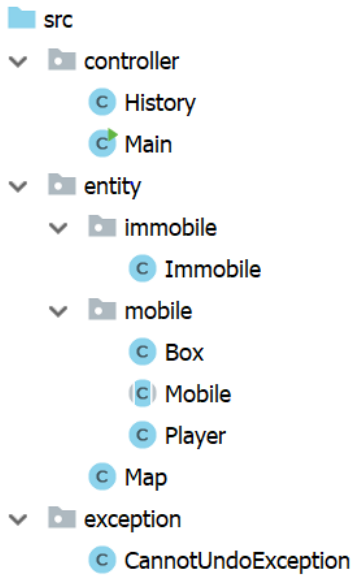
推箱子说明文档

一．代码的总体结构及设计思路：

代码总体结构分为三部分，controller、entity、exception，如下图所示。



1.controller 包括 Main：游戏的控制逻辑和 History：游戏的历史记录，用于悔棋和重新开始。

2.entity 包括推箱子游戏中的所有实体。首先，是一个 Map，而地图又由各个实体构成。根据实体是否可以移动，又分为 immobile 和 mobile。不可移动的包括空地、墙壁、目标点这 3 种，可移动的有箱子和玩家，而箱子和玩家有很多类似的属性和方法，所以定义一个 Mobile 类（抽象类），箱子和玩家均继承 Mobile，并实现 Mobile 中的 public abstract boolean canMove(Direction direction);方法。

3.exception 中定义了“已经回到初始状态，不能悔棋”的异常。

二．各个对象的设计思路和说明：

1.Immobile 类：拥有确定位置的行、列属性，自身类型的 ImmobileType 属性和位于该 Immobile 对象上的 Mobile 属性（null、Player、Box）。其中，ImmobileType 是一个枚举类型，共有 3 种类型：BLANK 空地, WALL 墙壁和 TARGET 目标点。重写 toString 方法用来打印不可移动的实体。

2.Mobile 类：有行、列属性，用以确定和修改可动对象的位置。实现 move 方法，一个可移动对象移动的完成由 3 部分组成：set 该对象的位置，将该对象之前所处位置的 Immobile 对象的 Mobile 属性设为 null，将该对象移动到的位置的 Immobile 对象的 Mobile 属性设置为该对象。canMove 方法为抽象方法，需要 Box 和 Player 类实现。

3.Box 类：实现 canMove 方法，思路：箱子移动一定是由玩家进行推动，所以箱子移动的判断条件是移动方向（即对应的目的地）是否为墙壁和箱子，若是则不能移动，否则可以移动。对于 Box 类，还定义一个 isInTarget 方法，用来判断该对象是否在目标点，用于游戏胜利的判断和游戏地图的打印，重写 toString 方法用来打印箱子。

4.Player 类：对于 Player 类，定义一个 Object getBeside(Direction direction)方法，用来获得玩家将要移动到的位置（对应的 Immobile 对象）的 Mobile 属性。该方法会在 main 方法中进行玩家和箱子移动时被调用：若 player.getBeside(direction)返回 Box 对象，则能否移动取决于该 Box 对象能否移

动，即 box.canMove(direction)；否则，能否移动取决于该 player 能否移动，即 player.canMove(direction)。

实现 canMove 方法，思路：由于 Player 的 canMove 方法只有在玩家移动时不进行推箱子操作才会被调用（即 player.getBeside(direction)返回的不是 Box 对象），所以玩家能否移动只需考虑移动到的位置是否有墙壁。

5.Map 类：采用单例模式。属性为可移动对象数组 mobiles、不可移动对象二维数组 immobiles（元素为空地、目标点和墙壁）和一个静态的 map 对象。通过提供 public static void initMap(File file)方法，可以让外部调用 initMap 方法来初始化/修改 map 对象；通过提供 public static Map getMap()方法，可以让外部调用 getMap 方法获取 map 对象。实现 public boolean checkWin()方法，通过比较在目标点的箱子数量是否等于箱子总数来判断本场游戏是否胜利。

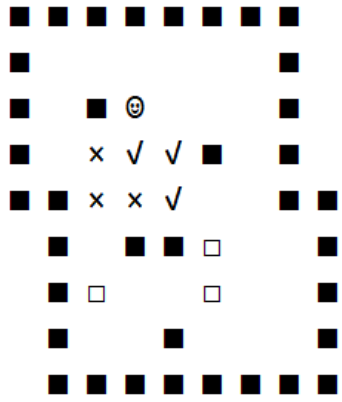
6.History 类：在 History 类中定义了两个内部类：HistoryRecord 类和 Node 类。HistoryRecord 类记录了一个可移动对象（Box 或 Player）的移动轨迹，即包含以下 3 个属性：可移动对象、移出位置对应的 Immobile 对象、移入位置对应的 Immobile 对象。定义 Node 类是为了构建一个单向的链表结构（每一个 Node 可以访问前一个 Node）。因为 History 类只需要实现撤销移动和重新开始操作，所以每一个 Node 只需要得到前一个 Node 对象。Node 类包含 2 个属性：移动记录 HistoryRecord 属性和前一个 Node 对象。

History 类采用单例模式，包含一个表示当前结点的 Node 属性（current）和一个静态的 history 对象，外部可通过 getHistory 方法得到 history 实例。void restart()方法实现：将 history 设为 null。void addRecord(Immobile from, Immobile to, Mobile mobileInstance)方法实现：创建一个新 Node，并将 current 属性指向这个新 Node。在实际调用 addRecord 方法时，若只有 Player 进行了移动则直接将其进行记录，若 Player 和旁边的 Box 均进行移动，则先增加 Box 的移动记录再增加 Player 的移动记录。（对应 undo 方法先撤销 Player 再撤销 Box）

为了实现 void undo()方法，定义一个辅助的撤销方法 void undoHelper()。undoHelper 方法将 current 结点记录的移动（Box 或是 Player）进行回退，并更新 current。undo 方法的实现：首先判断是否能够进行撤销操作，如果不能则抛出 CannotUndoException，如果可以则继续进行撤销。若 current 前一个 Node 记录的是 Box 的移动，则调用两次 undoHelper 方法，否则调用一次 undoHelper 方法。

7.Main: 负责与用户进行交互和逻辑控制，即完成移动、回退、重新开始某一个地图、退出。定义一个枚举类型 Direction，并实现 Direction parseDirection(String direction)方法，将用户输入的方向转换成对应的上下左右。

三．关于地图界面的说明：



使用实心方块代表墙壁，空心方块代表箱子，×代表目标点，√代表箱子在目标点，笑脸代表人物当前位置。