

**实验报告**

|  |  |
| --- | --- |
| **课程名称**: | 面向对象高级编程 |
| **开课学期**: | 2023-2024学年 第1学期 |
| **专业**: | 软件工程 |
| **班级年级**: | 2022级软工中外01 |
| **学生姓名**: | 吴孜远 |
| **学号:** | 222022321062009 |
| **实验教师:** | 王晓蒙 |

**计算机与信息科学学院 软件学院**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | | 实验6 对象集合 | | | |
| 实验时间 | |  | Type | \*验证性 □设计性 □综合性 | |
| 需求描述：设计一个游戏场景，支持地图漫游和多个物体移动操作。。   1. **调试程序，正常运行**        1. **完善除GUI层以外的程序类图，包括类及关系**      1. **分析程序中应用的Set、List、Map等集合接口和类是否合适，简单阐述**   Set被用于储存已经执行过的命令，作为一个临时的集合，set是最简单的，没有必要用List等集合，但是注意这里的removeAll只能接受collection，所以必须是collection大类，数组不行。  ArrayList被用于储存ball和待执行的command，它作为一种动态数组，可以动态开辟内存空间，这一点对于不知数量的ball和command刚好合适。但是在本实验中，我认为使用List的其他两种实现也未尝不可。使用List作为它们的通用接口类型，这就允许根据需要选择不同的List实现，而不影响代码的其余部分。  HashMap用于储存地块的状态值，key是坐标编码，value是状态值，这种两个值一一对应的情况最适合用Map来解决，只用一个集合就说明了坐标和状态值的关系。   1. **分析Move命令对象中为什么要重写Object.equals()函数**   在controller类中有一个接收GUI层移动命令的moveTo函数，其中要判断commands列表中是否contain接收的命令，若contain，则替换；若没有，则add进去。这里的contain函数在java中是这样描述的：  boolean [java](eclipse-javadoc:%E2%98%82=JavaLearning/D:%5C/program%20files%5C/JAVA%5C/jre%5C/lib%5C/rt.jar=/javadoc_location=/https:%5C/%5C/docs.oracle.com%5C/javase%5C/8%5C/docs%5C/api%5C/=/%3Cjava).[util](eclipse-javadoc:%E2%98%82=JavaLearning/D:%5C/program%20files%5C/JAVA%5C/jre%5C/lib%5C/rt.jar=/javadoc_location=/https:%5C/%5C/docs.oracle.com%5C/javase%5C/8%5C/docs%5C/api%5C/=/%3Cjava.util).[List](eclipse-javadoc:%E2%98%82=JavaLearning/D:%5C/program%20files%5C/JAVA%5C/jre%5C/lib%5C/rt.jar=/javadoc_location=/https:%5C/%5C/docs.oracle.com%5C/javase%5C/8%5C/docs%5C/api%5C/=/%3Cjava.util(List.class%E2%98%83List).contains([Object](eclipse-javadoc:%E2%98%82=JavaLearning/D:%5C/program%20files%5C/JAVA%5C/jre%5C/lib%5C/rt.jar=/javadoc_location=/https:%5C/%5C/docs.oracle.com%5C/javase%5C/8%5C/docs%5C/api%5C/=/%3Cjava.util(List.class%E2%98%83List~contains~Ljava.lang.Object;%E2%98%82java.lang.Object) o)  Returns true if this collection contains the specified element. More formally, returns true if and only if this collection contains at least one element e such that (o==null ? e==null : o.equals(e)).  这里使用了Object大类中的equals函数，传入的object不为null时就判断List中是否存在e与它equal。  默认情况下，equals 方法在Object类中的实现是比较对象的引用相等性，即比较两个对象是否是同一对象（内存地址相同）。  在我们的语境下，一个ball不会同时有两个move命令，即使这两个move命令的target不相同，所以默认的判断equals（判断相等）的方式在这里不可行。所以要在Move类中重写Object.equals()函数，使其判断当两个move命令的ball相同时就返回true，避免了一个ball同时接收两个不同move命令的问题。若不这样写，两个不同的move命令会同时存在，每一次单步执行游戏都要将commands中的所有命令都执行，这样就会使一个ball在某一时间同时向两个不同的方向移动，这是不可以的。  另外，HashSet为了避免重复元素出现，也会进行equals判断。   1. **尝试改进程序，包括设计的合理性或增加新功能**   我是想将蓝色区块定义为水，这样球不能生成在水上，不能移动到水上。要让球的目标不是水很简单，直接不调用移动命令就行；让球不生成在水上，也简单；但是写完这两个之后我发现我们球的移动方式是很傻的，就是单纯先走x方向再走y方向，如果让球一碰到水就停下那移动起来很困难。最好是能让球利用回溯算法寻找路径，但是想了想还是太复杂了，复习期末更为重要，日后一定实现。（对不起）  以下是我已经改过的部分。（蓝色图是map2，对应的地块状态值就是2）  这是防止移动目标在水上：    这是让球随机生成且不在水上：（我觉得这挺好）     1. **自学命令模式，加深对设计的理解**   我感觉命令模式就是将命令的发出接受过程涉及到的每一个对象都抽象出来，有命令本身，命令的发出者，命令的接收者。命令本身是个接口，可以被实现为多个具体操作，而命令被发出者所声明，并传递给接收者。  在本实验中，移动是命令的具体实现，命令的接收者是ball，调用者是Controller，接收者是球，这样使得球能够移动。 | | | | | |
|  | | | | | |
|  | | | | | |
| 教师评阅 | 程序完整，执行正确（40%）： | | | |  |
| 设计逻辑合理，类图正确（40%）： | | | |  |
| 代码、文档格式规范（20%）： | | | |  |
| 实验成绩： | | | | |