UCAS-大富翁

2019K8009937003

> 面向对象程序设计课程大作业——第二阶段报告

序言

在第一阶段的报告中，笔者已经对整个游戏的设定以及拟实现的功能进行了分析，而根据面向对象的设计思想，接下来就要具体对项目中拟实现的各个类以及它们之间的关联进行分析。故这一阶段的报告主要针对第一阶段中介绍的功能，分析类的设计与关联。

整体框架

现阶段拟采用的框架为经典的 MVC 架构模式，MVC 即 Model-View-Controller（模型-视图-控制器），它可以将设计中模型与用户界面的代码分离，并通过控制器来保证两者的统一。类的分析将分别从这三个方面切入。

面向对象设计思路

在设计类时，其内部的属性都被封装了起来，对外只提供一些方法来访问或修改这些属性，使得操作简单；而对于一些具有共性的类（如建筑类和卡牌类），实现了一些抽象类以及接口，这些类通过继承或实现它们完成相应的功能，能在一定程度上简化了代码，增强代码的可扩展性。具体设计思路见下文。

类/接口的设计

以下是项目中拟设计的各个类分析，其中的属性和方法都不代表最终版本。

模型

1. **计时类 Tick**

属性	备注
protected long **curTick**	当前的tick
protected long **startTick**	事件开始的tick
protected long **nextTick**	事件结束的tick

方法	备注
public void **setCurTick**(long curTick)	设置当前tick
public long **getCurTick**()	获取当前tick
public void **setStartTick**(long startTick)	设置事件开始tick
public long **getStartTick**()	获取事件开始tick
public void **setNextTick**(long endTick)	设置事件结束tick
public long **getNextTick**()	获取事件结束tick

p.s. 该类可以作为所有游戏对象的父类，通过继承它可以记录时间以及一个事件的持续时间

2. **接口 Port**

抽象方法	备注
public abstract void **update**(long tick)	模型更新
public abstract void **gameInit**()	游戏初始化

3. **玩家模型 PlayerModel （extend Timer implement Port）**

属性	备注
private String **name**	玩家姓名
private int **x**	当前坐标X
private int **y**	当前坐标Y
private int **cash**	当前持有的现金
private int **deposit**	当前存款
private List **buildings**	玩家拥有的房屋
private List **cards**	玩家拥有的卡片
private int **hospitalRemain**	剩余住院天数
private int **prisonRemain**	剩余坐牢天数
private Control **control**	总控制器

方法	备注
public **PlayerModel**(Control control)	构造方法，初始化控制器
public void **setName**(String name)	设置玩家名字
public String **getName**()	获取玩家名字
public void **setCash**(int cash)	设置玩家现金
public int **getCash**()	获取玩家现金
public void **setDeposit**(int deposit)	设置玩家存款
public int **getDeposit**()	获取玩家存款
public void **setX**(int x)	设置坐标X
public int **getX**()	获取坐标X
public void **setY**(int x)	设置坐标Y
public int **getY**()	获取坐标Y
public void **setHospitalRemain**(int hospitalRemain)	设置玩家住院天数
public int **getHospitalRemain**()	获取玩家住院天数
public void **setPrisonRemain**(int PrisonRemain)	设置玩家入狱天数
public int **getPrisonRemain**()	获取玩家入狱天数
public void **setBuildings**(List cards)	设置玩家房屋
public void **setCards**(List cards)	设置玩家卡片
public List **getBuildings**()	获取玩家房屋
public List **getCards**()	获取玩家卡片
public void **gameInit**()	玩家初始化(implement Port)
public void **update**(long tick)	更新玩家目前时间，并根据当前时间、开始时间以及结束时间触发相应的控制器操作(implement Port)

4. **地图模型 LandModel（extend Timer implement Port）**

属性	备注
protected int[][] **land**	二维数组，存储整个地图信息（即每格分别为什么）
public final static int **NTH** = 1	<空>（未设置）
public final static int **ORIGIN** = 2	起点
public final static int **SHOP** = 3	商店
public final static int **QUESTION** = 4	问号
public final static int **BANK** = 5	银行
public final static int **HOSPITAL** = 6	医院
public final static int **PRISON** = 7	监狱
public final static int **CANTEEN** = 8	食堂（地产）
public final static int **JXL** = 9	教学楼（地产）
public final static int **RWL** = 10	人文楼（地产）
public final static int **DORM** = 11	X公寓（地产）
public final static int **LIBRARY** = 12	图书馆（地产）
public final static int **JTJS** = 13	阶梯教室（地产）
public final static int **HALL** = 14	礼堂（地产）
public final static int **BATH** = 15	澡堂（地产）

方法	备注
public int[][] **getLand**()	获取地图
public int **matchLand** (PlayerModel player)	匹配地图事件
public void **gameInit**()	初始化地图信息(implement Port)
public void **update**(long tick)	更新地图目前时间(implement Port)

5. **骰子模型 DiceModel（extend Timer implement Port）**

属性	备注
private int **point**	骰子点数
private Image[] **dicePoints**	骰子点数图像
private int **diceState**	骰子当前状态（是否可以使用）
private GameRunning **running**	运转控制器

方法	备注
public **DiceModel**(GameRunning running)	构造方法，初始化运转控制器
public int **getPoint**()	获取当前骰子点数
public void **setPoint**(int point)	设置骰子点数
public void **setDiceState**(int diceState)	设置骰子状态
public int **getDiceState**()	获取当前骰子状态
public Image[] **getDicePoints**()	获取当前骰子图片
public void **gameInit**()	初始化骰子信息(implement Port)
public void **update**(long tick)	更新骰子目前时间(implement Port)

6. **全局建筑 BuildingsModel（extend Timer implement Port）**

属性	备注
private List buildings	全局建筑
private LandModel land	地图模型

方法	备注
public **BuildingsModel**(LandModel land)	构造方法，初始化地图模型
public List **getBuilding**()	获取全局建筑链表
public Building **getBuilding**(int x,int y)	获取当前坐标建筑
public void **gameInit**()	设置地图中需要加入建筑的格子(implement Port)
public void **update**(long tick)	更新目前时间(implement Port)

7. **建筑抽象类 Building**

属性	备注
protected PlayerModel **owner**	建筑拥有者
protected String **name**	建筑名称
protected boolean **purchasability**	是否可被购买
protected int **price**	购买价格
protected int **rent**	租金（过路费）
protected int **level**	建筑等级
protected int **maxLevel**	最大等级
protected int **posX**	坐标x
protected int **posY**	坐标y

（抽象）方法	备注
public **Building**(int posX, int posY)	构造方法，初始化建筑坐标
public void **setPurchasability**(boolean purchasability)	设置是否可被购买
public boolean **isPurchasability**()	查询是否可被购买
public boolean **canUpLevel**()	查询是否可以升级
public PlayerModel **getOwner**()	获取拥有者
public void **setOwner**(PlayerModel owner)	设置拥有者
public int **getLevel**()	获取建筑等级
public void **setLevel**(int level)	设置建筑等级
public int **getPrice**()	获取建筑价格
public int **getRent**()	获取租金
public abstract int **getEvent()**	获取建筑事件
public abstract int **passEvent**()	获取经过建筑事件

p.s. 该类是所有建筑类的父类，具体细分的建筑类（如地产，商店，医院，监狱等）略

8. **卡片抽象类 Card**

属性	备注
protected PlayerModel **owner**	卡片拥有者
protected String **name**	卡片名称
protected Image **img**	卡片图片
protected PlayerModel **target**;	卡片作用对象
protected int **price**	卡片价格

（抽象）方法	备注
public **Card**(PlayerModel owner)	构造方法，初始化卡片拥有者
public abstract int **useCard**()	使用卡片效果
public PlayerModel **getOwner**()	获取拥有者
public void **setOwner**(PlayerModel owner)	设置拥有者
public int **getName**()	获取卡片名字
public void **setName**(String name)	设置卡片名字
public PlayerModel **getTarget**()	获取卡片作用对象
public void **setTarget**(PlayerModel target)	设置卡片作用对象

p.s. 该类是所有卡片类的父类，具体细分的卡片类（如控骰卡，转向卡等）略

视图

有关图形用户界面的实现，笔者目前还在学习当中，现阶段拟采用 java swing 与 java awt 实现，故这里先简单列出拟实现的图形界面类的代表

1. **游戏主窗口 JFrameGame（extend JFrame）**

采用 JFrame 提供的方法设置窗口名字，大小，标签等，同时用来读取用户输入（包括鼠标与键盘）信息

2. **绘制层抽象类 Layer（extend JPanel）**

作为游戏中所有需要刷新的图层类（如建筑更新、骰子刷新、玩家位置、玩家信息等）的父类

控制器

其中的运转控制器主要负责游戏状态的转换（利用了经典的状态机）

1. **游戏总控制器 Control**

属性	备注
private List **players**	所有玩家
private BuildingsModel **building**	全局建筑
private LandModel **land**	地图
private DiceModel **dice**	骰子
private Timer **gameTimer**	全局计时器
private GameRunning **run**	游戏运转控制

方法	备注
public **Control**()	构造方法，初始化游戏对象并将玩家模型加入游戏状态中
private void **initClass**()	实例化所有模型类
private void **createGameTimer**()	创建游戏计时器
public void **pressDice**()	按下掷骰后，设置骰子对象点数并根据点数移动玩家
public void **movePlayer**()	移动玩家
public void **passBuilding**()	中途路过建筑事件处理
public void **playerStop**()	玩家移动完毕，停下操作处理
<一系列对于停下操作处理的方法，如购地、交租、随机事件等>	

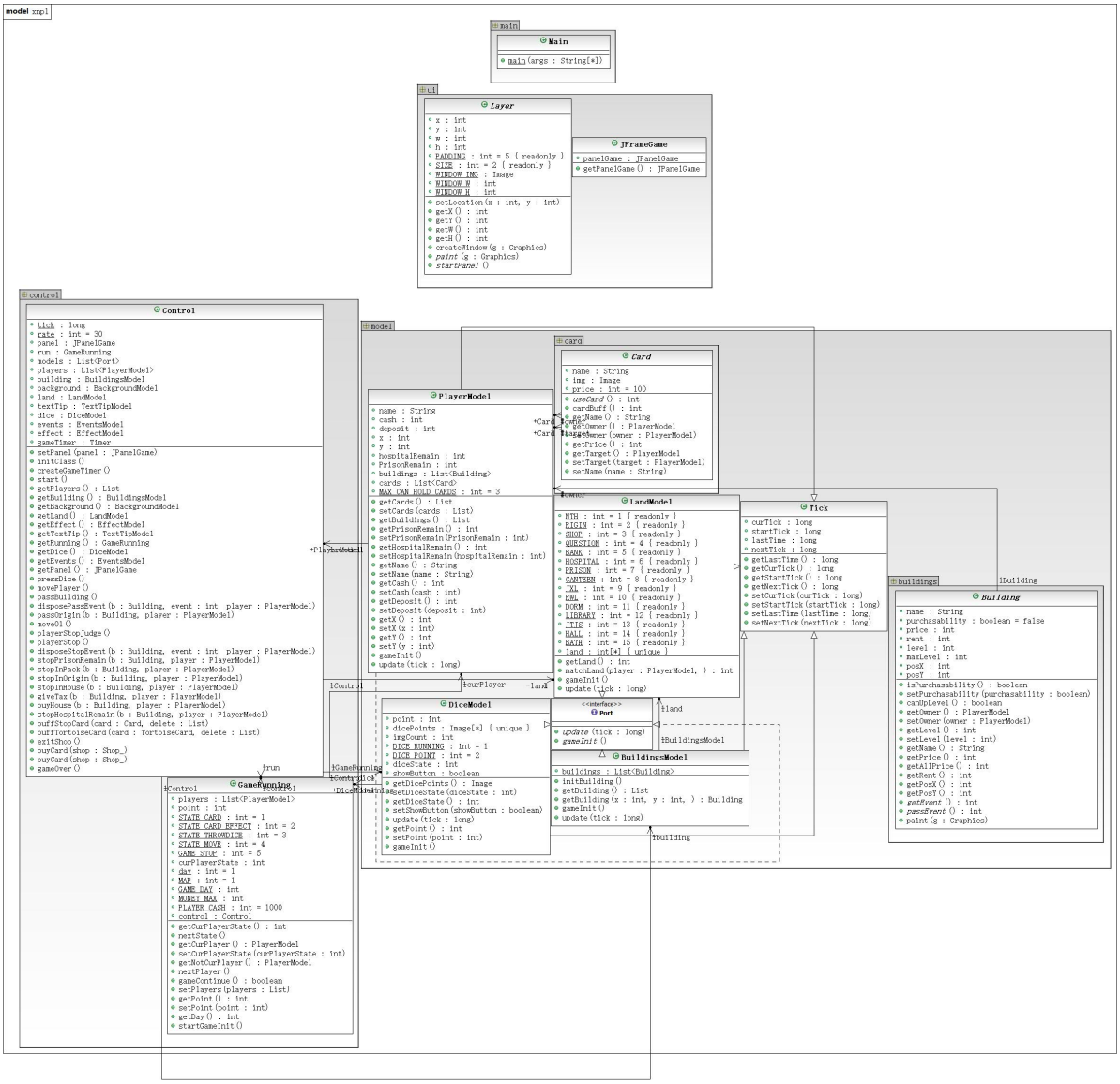
2. 运转控制器 GameRunning

属性	备注
private List **players**	所有玩家
private PlayerModel **curPlayer**	当前操作玩家
private int **curState**	游戏状态
<一系列状态定义，包括掷骰、移动、使用卡片等>	
private Control **control**	控制器

方法	备注
public **GameRunning**(Control control, List players)	构造方法，初始化玩家链表以及控制器
public void **nextState**()	根据转换当前状态，进入下一状态
public void **nextPlayer**()	交换玩家
public void **movePlayer**()	移动玩家
public boolean **gameContinue**()	判断游戏是否继续（结束条件是否成立）

类之间的关联

现阶段的UML类图如下（尚未完善，原图见同目录文件）



可以看到，控制器与绝大多数模型都存在关联关系，而模型类之间也有一定的关联或依赖，同时有两个类作为所有模型类（除建筑和卡片抽象类）的父类。由于现阶段还未实现视图端，故尚无法体现出 MVC 架构中三个层次之间的关系。