# 实训报告

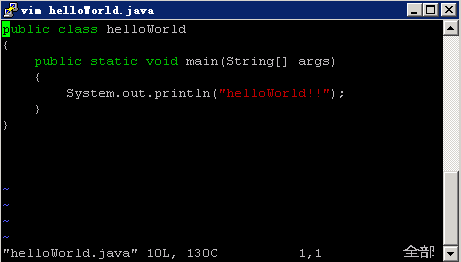
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **作业名称** | Vi, Java, Ant, JUnit自学报告 | | | | | |
| **院系** | 软件学院 |  | **班级** | 教务2班 | **姓名** | 李明宽 |
| **学号** | 11331173 |  | **完成日期** | 2013年07月02日 | | |
| **QQ** | 736459905 |  | **E-mail** | limkuan@mail2.sysu.edu.cn | | |

今天是中级实训的第二天，大概总结一下这两天做了什么。由于使用过Linux，对Linux不是很陌生，因此使用起来不是很困难，然后学习了神之编辑器Vi的一些基本用法，用Vi编辑了helloWorld.java，然后使用ant实现了helloWorld.java的自动编译（总觉得这个词不能这么用，应该是解释？）运行，以及打包成jar文件，最后学会了如何使用JUnit来测试部分的函数功能是否正确，检查环境无误后，成功编译运行了BugRunner.java。

###### Vi的学习心得

以前用的都是另一个神级编辑器Emacs，因为有现成的配置文件，然后根据自己的习惯又对配置文件修改了写，然后让Emacs使用起来非常方便。今天第一次使用Vim，体会到了大神常说的用Vim多了后Esc键会被按坏的意思了o(╯□╰)o

虽然说对于没用过Vim的人来说，编辑方式太复杂，经常要在命令模式与插入模式之间切换，而且一些常用的快捷键不按熟就会感觉很奇怪，例如使用h、j、k、l这四个键来进行光标移动，很明显好处就是能够手指不离开键盘进行光标移动，但是对于不熟练的使用者来说可能就经常会按错，而且在插入的时候如果打错会习惯输入光标上下左右，然后就会出现神奇的字母ABCD，然后又要回到命令模式里删除。虽然说Vim不支持鼠标定位，但是可以直接在命令模式里使用一些组合按键快速定位到后几个单词或者第几行等等这些位置，对于代码编辑来说是很灵活的。



在我今天使用的Vim，是没有经过配置的Vi，没有语法高亮，也没有自动补全之类的插件或者配置文件，再加上我个人以前对Vi都不是很熟悉，在看完Wiki上提供的帮助文档后，以及阅读了Vim的帮助文档Vimtutor，勉勉强强能在Vim上写出了HelloWorld.java，如上图所示（没使用虚拟机，使用了putty连接Linux主机），然后使用”:!{command}”这种方法执行带个shell命令，在Vim中对java源码进行解释运行，而不用再次退出Vim进入shell命令行界面运行，这对编程效率又是一个很大的提高，如果能完善Vim的配置文件，让它支持语法高亮以及自动缩进自动补全，然后还要熟练掌握Vim的快捷键，这样，Vim一定会变成帮助我们写代码的利器。

###### Java学习心得

首先要安装JDK完成Java环境的配置，在Ubuntu下，我选择了使用包管理软件apt-get来安装，在终端中输入命令：sudo apt-get install openjdk-7-jdk，然后包管理软件就会在软件源里找到相应的软件包并下载完成安装，不用再次麻烦配置Linux下的环境变量。

对于Java这门语言，由于有学习过Android的开发，当时粗略地看了一下Java的相关书籍，就是感觉语法跟C++很像，对C++比较熟悉的话是比较好学习的，但是它是解释性语言，而且所有的东西都是对象，每一个文件都包含一个类，类名要求与文件名相同。自我感觉Java对类掌握要求较高，一些继承，多态等特性会经常使用。

通过编写了简单的HelloWorld程序，大致了解了Java的一些特性，包括使用命令javac命令对源代码进行解释，使用java命令来运行Java程序。但是由于对Java的一些特性不熟悉，在后来对BugRunner的编译运行中出现了很多奇怪的问题还没解决，可能要等到以后慢慢熟悉整个GridWorld的框架，这样才好找出解决问题的方法。

在接下来的学习过程中，我们要继续深入学习Java，并了解在Eclipse IDE下对Java工程进行编译运行的方法，继续深入学习对GridWorld的具体实现。

###### Ant学习心得

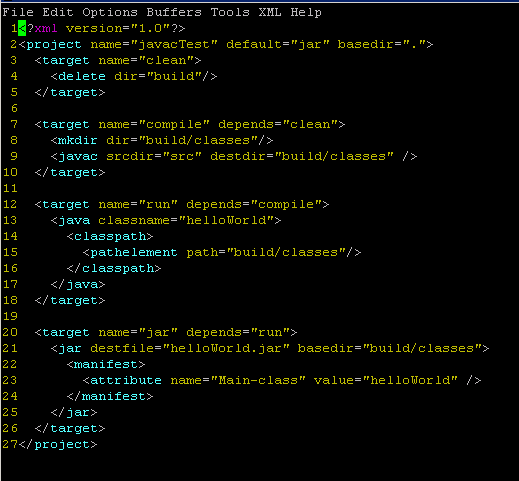
第一次听说Ant这个软件，以前对它的一些用途用法都不是很熟悉，经过学习了帮助文档，渐渐感觉到了Ant的强大。Ant是一个基于Java的生成工具，可以实现对Java工程的自动编译打包运行，如果一个代码项目很大了之后，每次对代码的重新编译，打包测试都很复杂，但是使用了Ant后，只要编写好这个项目的build.xml，就可以很方便对项目进行自动编译运行。

了解了Ant的用法后，感觉Ant长得很像C/C++里面的Make，功能都差不多，都是使用脚本对源代码进行批量编译或者运行，但是Ant使用的脚本是基于XML的，因此书写维护起来都比较方便。而且Ant还有诸多如跨平台，操作简单，结构清晰等优点是Make无法达到的。

使用Ant首先要编写好build.xml的运行脚本。脚本里大概包含3大部分：

* 首先是一些定义的property元素，它包含name和value属性，一般是一些工程里使用到的目录，文件名等等，例如源代码目录src等。在后面使用到的时候可以使用${name}的方法来使用在这里定义好的value。这个部分在写HelloWorld的build.xml的时候没有用到。

下图是HelloWorld的build.xml，包括了编译，运行，以及打包成jar。

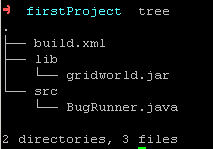
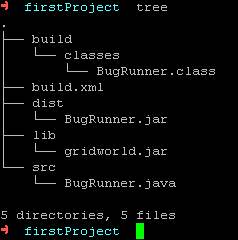


* 第二大部分是project元素，它定义了这个构建文件的属性，每个构建文件都有一个project元素。这个元素有下面一些属性：default表示默认的运行目标，这个属性是必不可少的，basedir表示项目的基准目录，默认是当前目录，name表示项目名，description表示项目的描述。
* 第三部分是target元素，一个project元素下可以有一个或多个target标签。target有以下的一些属性，name表示这个target的名字，这个属性是必须的，depends表示这个target依赖的目标，这要求运行这个target前一定要运行它依赖的目标，description表示项目的描述。

然后在运行Ant的时候，一般是在终端运行ant+target.name，如果不加target的名字，那运行的就是project中default中默认的目标。

在学习Ant后，写了HelloWorld的自动运行脚本，在配置完GridWorld的运行环境后，尝试使用ant来实现BugRunner.java的自动编译运行，但是过程中遇到了很多问题，主要是编译BugRunner需要使用外部的jar包gridworld.jar，在ant中要引用这个包，教程中没有提到这种方法，通过查询网络上的资料，终于找到了解决办法。

左图所示的目录树是BugRunner所在的Project的目录，其中包括build.xml，lib与src子目录，lib里包含gridword.jar，src中包含BugRunner.java。运行后的目录变化如右图所示，会在添加子目录build，这里是解释出来的BugRunner类，然后子目录dist是可执行的BugRunner的jar包，可以直接使用java –jar BugRunner.jar运行。

下图是build.xml的代码，包括了4个target，分别是clean，complie，jar，run。



这个实现的要点是创建project.class.path，包含lib里的文件夹所有要用到的jar文件，在编译的时候使用classpath包含进去。然后要先打包然后使用这个jar包来运行。打包的时候manifest中attribute中，使用Class-Path包含外部的jar包，要具体写每一个包的名字，而不能使用\*.jar代替。

完成了这个build.xml的编写，感觉对ant的脚本文件的编写又有了进一步的理解，更加深入理解了ant的用法。

###### JUnit学习心得

JUnit是java的单元测试工具，用来测试项目中的文件的部分代码的正确性，这对于大工程是很有帮助的，因为项目里有许多函数，如果一些函数的错误会给后面的开发带来很大的不方便。而如果要确认自己的函数是否正确，传统的方法就是写主函数然后编译运行，看输出的结果是否与正确的结果吻合，这样子显然很麻烦。有了Junit的帮助，就可以使用@before，@after，@test等元数据编写一个测试代码，从而检验自己的函数功能是否正确。

在例子中，我们都是使用命令行进行JUnit的测试，就要在命令行中输入：

javac –classpath .:junit-4.9.jar HelloWorldTest.java

java –classpath .:junit-4.9.jar –ea org.junit.runner.JUnitCore HelloWorldTest

使用这两句命令进行编译执行。运行后，如果函数的功能都正确，那么会返回测试通过的信息，并且会显示做出了几个测试。如果测试不通过，那会在错误提示中显示出函数运行结果与预想的不同的地方。

以后我们可以在Eclipse等IDE中进行JUnit测试，这样极大提高了效率以及代码的准确性，对写Java项目是很有帮助的。

实训刚刚开始，继续加油！O(∩\_∩)O