自学报告

仅供参考

标签： vi Java Ant JUnit SonarQube

- vi编辑器的使用
  刚开始我在虚拟机上按照教程学习使用vi，真的是一场煎熬。俗话说：孰能生巧，在使用vi的过程上我深深体会到。下面我主要讲vi的基本操作，这也是我在学习中经常用到的一些操作。
  vi分为三种模式，分别是command mode，insert mode，last line mode。我们用在终端输入vi+文件名进入vi编辑器时，是command mode，在命令行模式下可以进行的操作有很多，最为常用当然是插入：i，a，o，按这些字母将进入插入模式，此外还有移动光标操作，删除文字等等。若要从其他模式退出来，我们按ESC键即可回到命令行模式。另外，我们写文件当然要保存了，首先要确保在命令行模式下，我们按下:号(冒号),即可进入last line mode，一般输入wq即可保存并退出。若只需保存仅按w，若只需退出则按q！(注意感叹号)。
  vi还有很多命令，在此就简单review上面几点。
  说实话，vim很强大，各种插件很多，但是原生vi对新手来说真的是一种折磨。通过这次vi的学习，逐渐了解了vi，接下来还是需要多使用vim，特别要体验它的插件。
- Java语言的学习
  首先简单介绍一下我在虚拟机上配置Java环境吧。根据教程在Sun公司下载JDK，然后安装JDK(涉及chmod解决可执行权限问题，执行bin文件等)，配置环境变量JAVA_HOME,PATH,CLASSPATH，source生效，重启系统。在整个过程中，当然是配置环境变量最困难了，因为/etc/profile是readonly，又是因为尝试使用vi去添加环境变量，vi真的难用，所以一开始一直失败，弄了好久才配置好。
  `java -version` 哇，终于配好了。
  接下来，学习Java语法。第一感觉，和C++很像，基本语法相差无几。和C++不同的是，Java是纯粹面向对象的程序设计语言，处处体现着面对对象编程的思想,因此Java并没有C++中的指针操作，不能再类外定义全局变量，只能在某个类中定义一种公用静态变量来实现全局变量的功能；不再支持头文件等等。
  因为寒假看过一些Java，但我发现还有很多知识需要学，也想通过这次实训更多地学习Java，进一步掌握Java。
  今天算是实训的第一天，要求用Java写个简单的计算器。其实计算器加减乘除逻辑性很简单，难点在于

GUI界面，怎么让写的代码可以通过交互来实现计算器功能。通过资料查找发现有awt、swing库，可以实现布局管理，我的计算器用到了GridLayout这种布局。布局好后，通过设置监听器和实现其接口即可简单实现计算器。

- Ant配置和学习

  关于Ant配置和学习，由于云桌面已经帮我们配置好，所以我是在虚拟机上配置的(云桌面有权限，不能修改profile，但好像可以修改bashrc来代替)。按照提供的教程，在官网上下载后解压到指定目录，设置系统环境变量(嗯，我又用到vi了，就是这样不断熟悉vi的)，重登系统，直接ant和ant -version测试是否安装成功。

  环境配置：vi /etc/profile(虚拟机上)

  ```
  export ANT_HOME=/usr/local/ant
  export PATH=$PATH:$ANT_HOME/bin
  ```

  检查测试：

  命令行：

  ```
  ant
  ```

  显示：

  ```
  Build build.xml does not exist!
  Build failed
  ```

  命令行：

  ```
  ant -version
  ```

  显示：

  ```
  Apache Ant version XXX compiled ...
  ```

  配置成功了！

  接下来就是学习Ant了。几大关键元素project、target、property、task；对应的属性的作用；Ant的常用任务copy、delete、move、mkdir、echo；最重要的：利用Ant构建和部署Java工程(利用javac任务编译Java程序，利用java任务运行Java程序，jar任务生成jar文件等等)。

  Ant是一个非常强大的工具，可实现项目的自动构建和部署等功能，类似于我们之前用过的makefile，通过这次简单地使用，对项目管理有一定的帮助，值得我们深入研究。

- JUnit的学习

  在JUnit的学习过程中，由于教程上面那个@sser被搞蒙了好久，睡了一觉发现自己好傻，真傻，明明那个是用户名...

  JUnit4是JUnit框架有史以来的最大改进，是用来进行单元测试的框架。

  由于要求使用junit-4.9，所以要先把junit-4.9.jar解压放到同个项目文件夹里(这个非常重要，如果没这个，要配置好eclipse的junit)，编写HelloWorldTest.java进行简单的单元测试，使用以下两条命令运行：

  ```
  javac -classpath .:../../junit-4.9.jar HelloWorldTest.java
  java -classpath .:../../junit-4.9.jar -ea org.junit.runner.JUnitCore HelloWorldTest
  ```

  (这里路径要保证正确，上面仅供参考)

  当终端显示JUnit version 4.9等信息说明配置成功。

接下来就是写Ant实现JUnit的自动测试了。由于实训提供的教程并没有实现ant和JUnit的自动测试，TA也没检查这个，一开始一直不知道build.xml中要怎么添加junit-4.9.jar的包，我在网上找了一个教程Ant实现JUnit自动测试，完美解决了我的问题，现在已经可以实现自动测试了，目录结构在这省略不写。关键xml代码如下：

```
<!-- junit包路径lib -->
<path id="compile.path">
    <fileset dir="${libDir}">
        <include name="*.jar"/>
    </fileset>
    <pathelement path="${destDir}"/>
</path>
<!-- 执行测试，注意compile.path与上面对应 -->
<target name="junit" depends="clean,compile">
    <junit printsummary="yes">
        <classpath refid="compile.path"/>
        <test name="HelloWorldTest"/>
    </junit>
</target>
```

补充

SonarQube是一个用于代码质量管理的开源平台，用于管理源代码的质量。下面简单谈谈SonarQube的配置。

1.已安装Java环境

2.安装sonar和sonar-runner：将zip包解压。

3.设置SONAR_HOME,SONAR_RUNNER_HOME环境变量，并将SONAR_RUNNER_HOME加入PATH。

具体操作：

`sudo vi /etc/profile`

(云桌面 `gedit ~/.bashrc`)

添加下面几条命令

`export SONAR_HOME=.../sonar-3.7.4/bin/linux-x86-64` (...表示路径名)

`export SONAR_RUNNER_HOME=.../sonar-runner-2.4` (云桌面我们是用sonar-scanner,所以这里我们要把sonar-runner改成sonar-scanner)

`export PATH=$SONAR_RUUNER_HOME/BIN:$PATH`

最后保存退出，在终端输入 `source /etc/profile` (云桌面上 `source ~/.bashrc` ;重启系统。

4.添加数据库(略)

5.启动服务

shell里输入

`cd $SONAR_HOME`

`./sonar.sh start` 启动服务
`./sonar.sh stop` 停止服务
`./sonar.sh restart` 重启服务

使用SonarQube Runner分析源代码
创建sonar-project.properties配置文件，进入含该文件的目录下输入sonar-runner，打开localhost:9000查看即可。
注意每次使用完sonar记得关闭，进入启动目录， `./sonar.sh stop`