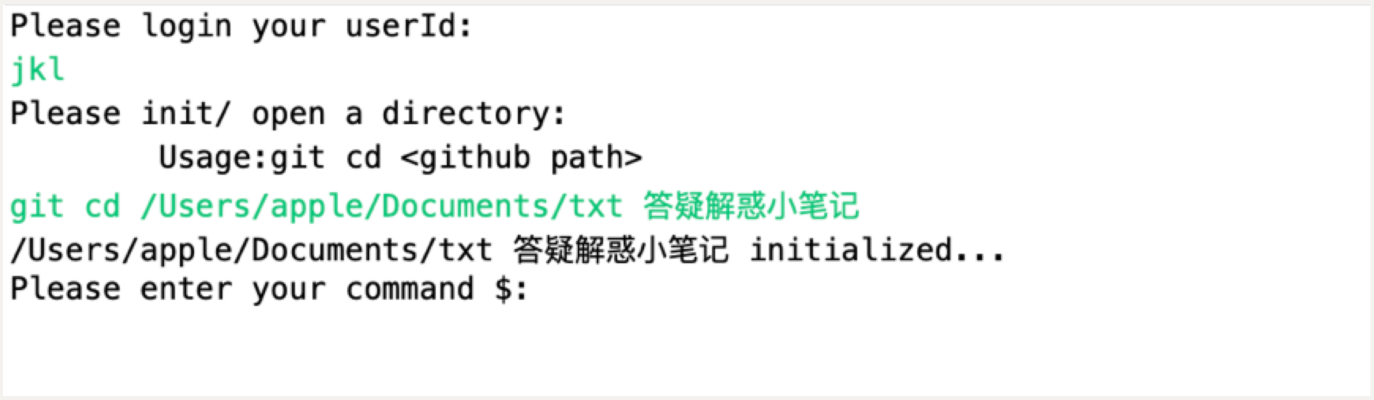
错误分析： 在分支dat文件不存在的情况下，链表设置成null了，删除该代码，运行成功



```
<terminated> test3_commit [Java Application] /Users/apple/.p2/pool/plugins/org.eclipse.justj.o
43696dbb3ab61e2a6c77afbe82041f6c1b8865f
 source file: /Users/apple/Documents/txt 答疑解惑小笔记/11月19日 周四 s
 version path: /Users/apple/Documents/txt 答疑解惑小笔记/second/43696
 author: xyl, committer:null
 -message: xyl second Commit
43696dbb3ab61e2a6c77afbe82041f6c1b8865fCommitted at Tue Jan 05 23:5
```

gitCommand类测试

1、打开仓库，默认创建master分支

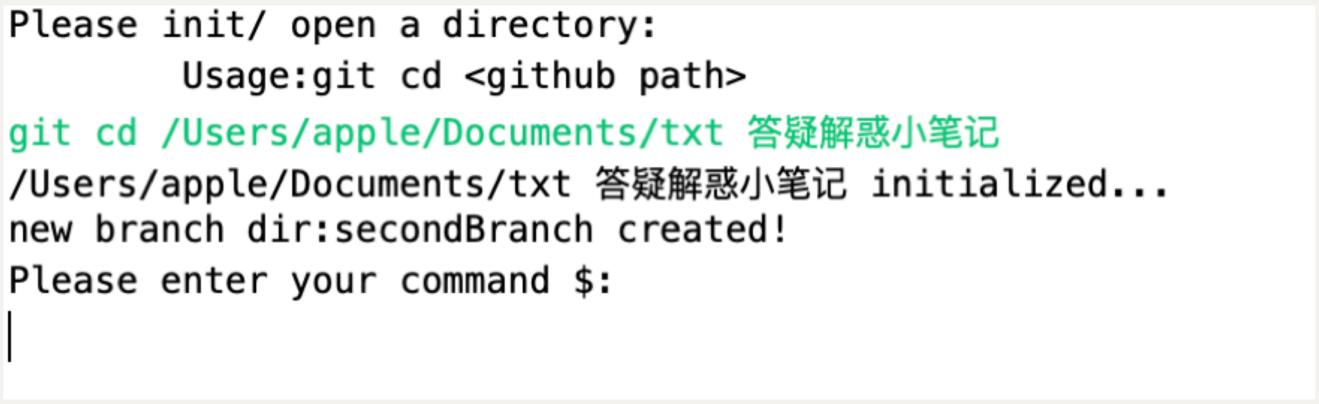


```
Please login your userId:
jkl
Please init/ open a directory:
        Usage:git cd <github path>
git cd /Users/apple/Documents/txt 答疑解惑小笔记
/Users/apple/Documents/txt 答疑解惑小笔记 initialized...
Please enter your command $:
```

结果分析： 使用git cd打开仓库，提示初始化。

1.第一次打开仓库，会默认创建master branch

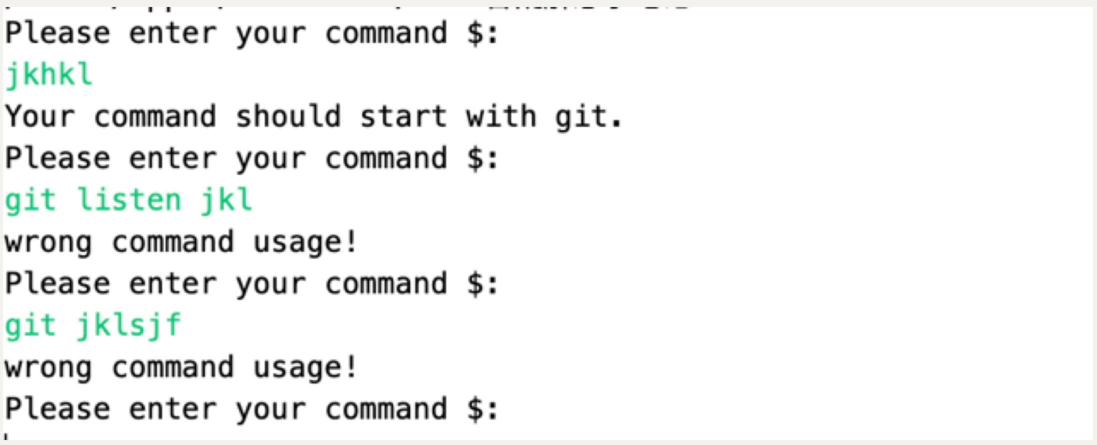
2.并将head指针指向master



```
Please init/ open a directory:
        Usage:git cd <github path>
git cd /Users/apple/Documents/txt 答疑解惑小笔记
/Users/apple/Documents/txt 答疑解惑小笔记 initialized...
new branch dir:secondBranch created!
Please enter your command $:
|
```

3.如果是重新打开仓库，会默认进入上次新创建的分支之中。

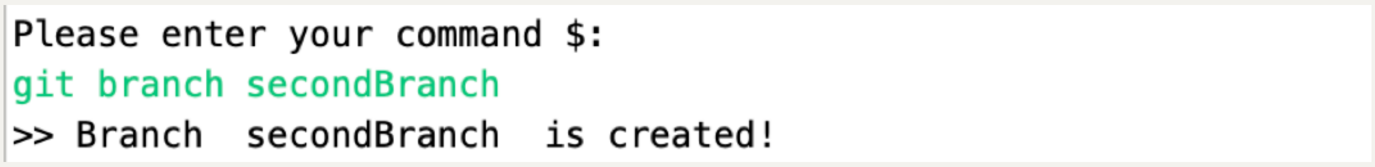
2、命令合法性检验


```
Please enter your command $:
jkhkl
Your command should start with git.
Please enter your command $:
git listen jkl
wrong command usage!
Please enter your command $:
git jklsjf
wrong command usage!
Please enter your command $:
```

结果分析：

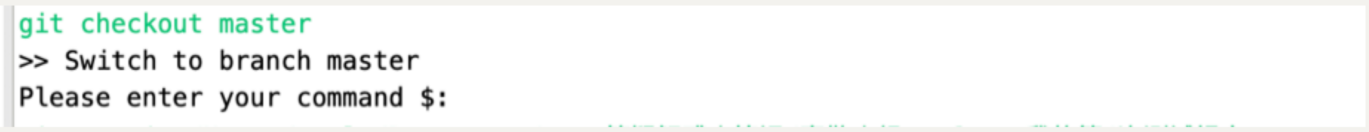
1. 如果不以git命令开头，会提示需要以git开头
2. 如果git命令开头，后面是乱敲的命令，
   会提示错误命令

3、 branch

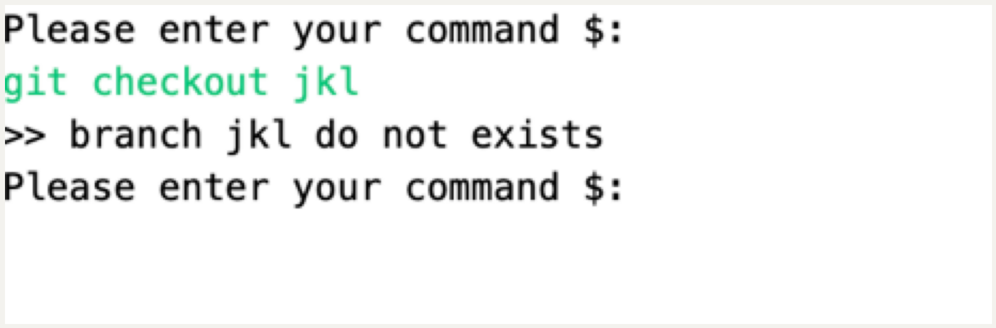

```
Please enter your command $:
git branch secondBranch
>> Branch  secondBranch  is created!
```

结果分析：创建新的分支，并提示用户新分支建立

4、checkout


```
git checkout master
>> Switch to branch master
Please enter your command $:
```

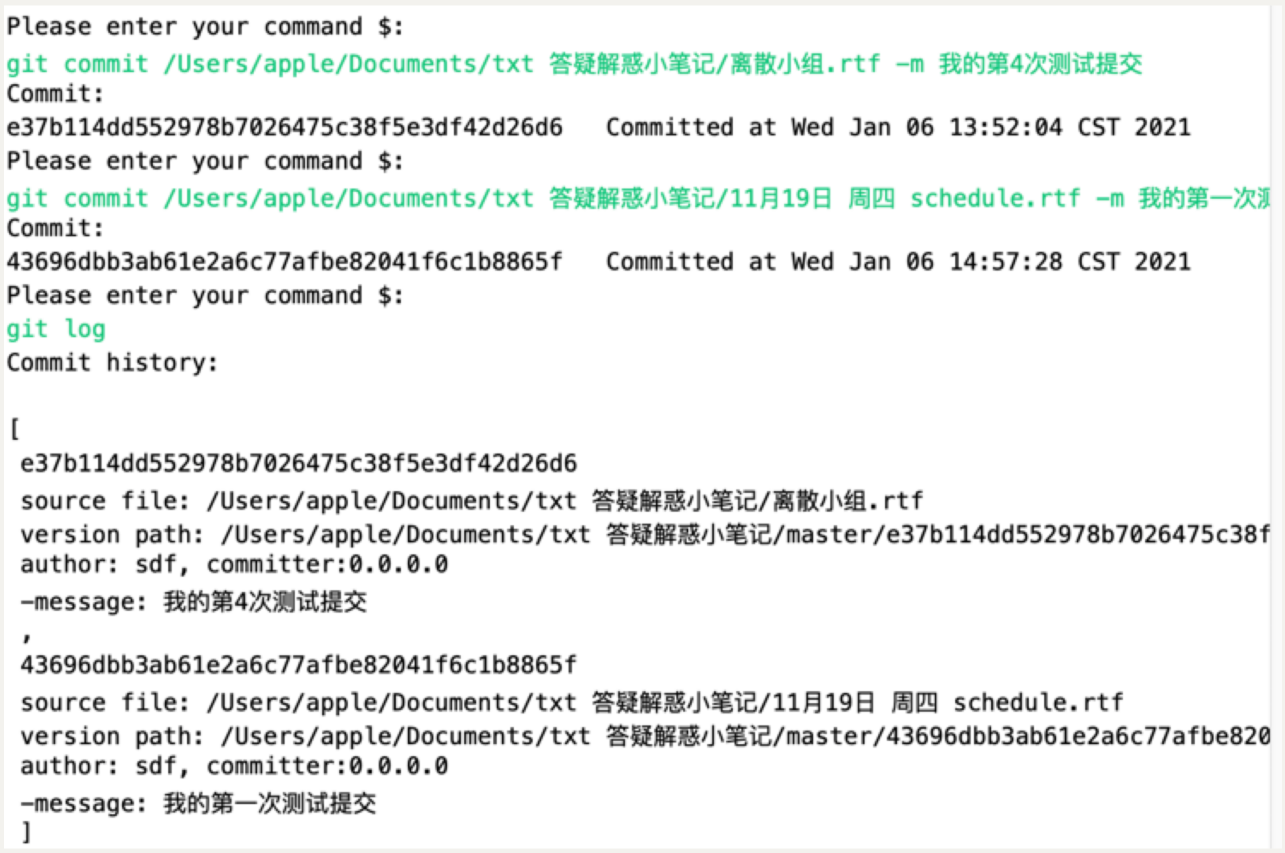
结果分析：提示切换分支



```
Please enter your command $:
git checkout jkl
>> branch jkl do not exists
Please enter your command $:
```

结果分析： 如果乱敲一个从未创建过的分支，则会提示branch不存在

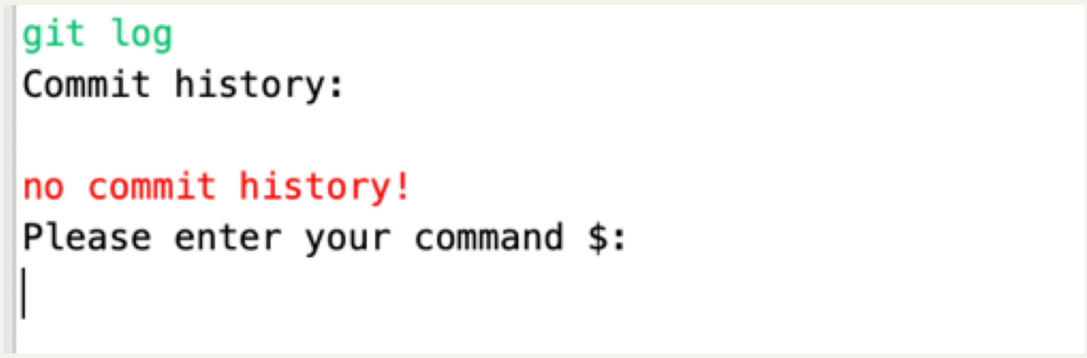
5、commit & log



```
Please enter your command $:
git commit /Users/apple/Documents/txt 答疑解惑小笔记/离散小组.rtf -m 我的第4次测试提交
Commit:
e37b114dd552978b7026475c38f5e3df42d26d6   Committed at Wed Jan 06 13:52:04 CST 2021
Please enter your command $:
git commit /Users/apple/Documents/txt 答疑解惑小笔记/11月19日 周四 schedule.rtf -m 我的第一次测
Commit:
43696dbb3ab61e2a6c77afbe82041f6c1b8865f   Committed at Wed Jan 06 14:57:28 CST 2021
Please enter your command $:
git log
Commit history:

[
 e37b114dd552978b7026475c38f5e3df42d26d6
 source file: /Users/apple/Documents/txt 答疑解惑小笔记/离散小组.rtf
 version path: /Users/apple/Documents/txt 答疑解惑小笔记/master/e37b114dd552978b7026475c38f
 author: sdf, committer:0.0.0.0
 -message: 我的第4次测试提交
 ,
 43696dbb3ab61e2a6c77afbe82041f6c1b8865f
 source file: /Users/apple/Documents/txt 答疑解惑小笔记/11月19日 周四 schedule.rtf
 version path: /Users/apple/Documents/txt 答疑解惑小笔记/master/43696dbb3ab61e2a6c77afbe820
 author: sdf, committer:0.0.0.0
 -message: 我的第一次测试提交
 ]
```

结果分析：

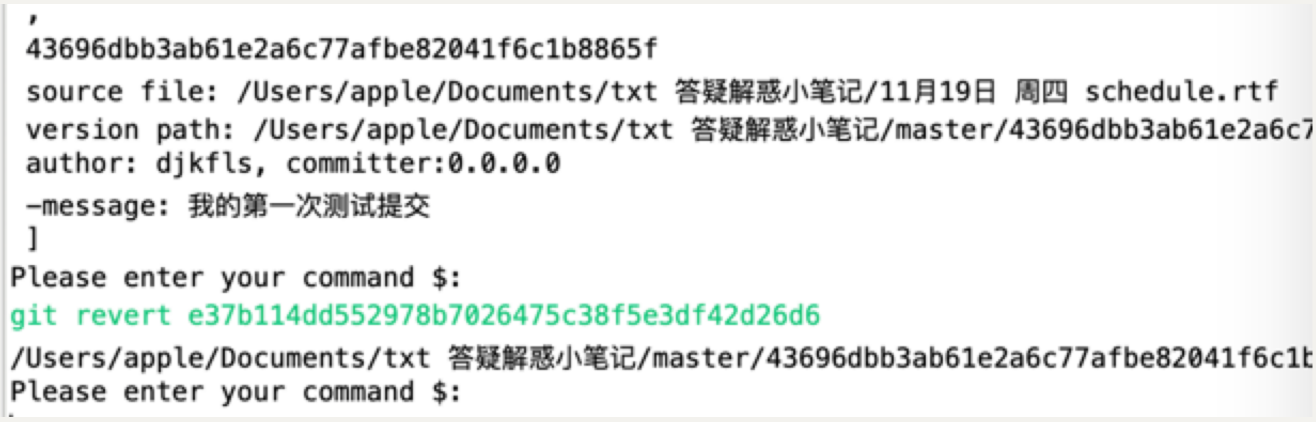
1. 输入git commit -m 进行提交，会提示用户于何时何刻提交，并反馈哈希值；
2. 通过 git log进行查看



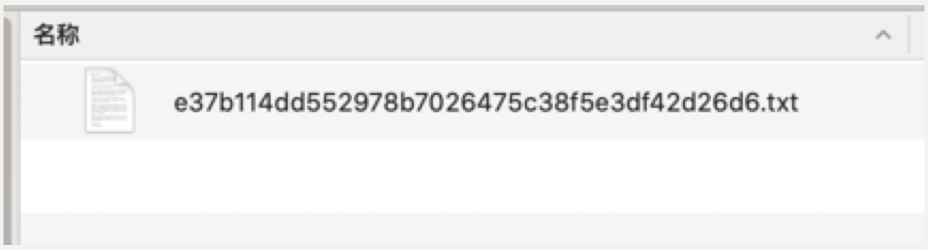
```
git log
Commit history:

no commit history!
Please enter your command $:
```

3. 如果没有提交，则会提示无历史记录

5、git revert



```
 ,
 43696dbb3ab61e2a6c77afbe82041f6c1b8865f
 source file: /Users/apple/Documents/txt 答疑解惑小笔记/11月19日 周四 schedule.rtf
 version path: /Users/apple/Documents/txt 答疑解惑小笔记/master/43696dbb3ab61e2a6c7
 author: djkfls, committer:0.0.0.0
 -message: 我的第一次测试提交
 ]
Please enter your command $:
git revert e37b114dd552978b7026475c38f5e3df42d26d6
/Users/apple/Documents/txt 答疑解惑小笔记/master/43696dbb3ab61e2a6c77afbe82041f6c1b
Please enter your command $:
```



名称

e37b114dd552978b7026475c38f5e3df42d26d6.txt

结果分析：Git revert之后，结点后的文件就会被删除