开发文档

沈征宇 18302010009

**程序结构设计与分析**:任务刚布置下来时,我还没有多少**面向对象**思维,仍然在使用**面向过程**思路,发现设计过程非常复杂，需要完备地考虑各种可能发生的情况，即开始之前就设计好整个过程，这**极大地挑战**了我的思考能力，最终以失败告终。咨询了**张星宇**学长之后，他为我提出了“先去自学面向对象吧”的建议。我接受了这个建议，开始了面向对象的自学。对开拓思路有了很大帮助。比如：将“方块”设计成**抽象类**，各种形状的方块设置为其**子类**，当前的方块，即将到来的方块都是一个**对象**。它的行列、小方块分布、旋转方法都存储在这个对象之中，成为这个对象的属性。将“游戏”设计成**类**，其中有地图，它以“|”为左右墙。“—”为下底，中间以“ ”先行填充。在Java程序设计的过程中，我一边学习，一边改进自己的代码，进行了**多次重构**，取得了比较好的效果。比如，学习了“**数组**”，我把俄罗斯方块存为数组而非手绘数组，学习了“**方法**”，我将方块的各种动作抽象为方法，大大方便了程序的编写，使其看起来**精简**的多，学习了“**包**”、“**继承**”、“**多态**”，我将各种类型的方块存到一个包中，并编写一“抽象类”方块类，其繁复多样的子类（I方块，J方块，O方块等等）**继承于**抽象类“方块”，在方块子类中重写父类旋转方法以**适应**不同种类方块的旋转方式。

主要函数的功能：

**Main函数**：游戏运行

**getRandomBlock函数**：获得一个七种方块之中任意类型的方块对象。

**nextBlock函数**：为对象“nextBlock”赋当前位置，方块类型。

**Print函数：**根据当前地图的情况（堆积起来的方块的情况），现在正在掉落的方块的情况（形状、坐标），生成字符串，返回一个字符串，在Main方法中直接打印。

**executeCommand函数**：从键盘接受指令，再执行相应的操作。

**Collide函数**：返回一个布尔类型值，它表现了在进行了某个操作之后，是否现在的方块会和墙、旧方块发生碰撞。

**reachTheBottom函数**：判断是否方块已到底，如果到底，就可以把它固定到地图里了。

**LoadBlock函数**：当方块已无法再降，将他装到地图里，成为地图的“元素”。

**Eliminate函数**：消除已满的行，为玩家加分。

**goLeft,goRight,goDown,goBottom函数**：方块的移动操作。

**printRanking函数**：知晓本局的分数，用户名之后，与历史战绩比较，插入，打印。

**nextRotatedCells函数**：返回一个当前方块旋转后的方块，但对当前方块不进行任何改动。

**Rotate函数**：对当前方块进行真正的旋转。

**getCells函数**：获得当前的方块，这是为了更好的封装性。

编程中遇到的问题和解决⽅法：

1. 在判断旋转后是否会碰撞的时候，不能真去旋转，较好的解决办法是造一个旋转后的方块而不去改变现有的方块，利用这个被造出来的方块来判断是否碰撞，如果不碰撞，就可以放心地去真正地旋转方块了。
2. 拓展部分要求实现排行榜，我自学了文件I/O的知识，以txt文件保存了排行榜的信息，但在“读取路径”这个问题上，我犯了难，当我把项目发送给助教之后，该文件的绝对路径是不一样的，所以，我学习了相对路径的知识，解决了问题，我也了解到，相对路径其实也是绝对路径，只是编译器帮我们在底层添加了上几层的目录。
3. 在装载方块进入地图数组的时候，因为我采用了4\*4的容器来装方块（3\*3容器更简单，但长条只能实现三格，故不采用），极有可能容器底部一行没有方块，因此触底的一行实际上是中间的两行之一，那么装载时，触底那一行被装到了地图数组底部，触底行的下一行就被装载到地图下面，这会造成数组越界。我的办法是：先判断方块最底行在哪一行，然后从上向下装载方块，最底行还底的行不进行装载，完美解决。
4. 有些常被使用来显示的文字常量经常出现，在源代码中显得冗余，因此我新建一个“Word”类，专门用来存放会被用到的话语，使代码简洁美观。

进阶部分说明：

一：排行榜：我的排行榜在游戏结束后显示，自动显示前10名分数。

二：存档&读档：我开发了支持多玩家，多存档位置的存读档方式，多名玩家可以使用存档功能，每个玩家可以有多个存档位置自行选择，且这些存档一直存在，不会被销毁。

三：创意：道具系统（combo使用，连击奖励）