

Jumper

程序说明

version 1.0.0

修订历史

日期	版本	作者	描述
2014.8.7	1.0.0	邝媛媛 刘建安	Jumper

目录

1	程序功能.....	3
2	实现过程.....	4
3	总结体会.....	5

1 程序功能

(1) Jumper 每次移动两格，移动后不留下任何痕迹。

(2) 当前面两格的格子为空时：

STATION	前面的格子	前面两格的格子	结果
1	Flower	空	move()
2	Rock	空	move()
3	Other actors	空	move()
4	空	空	move()

Jumper 将跳过任何障碍物。

(3) 当前面的格子为空前面两格的格子不为空时

STATION	前面的格子	前面两格的格子	结果
5	空	Flower	move()
6	空	Rock	turn()
7	空	Other actors	turn()

覆盖 Flower，其余转向。

(4) 当前面的格子或前面两格的格子不存在时

STATION	前面的格子	前面两格的格子	结果
8	不存在	不存在	turn()
9	任意	不存在	turn()

Jumper 将转向

附使用说明：

点击屏幕“step”使 Jumper 移动一次。

点击屏幕“run”使 Jumper 自动移动。

左右移动“slow—fast”可调整 Jumper “run”速度。

2 实现过程

(1) 经过判断, Jumper class 继承 Bug class.

(2) 写 Jumper 时, 重新写一个构造函数, 设置 Jumper 的默认颜色为橙色;

(3) 判断是否可以前进, 使其前进两格

由于 Jumper 每次是前进两格, 所以我们要判断前进方向的第二格是否为空, 或者有什么 Actor 占据着。

为了实现这个功能, 我们需重写 canMove 函数, 并且用两个变量来保存当前位置和要移动到的目的位置: oldLoc 和 newLoc;

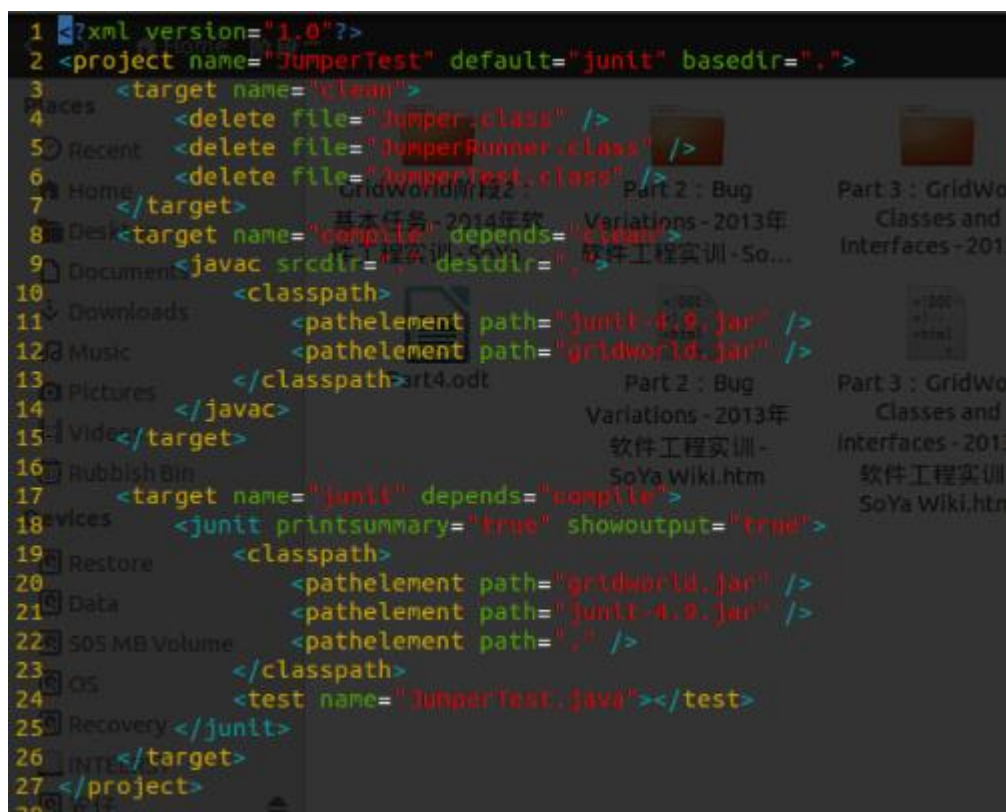
在 canMove 方法里面, 我们先取得当前位置 oldLoc 和当前方向 direction, 然后 oldLoc 调用两次 getAdjacentLocation 来取得在当前方向上的第二个, 并保存到 newLoc 中。接下来, 我们在对 newLoc 进行判断, 看其是否在 grid 内; 同时, 如果 newLoc 为空, 或者该位置被 Flower 占据着, 即表示 KingCrab 可以到移动到该位置; 否则则不能移动到该位置, 只能转向。

在调用 canMove 方法取得 newLoc 后, 我们在调用 move 方法来移动到新的位置或者调用 turn 方法来转向。

(4) 除了 canMove()需要修改, 其他部分基本可以与 Bug 一致。

(5) 最后写 JumperRunner.java。

(6)用 ant 构建程序:



```
1 <?xml version="1.0"?>
2 <project name="JumperTest" default="junit" basedir=".">
3   <target name="clean">
4     <delete file="Jumper.class" />
5     <delete file="JumperRunner.class" />
6     <delete file="JumperTest.class" />
7   </target>
8   <target name="compile" depends="clean">
9     <javac srcdir="src" destdir="build">
10       <classpath>
11         <pathelement path="junit-4.9.jar" />
12         <pathelement path="gridworld.jar" />
13       </classpath>
14     </javac>
15   </target>
16   <target name="junit" depends="compile">
17     <junit printsummary="true" showoutput="true">
18       <classpath>
19         <pathelement path="gridworld.jar" />
20         <pathelement path="junit-4.9.jar" />
21         <pathelement path="." />
22       </classpath>
23     <test name="JumperTest.java" />
24   </junit>
25 </target>
26 </project>
```

附：

/*Jumper.java 中函数 canMove()*/

```
public boolean canMove()
{
    Grid<Actor> gr = getGrid();
    if (null == gr)
        return false;
    //oldLoc表示当前所在位置
    oldLoc = getLocation();
    int direction = getDirection();
    //newLoc表示跳跃两个位置之后的位置
    newLoc = oldLoc.getAdjacentLocation(direction);
    newLoc = newLoc.getAdjacentLocation(direction);

    //如果要跳跃的位置不再grid内，就返回false
    if (!gr.isValid(newLoc))
        return false;
    //如果newLoc所在位置为空或者为flower，就返回true.
    Actor neighbor = gr.get(newLoc);
    return (neighbor == null) || (neighbor instanceof Flower);
}
```

3 总结体会

我们小组一共两人。虽然人少，但我们也按时完成了编码的要求以及报告的编写，这与我们每个组员的努力是密不可分的。

首先我们阅读了网站上 Part4 的内容，了解了编写代码的具体要求，最后由刘建安主刀，编写了这一块代码。然后经由两人测试，完成了整个程序的要求。

遇见的困难是多样的，百度与 TA 帮助了我们不少忙，虽然这份代码很小很少，但也是经过了一定努力的。

在编写的时候，大体方向我们处理的很好，在一些细节方面仍要改进。譬如说要加入判断 gr 变量是否为空，如果没有在代码中注意到这一点，在实际运行的时候就很有可能引发异常。同样的情况还有判断 jumper 要去的 Location 是否 valid, 如果为无效值，就应返回 false。

除了代码编写能力提高，对 linux 系统更加熟悉，更加熟练使用 java 编程以外，我们在对程序的整体把握上也有不少提高。对 java 的语法，使用 ant 构建项目，使用 Junit 测试程序，使用 sonar 检测程序，编写程序说明，测试报告等等，都使我们获得了长足的进步。

我们了解到了团队精神和协作能力的重要性，刘建安擅于编码，笔者更细心并优于文档编写能力，所以我们分工合作，取得了满意的结果。任何个人的力量都是有限的，当我们步入社会，做一个编码的独行侠听起来很帅，但缺乏团队协作能力，对大公司大工程来讲，这种人就是不合格的程序员。

虽然我们此刻写的是一个小小的程序，但管中窥豹，见微知著，麻雀虽小，五脏俱全，

一个优秀的程序是不能缺少良好的文档说明的。这种文档习惯也会让我们以后受益匪浅。一个程序缺乏文档，就如同一份代码没有注释，失去了灵魂，难以被后人掌控。缺乏文档，一个软件系统就缺乏了生命力，在未来的查错，升级等等时就会遇到极大的麻烦。

Sonar 测试是一个有效的帮助我们规范化代码的测试，它涵盖面广，检查仔细。一个代码写的别人都看不懂的程序员，不会是一个合格的程序员，规范化编写这是程序员的基本要求，也是真正拥有职业素养的第一步。

由此，我们由衷感谢这次锻炼机会，感谢组员们的辛勤奉献。