付出时间与努力，终可收获！

暨2015级软件工程中级实训总结报告

15331046 陈志扬 教务一班

从第四周到第十一周的实训今天终于宣告结束，虽然中间空了几周，但还是占了很多个周末。付出时间与努力，终可收获！谈到收获，我想可在报告中简单对每个阶段做些总结，下面我就简要谈谈我对每个阶段的思考。

第一阶段总结：

1. vi编辑器的使用：vi分为三种模式，分别是command mode，insert mode，last line mode。

我们在终端输入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。若只需保存只需按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。我们当然可以使用gedit或者sublime text来代替。通过这次vi的学习，我逐渐了解了vi，接下来还是需要多使用vim，特别要体验它的插件。

1. Java语言的学习及手动配置：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。

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。

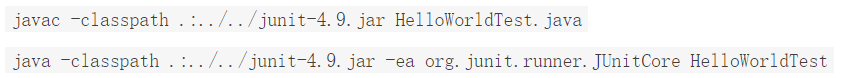
1. Ant配置和学习：关于Ant配置和学习，由于云桌面已经帮我们配置好，所以我是在虚拟机上配置的(云桌面有权限，不能修改profile，但好像可以修改bashrc来代替)。按照提供的教程，在官网上下载后解压到指定目录，设置系统环境变量(嗯，我又用到vi了，就是这样不断熟悉vi的)，重登系统，直接ant和ant -version测试是否安装成功。



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)。

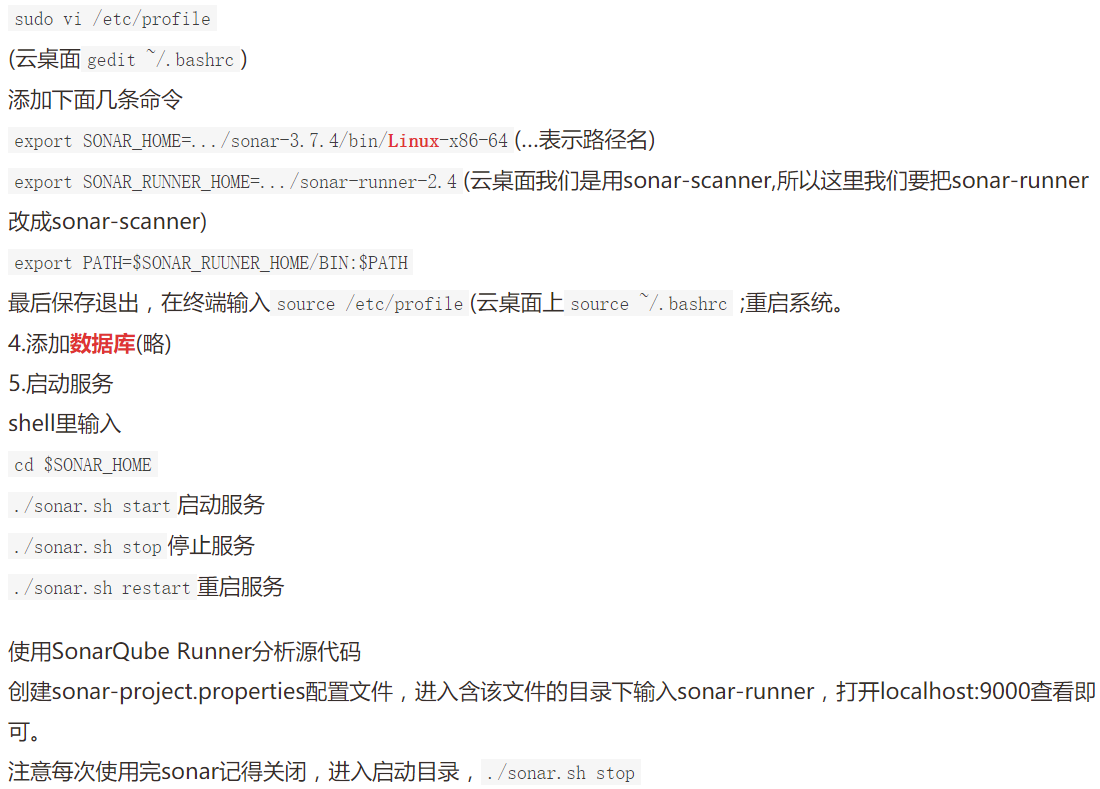
Ant是一个非常强大的工具，可实现项目的自动构建和部署等功能，类似于我们之前用过的makefile，通过这次简单地使用，对项目管理有一定的帮助，值得我们深入研究。

1. Junit的学习：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：



在运行需要引用jar包的程序时，需要使用上述命令行编译运行，其中-classpath可简写为-cp，若有多个jar包可用:号（冒号）隔开。

1. 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（注意拼写正确的路径名）  
   1.已安装[**Java**](http://lib.csdn.net/base/javase)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：





sonar-runner成功后主要检查几个方面：critical、major、minor、duplication、注释等等，然后逐一修改到满足要求即可，一般要求注释要超过10%，总评要高于60%，这个修改的过程也是极其艰难的，每一次在sonar方面都花了很长时间，其实这也是好的，代码规范的问题程序员有责！

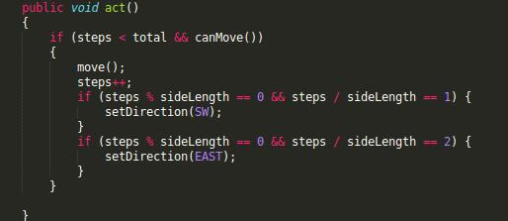
第二阶段总结：

1. 各个Bug的代码编写：这次的实训和初级实训最大的不同之处在于：中级实训只需要我们实现逻辑代码，整个框架都已经给出，所以相对而言也比较简单，我们在实现逻辑代码的过程中只需要理解每个api接口的作用，然后灵活运用api实现我们的代码。考察的是理解代码的能力以及能正确使用已实现的api接口。

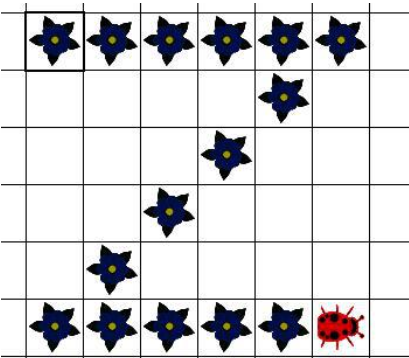
另外，需注意的一点是，各个bug之间的关系以及bug和actor的继承关系，例如CircleBug、Dancing和Bug类的继承关系：



各个bug之间也只有act函数不同，通过修改act函数可对每个bug定义不同的行为，例如ZBug的act函数如下：



运行结果如下：



1. Jumper类的设计：Jumper的设计需要参考part2的Runner类的写法，根据问题以及自己对actor的定义来编写，之前已提交过设计文档和测试文档，这里不做赘述。请TA谅解！
2. Critter类的设计：各个Critter类的设计思路和part2各个bug的设计思路大体相同，都是在继承Critter类然后定义makeMove、getActors、getMoveLocations这些函数，定义不同，实现不同的Critter。

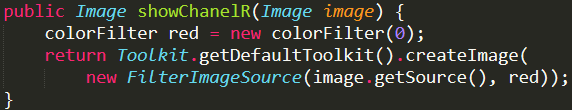
各个Critter类的不同之处在于：

* 1. 遇到不同的actor如何处理，例如RockHound遇到Rock是把Rock清除掉，而不是换方向。
  2. 每个Critter有不同的方向，例如Crab只有左右方向，而其他的有八个方向。
  3. 速度，例如QuickCrab每次移动两格，而其他是一格。

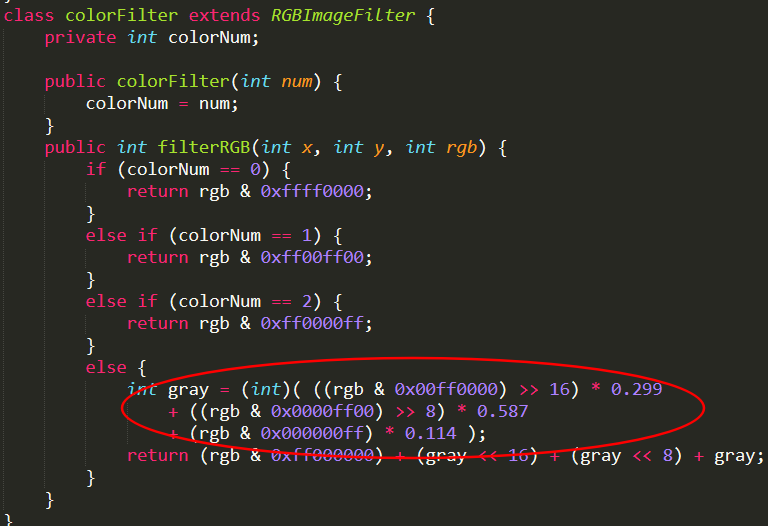
1. Grid类的设计：继承AbstractGrid，实现不同功能的Grid，主要问题是：
   1. 可用三种不同的方法：ArrayList、HashMap、TreeMap来实现每个Grid，区别好HashMap和TreeMap的用法，ArrayList使用较为简单。
   2. 编写UnBoundedGrid可参考设计文档中，大部分代码也已给出。

第三阶段总结：

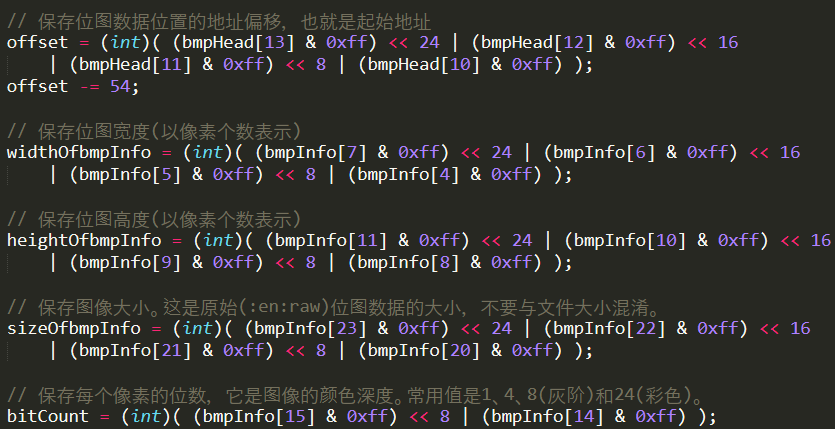
1. ImageProcessing：用二进制流读取图像，要注意各个api的调用方法，好在这些函数的用法很容易查阅到。每一种颜色的写法都是类似的，只要定义好一种颜色，其他两种颜色照搬即可。

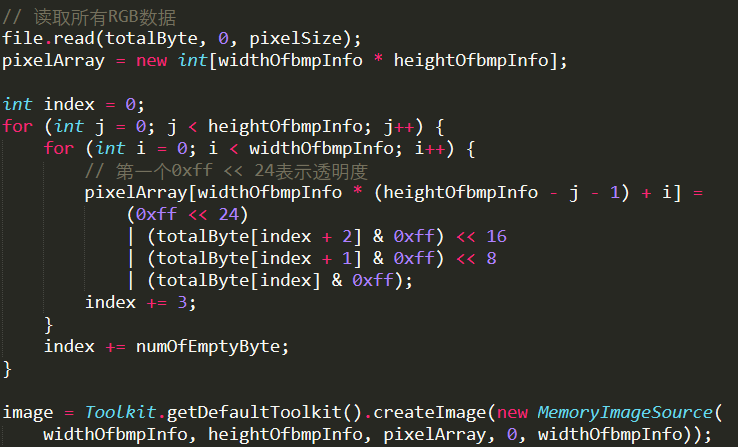


各个颜色的处理，圈起来的是灰度：



1. ImageIO：实训要求“读”不能使用api，所以只能通过位图信息来处理图像，这里简单截图来说明一下：





1. MazeBug的深度优先算法DFS：
   1. 先将树的所有节点标记为“未访问”状态。
   2. 输出起始节点，将起始节点标记为“已访问”状态。
   3. 将起始节点入栈。
   4. 当栈非空时重复执行以下步骤：
      1. 取当前栈顶节点
      2. 如果当前栈顶节点是结束节点（迷宫出口），输出该节点，结束搜索
      3. 如果当前栈顶节点存在“未访问”状态的邻接节点，则选择一个未访问节点，置为“已访问”状态，并将它入栈，继续步骤1
      4. 如果当前栈顶节点不存在“未访问”状态的邻接节点，则将栈顶节点出栈，继续步骤1.
2. Jigsaw的广度优先算法BFS：

BFS：从图中某节点V出发，在访问了V之后依次访问V的各个未曾访问过的邻接点，然后分别从这些邻接点出发依次访问它们的邻接点，并使"先被访问的节点的邻接点"先于"后被访问的节点的邻接点"被访问，直至图中所有已被访问的节点的邻接点都被访问到。如果还存在未被发现的节点，则选择其中一个作为源节点并重复以上过程。

BFS算法步骤：

* + 1. 将起始节点放入一个openList中。
    2. 如果openList为空，则搜索失败，问题无解；否则重复以下步骤：
       1. 访问openList中的第一个节点v，若v为目标节点，则搜索成功，退出
       2. 从openList中删除节点v，放入closeList中
       3. 将所有与v邻接且未曾被访问的节点放入openList中

BFS部分代码：



1. Jigsaw的启发式搜索算法：

与盲目搜索不同，启发式搜索（如A\*算法）利用问题拥有的启发信息来引导搜索，动态地确定搜索节点的排序，以达到减少搜索范围，降低问题复杂度的目的。在N-数码问题中，每搜索到每一个节点时，通过"估价函数"对该节点进行"评估"，然后优先访问"最优良"节点的邻接节点，能够大大减少求解的时间。

估价函数f(n)用来估计节点n的重要性，表示为：从起始节点经过节点n，到达目标节点的代价。f(n)越小，表示节点n越优良，应该优先访问它的邻接节点。Wiki上给了下面三种估价方法：

* + - 1. 所有放错码的数码个数
      2. 所有放错码的数码与正确位置的距离之和
      3. 后续节点不正确的数码个数

可同时使用多个估价方法，通过适当调整权重能够加快搜索速度。

启发式搜索算法步骤：

一．将起始节点放入一个openList中。

二．如果openList为空，则搜索失败，问题无解；否则重复以下步骤：

1.访问openList中的第一个节点v，若v为目标节点，则搜索成功，退出；

2.从openList中删除节点v，放入closeList中；

3.利用估价函数，将所有与v邻接且未曾被访问的节点进行估价，按照估价大小（小的在前）放入openList中

估价函数如下：



心得体会：中级实训确实比初级实训简单，主要原因是框架代码都已编写好。阶段一遇到的困难主要在配置环境上和vi编辑器的初体验，手动配置环境真的要注意很多问题；另外，sonar要改的问题也是很多，magic number、空格和tab键的替换、final class等等。阶段二编写逻辑代码，一旦理解了各个函数的功能作用，就显得简单了，特别是wiki上这部分还有很多问题需要回答，进一步理解了各个api接口。

收获：

1. 初探Java语言。
2. 能熟悉使用Eclipse。
3. 本实训涉及到的关于Linux环境下配置。
4. 提高了理解代码和利用api接口的能力。

最后，还是感叹一声：终于结束了！