一、关于Vi/Vim的学习

Vi/Vim是大部分linux系统下比较常见的编辑器，但是我个人觉得这两种编辑器其实不怎么好用，可能和用惯的windows系统的界面不大一样，用不顺手。相对来说，我觉得sublime比较好一点，界面比较舒服，而且还能下载插件，有自动高亮，自动补全功能。而Vi/Vim不能用鼠标，高亮的颜色我也不会调。

关于使用问题，在终端下键入vim/vi 均可以进入编辑器，如图：



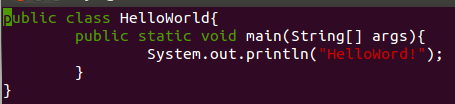
按“i”可以进入输入模式，开始敲代码，需要退出的话则先按“ESC”，然后“：wq”保存并退出。

二、关于java的学习

首先是安装java，手动下载然后配置环境变量，因为当初在windows下尝试过，觉得挺麻烦的，果断选择了包管理自动安装，这个是我最喜欢linux下编程的一点，比windows下方便许多，只要有包，所有需要用的软件都可以一句命令“sudo apt-get install”完成。

接下来就是关于java的具体学习，在假期的时候零零碎碎地学了点，在写java小程序的时候写得还是比较顺的，但是那个代码并没有做太大的优化，因为我在看书上的一个计算机的源代码的时候发现它所用到的知识点远远比我所写的简单计算器要多。

简单的HelloWorld源码如下：



在终端下编译执行java程序也很简单，命令如图：

A description...

javac是编译，java是运行。

最后的输出也是直接在终端下显示的。

在学习的过程中我发现java的基础语法其实和C++非常相似，例如类、继承等知识点。所以学习起来的难度并不高，但是在学习到GUI部分的时候就开始有许多的新东西，光是看知识点其实并不够，多看一些源代码理解起来会更快一点。

在刚开始学最需要注意的一点是，java文件名和里面的类名需要相同，否则会报错。

简单完成了java小程序之后使用sonar检测代码，发现了一些不规范的写法，例如在判断相等的时候，最好使用equals而不是“==”

三、关于ant的学习

我同样是使用包管理自动安装，因此不需要配置环境变量，而且我进入 /etc/profile 查看的时候发现环境变量并没有改变。

测试是否安装成功：A description...

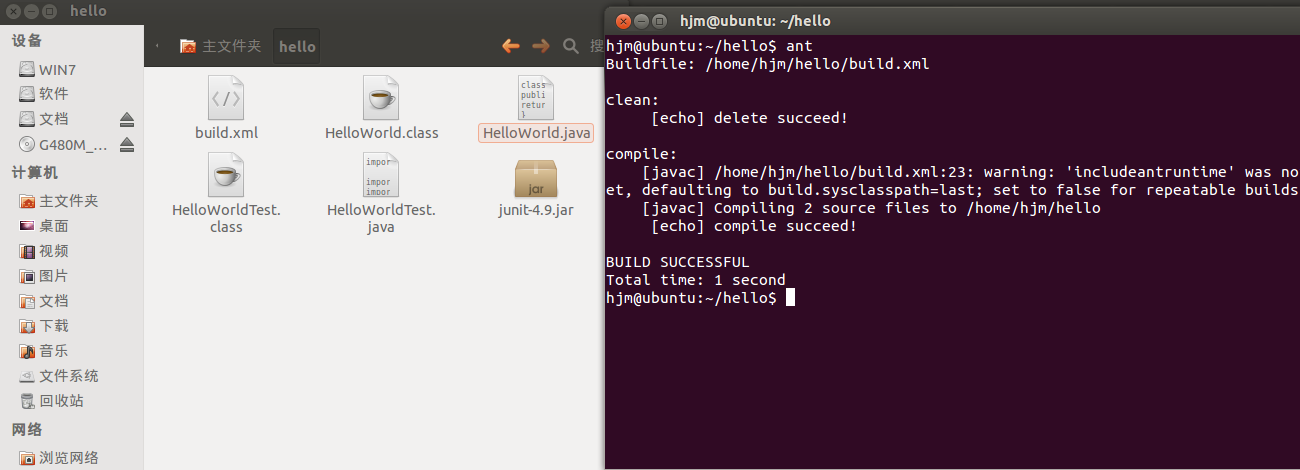
如图所示，我在还没有写build.xml的时候build failed了，也就是说我安装成功了。

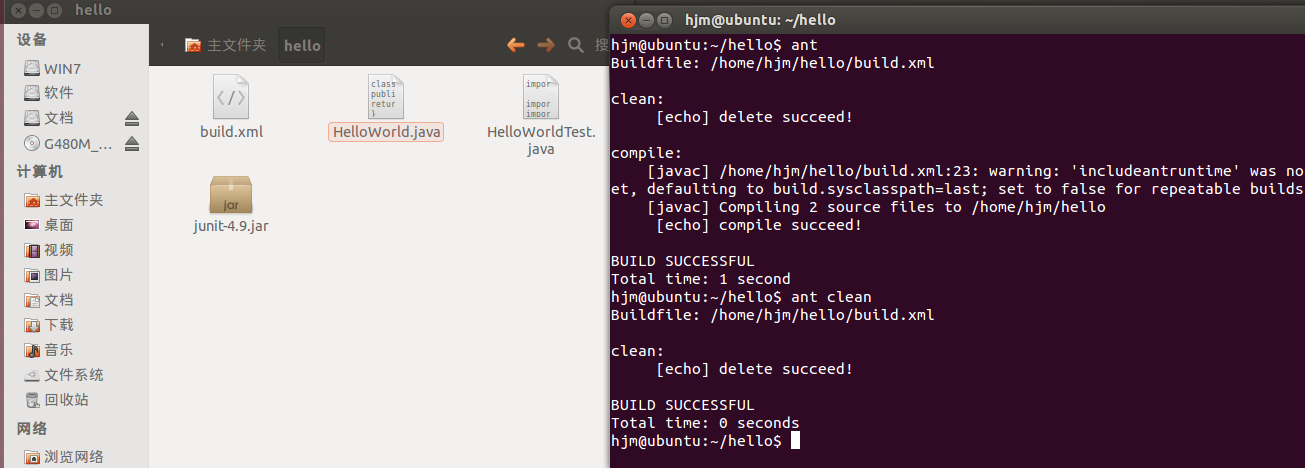
然后就是学习ant的那些属性，标签等等的含义，一次过背下来太不现实了，我都是需要用的时候去查，用多了就熟悉了。

我个人理解ant其实就是一个脚本的功能，写好之后执行ant就可以执行你所需要的执行的事件。

终端下直接敲ant就会执行设定default事件，若例如“ant clean”就会执行clean事件。用着我发现和上一年实训用到的make有点异曲同工之妙。

执行ant clean前后截图如下：



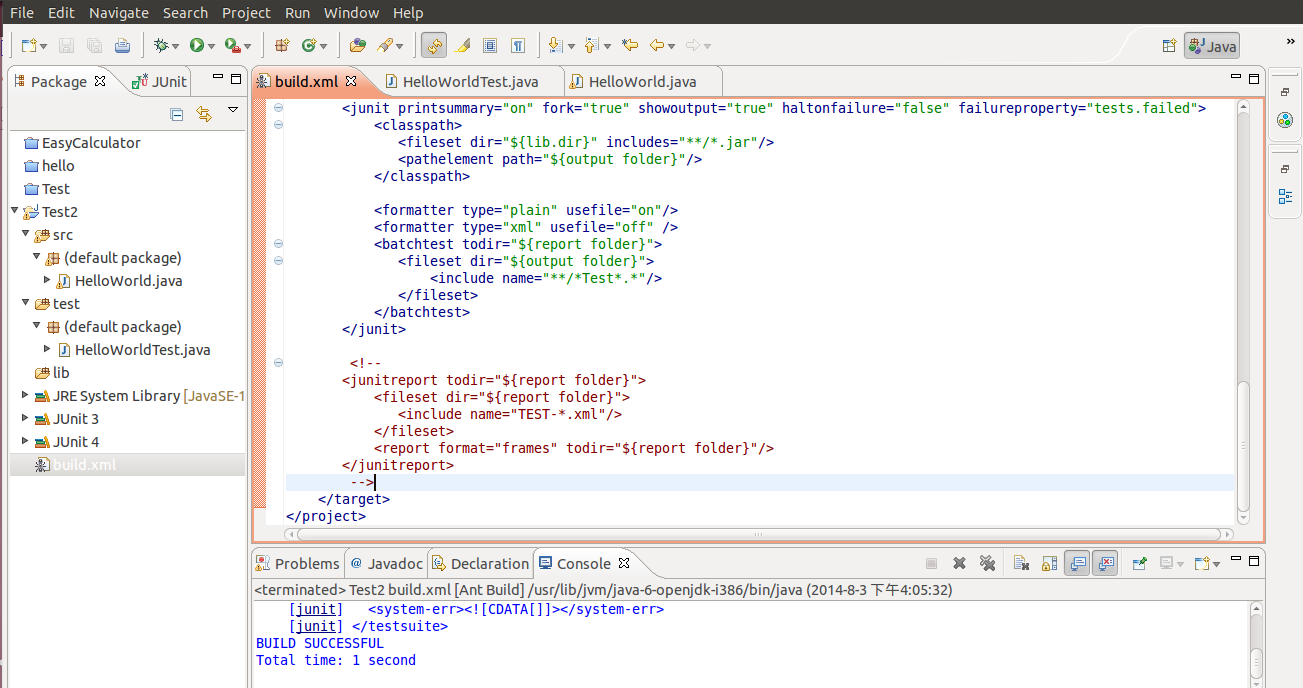


class文件确实在执行之后被清除了。

每次都清除是一个很好的习惯。

在Eclipse里面能集成ant，需要进行一点配置，和终端下输入命令相比，我觉得这个出错的可能性会稍微低一些。

我在Eclipse下使用ant的截图如下：



可以在下面看出我“build succeed”，成功运行并且JUnit也通过测试了。

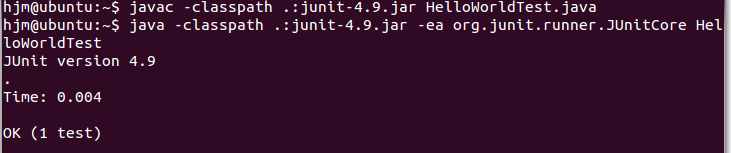
这是ant和JUnit结合，这个学习过程是通过大量的百度，学习整合别人的源码慢慢摸索出来的，在Eclipse下单独使用ant编译打包jar和单独使用JUnit进行单元测试我接受得都很快，但是两者结合起来，因为百度上也有一些不是非常正确的教程误导我，试了比较长时间才成功。

四、关于JUnit的学习

Junit是单元测试，类比上一年实训，我觉得和GTest也还是蛮像的，都是判断期望输出和实际输出是否相同。

可以比较方便的测试自己代码的函数是否有问题，否则debug大概就要在每个函数那里输出值或者打断点查看变量的值，没有JUnit来得方便。

如果在终端下编译运行的话必须要有依赖包，命令如下：

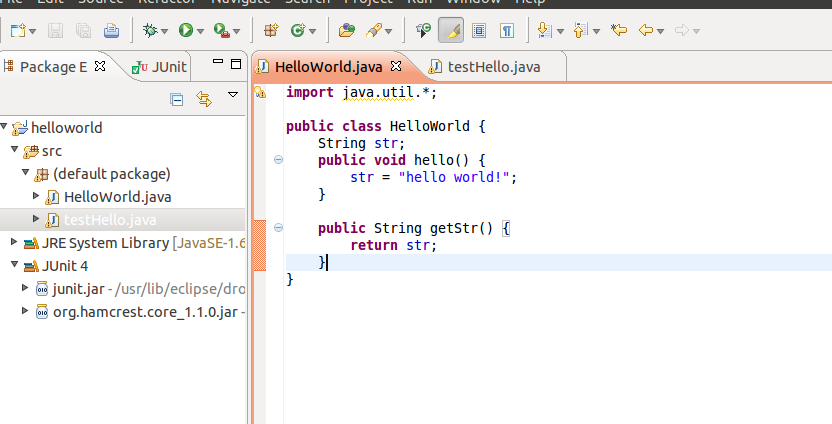


可以看到JUnit的版本，运行时间，最后“OK”表示测试通过了。

因此相对来说我更喜欢使用Eclipse，它自动集成了ant，Junit等，只要进行一些配置就可以很方便地使用。

配置为过程为：类右键选择“Build Path”，“Add Libraries”，“Junit”，“Junit4”

Eclipse下新建一个通类，一测试类，如图：

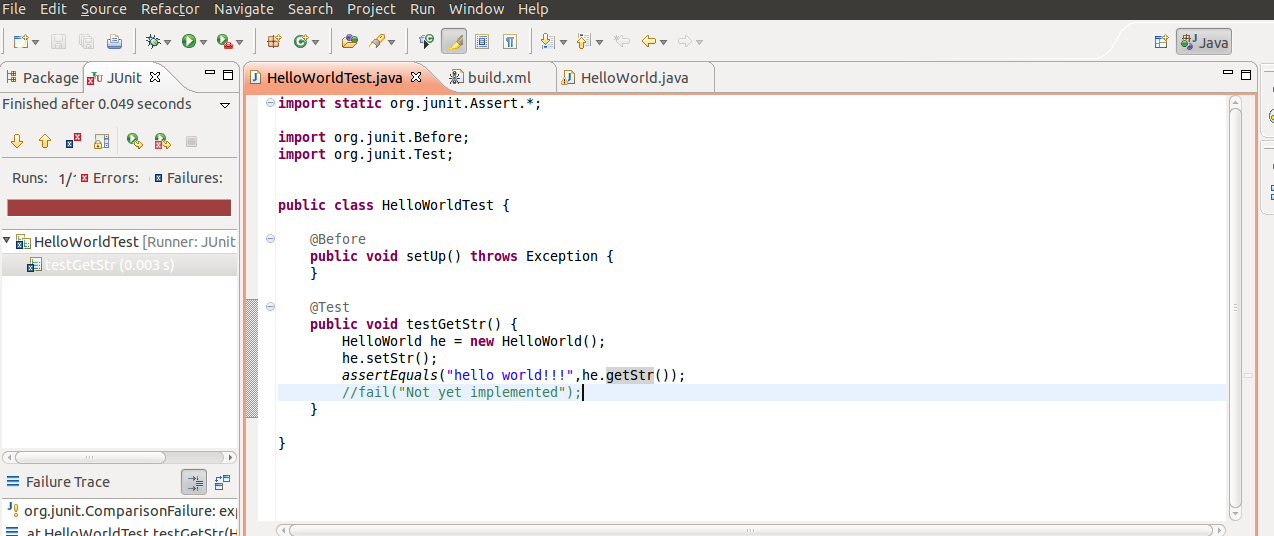




测试函数的时候必须要先有那个类的一个对象，才能调用相关函数。

运行的时候只需要“run as JUnit Test”

出来的结果：



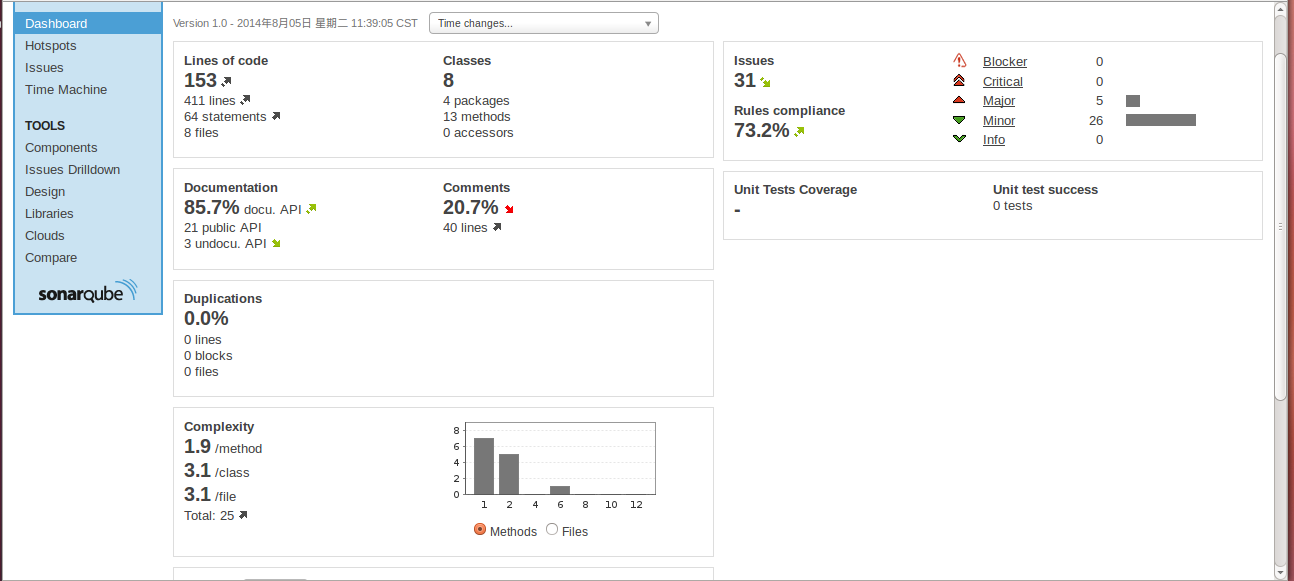
右侧的红色条代表不通过，如果是绿色则是通过了。

五、关于sonar的学习

在配置和学习sonar的过程中并没有遇到太多的问题，安装的过程比较顺利，运行的时候并没有出现无法解决的问题，只是不知道因为什么原因，start sonar之后打开localhost:9000，有时候会出现延时的情况，要刷新较多的次数，等待一段时间才能出现页面。

有一点需要注意的是，运行sonar-runner的时候，必须先打开localhost:9000，否则会build failed。

可以看到，页面很简洁，下面是我已经在检测一个项目了。

从页面上可以很明显得看到项目的复杂度，注释量，以及代码是否符合规范。

尤其是规范问题，我们刚开始学习java，有一些用法虽然可以用但是不推荐的我们并不大清楚，在检测之后，对照着改是很好的学习方法。因为不仅能看到你错了，还能看到错在哪里，改起来很方便。

还有注释，其实我之前不是非常习惯写注释，现在也慢慢开始有这个习惯了。