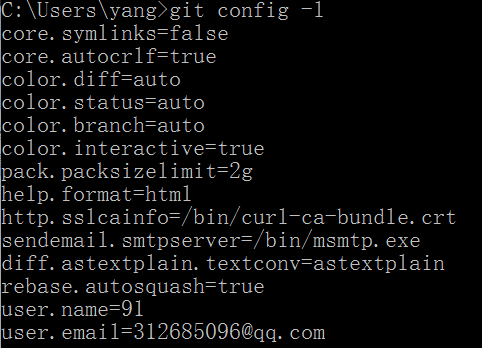
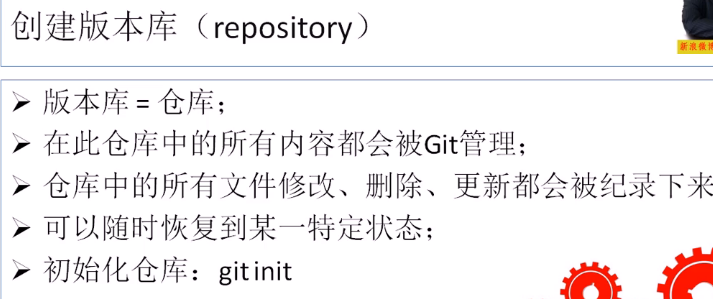


设置完成后如果成功不会有任何提示信息，而用户可以通过如下命令查看配置信息。

**范例：**查看全局信息

git config -l



此时这台电脑上所做出的所有修改，都会自动的将用户名、email地址发送到主程序之中。

创建文件夹（z:\mypro）

Z: md mypro

此时mypro文件夹是一个空文件。

进入文件夹

cd mypro

初始化仓库（将此目录变为可以被GIT管理的仓库）

git init



此时会提示，在mypro文件夹中创建了一个“git”的目录，这个目录就是仓库信息！不能改！

**添加文件**

****

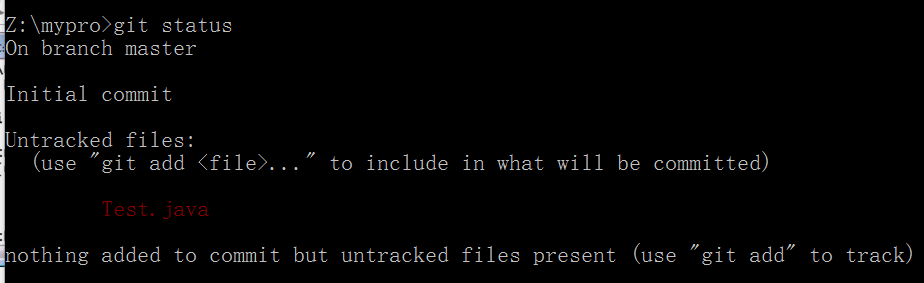
所有文件用UTF-8

**范例：**建立一个.java文件

Text.java文件保存在了mypro目录下，且mypro目录属于git管理的仓库

**范例：**查看当前仓库的状态

git status



On branch master——现在的状态开发属于主分支

Initial commit——初始化仓库的提交

Untracked files:——未标记的文件

随后是一些操作的命令——(use "git add <file>..." to include in what will be committed)

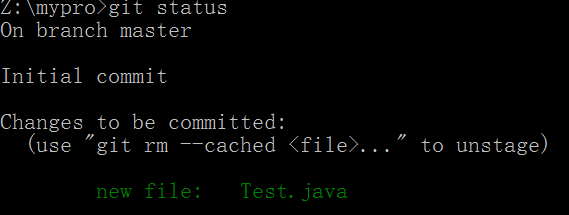
Test.java——未标记文件的列表

**范例：**将文件加入暂存库之中

git add Test.java

**范例：**随后继续查询状态

git status



现在的文件并没有真正地提交到主分支上（真正要运行的程序的所有代码）。

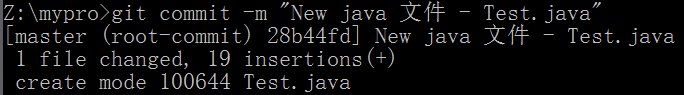
**注意：所有修改的代码都会被git自动检测到，所有的代码在使用commit提交之前一定要先用add增加进来，否则不会有任何提交。**

**如果现在不希望分两步进行，则可以在运行以下程序是增加一个“-a”的参数，表示先add而后commit（git commit-a -m“注释”）**

**范例：**提交文件信息

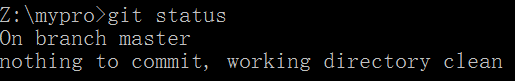
git commit -m“New java 文件 – Test.java”

在进行每次更新提交的时候，一般会增加一些注释数据，所以使用“-m”参数来进行注释编写。



此时，“Test.java”文件就真正提交到了主分支上，发布成功了

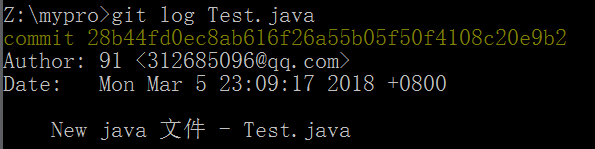
再次查询：git status



此时的状态提示：没有任何的信息需要被提交，工作目录很干净。而在git工具下，用户每一次进行的提交实际上都会被日志记录下来。

**范例：**查看针对“Test.java”文件的日志消息

git log Test.java



“28b44fd0ec8ab616f26a55b05f50f4108c20e9b2”，可以理解为是每一次提交的id号，如果有多次提交，那么这个日志信息会越来越多

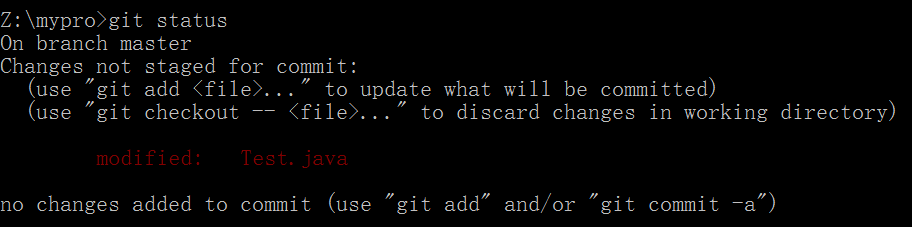
**修改仓库文件**

****

**范例：**修改Test.java文件

文件增删了几行的代码

查询现在的仓库状态：git status



Changes not staged for commit:——文件没有保存到暂存区中

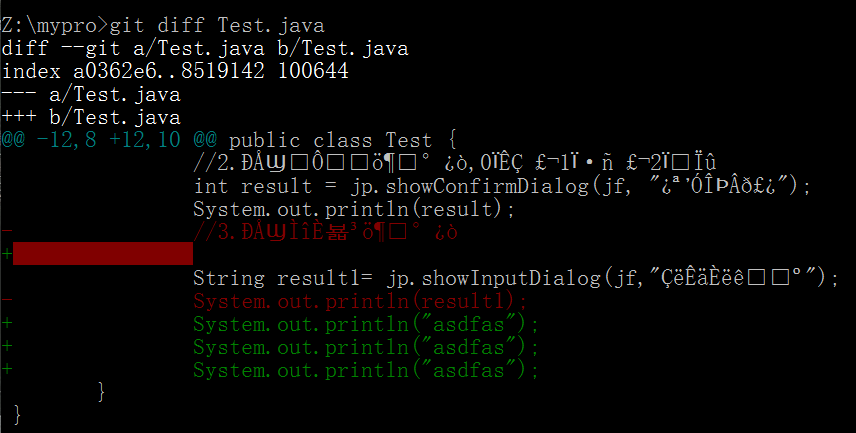
(use "git add <file>..." to update what will be committed)——选择暂存

(use "git checkout -- <file>..." to discard changes in working directory)——选择文件恢复

modified: Test.java——已经修改的文件

**范例：**查看文件的前后区别

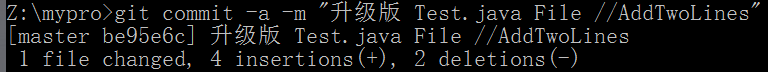
git diff Test.java



所有增加的内容使用“+”表示，减少的内容用“-”表示

**范例:**将修改后的代码加入到暂存区后进行提交

git commit -a -m “升级版 Test.java File //AddTwoLines”



**范例：**查看日志：git log Test.java



可以发现，只要是修改的操作git都可以进行及时的跟踪

**工作区和暂停区**

