程序说明
version 1.0.0

修订历史

日期	版本	作者	描述
2015.08.21	1.0.0	郑越升 龙嘉伟 黄勇进 林倩文	Jumper 的实现

目录

1 程序功能

功能	实现函数	说明
构造 Jumper	Jumper() Jumper(Color jumperColor)	无
行动一次	void act()	若 canMove, 前面两格是花或者 Bug, Jumper 可以覆盖(吃掉)它们, 若是 Actor, 则直接撞飞一格; 若!canMove, 则按照指定的角度旋转
向右转 45 度	void turn()	无
移动(跳)	void move()	跳到前面两格的位置
判断能否移动	boolean canMove()	判断的是: 1. 前面一格是不是花/石头/空,若是,则进行 2, 否则返回 false 2. 满足 1 情况下,判断前面两格是不是石头/Jumper,若是则返回 false,否则返回 true

2 实现过程

1. turnDegree 和 meetActor 属性:

在 Jumper 类中定义两个新属性，在 canMove()函数中，用来记录需要旋转的角度以及两格外所遇到的 Actor:

```
private int turnDegree = 0;

private Actor meetActor = null;
```

2. act()函数中，如果可以移动,对两格之外的 Actor 进行类型判断:

```
if (canMove()) {
    // 如果两格之外的 Actor 是 Bug 或者 Flower,吃掉吃掉!
    if (meetActor instanceof Bug || meetActor instanceof Flower) {
        meetActor.removeSelfFromGrid();
    } else if (meetActor instanceof Actor) {
        // 如果是 Actor, 则撞飞撞飞!
        Location loc = meetActor.getLocation();
        int d = getLocation().getDirectionToward(loc);
        meetActor.moveTo(loc.getAdjacentLocation(d));
    }
    move();
} else{
```

```
    // 如果不能移动, 则旋转一定角度, 可能是 90, 可能是 45
    setDirection(getDirection() + turnDegree);
}
```

3. canMove()函数中，先对前面一格进行判断，在对前面两格进行判断：

第一格的判断代码:

```
Location loc = getLocation();
Location next = loc.getAdjacentLocation(getDirection());
// 如果位置不合法(边界),返回 false, 设置 turnDegree 为 Location.RIGHT.
if (!gr.isValid(next)) {
    turnDegree = Location.RIGHT;
    return false;
}
//如果 actor 不是花,石头,或者空的话
//返回 false, 设置 turnDegree 为 Location.HALF_RIGHT.
Actor nextActor = gr.get(next);
if (!(nextActor instanceof Flower ||
        nextActor instanceof Rock || nextActor == null)){
    turnDegree = Location.HALF_RIGHT;
    return false;
}
```

第二格的判断代码:

```
Location next2 = next.getAdjacentLocation(getDirection());
// 如果位置不合法(边界),返回 false, 设置 turnDegree 为 Location.RIGHT.
if (!gr.isValid(next2)) {
    turnDegree = Location.RIGHT;
    return false;
}
Actor next2Actor = gr.get(next2);
// 如果 next2 的 actor 是石头或者 Jumper 的话.返回 false;
if (next2Actor instanceof Rock || next2Actor instanceof Jumper) {
    turnDegree = Location.HALF_RIGHT;
    return false;
}
//存储第二格的 Actor
meetActor = next2Actor;
```

3 总结体会

心得:

Jumper 的基本实现并不难, 只需要对前面两格的顺序判断即可实现.在对 Jumper 基本功能进行扩充的时候,小组经过讨论,决定了一些 Jumper 的特性,比如可以吃掉 Flower 和 Bug, 以及当遇到 Actor 的时候, 选择将它撞飞一格, 并且经过讨论, 认为遇到墙的时候应该

旋转 90 度而不是 45 度, 这样对旋转的分类提高了 Jumper 的'智力'.

程序使用 Ant 进行构建,代码完成后经过了 Sonar 的测试(90 分)以及 Junit 的单元测试.

总的来说,第一次小组合作十分顺利, 分工明确, 各司其职, 同时学到了不少新的知识.