## 基于Java Swing的大富翁游戏设计与开发【1】

## **一、开发需求**

1、游戏人数：2人

2、游戏胜利条件：一名玩家破产[1]，则另一名玩家获胜。

3、游戏准备：游戏角色分为两个，第一个角色姓名为：Player1第二个角色姓名为：Player2。

4、游戏规则：

轮到自己回合时，按照以下流程进行游戏，直到有人达到胜利条件，游戏立即结束。

3.1掷骰子：需要根据骰子的点数减去相应的金钱数。

5、概念解释：

[1]破产：角色持有的金钱数量小于等于0。

## **二、需求分析**

在需求分析中，我们仅仅关注游戏本身而不是软件。首先，定义了大富翁游戏中的重要概念，分析了该游戏的游戏规则；然后从整体上关注游戏，着重讨论了该游戏的交互、用例以及状态等关键内容，为后面的软件概要设计、详细设计以及代码实现做好准备。

**1、概念定义 (分析存在哪些实体类)**

角色(Player)：游戏中代表角色的虚拟人物，每个角色有自己的金钱。

骰子（Dice）:玩家需要点击骰子来决定扣除的金钱数量。

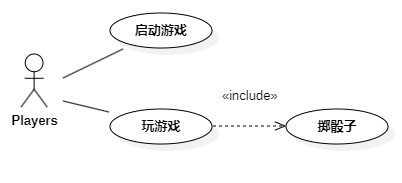
**2、游戏规则 (定义控制器的逻辑)**

这里需要详细分析大富翁的游戏规则，如下：

2.1 游戏开始时，玩家都拥有相同的金钱数量：100。且玩家1先进行投掷.

2.2 游戏时，玩家需要点击骰子来扣除相应的金钱，扣除是金钱数量为：投掷出的骰子点数\*10.

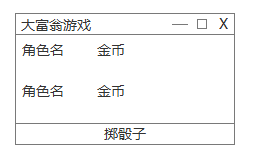
**3、系统用例 (定义功能，分为若干函数)**



图片1 大富翁游戏用例

## **三、界面原型设计**

设计的界面如图片2所示，该界面由Swing组件构成，角色1的角色名与金钱数在上方，角色2的角色名与金钱数在下方。通过点击掷骰子按钮可以减少角色的金钱。



图片2 界面原型设计

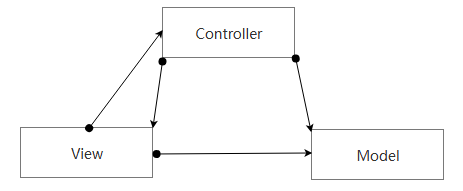
## **四、软件设计**

**1、MVC模式简介**

在软件设计开发中，存在一种将软件分层的设计架构——MVC模式，MVC模式将软件体系分为三层，第一层是View层，View负责展现数据，最直观解释View层—用户所看到的界面。

第二层是Model层，Model负责存储数据的对象，当然必要的时候，它也可以带有逻辑。第三层是Control层，Control层相当于View层与Model层的桥梁，当用户与View层进行交互要更改Model层的数据时，通过调用Control层中封装的函数来进行更改，如此实现视图View层与实体Model层的数据分离[4]。

三者的关系如下所示：Controller需要持有View层与Model层，因为Controller需要进行数据的更新，View层需要持有Controller来对Model层进行数据更新，而View指向Model层的箭头代表了View层需要持有Model层[5]。



图片3 MVC设计模式

**2、软件架构设计**

下面的架构设计非常重要，Main代表启动类，软件的View层、Control层与Model层在Main进行组装，GameFrame与Control的连线代表GameFrame与Control相互持有，Control指向Game的箭头代表Control持有Game。

注意，软件架构设计并非一成不变，在之后，为了实现更加好看的界面，我们会对软件架构作出改变。

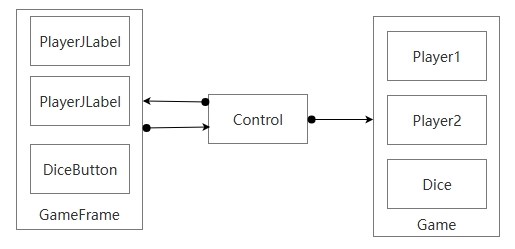


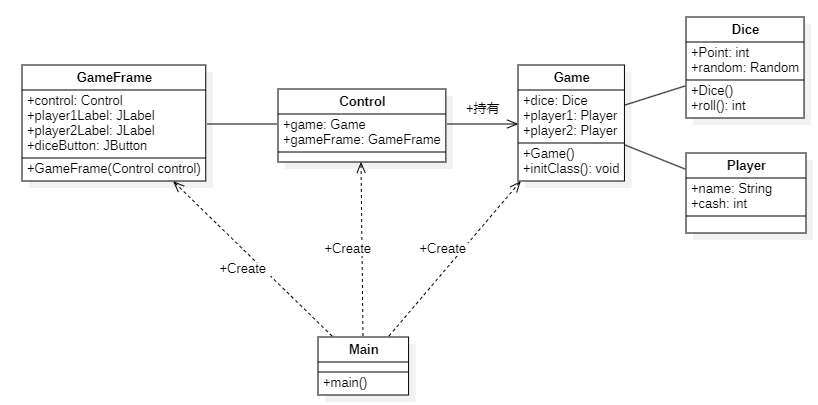
图4 软件架构设计

**3、功能实现**

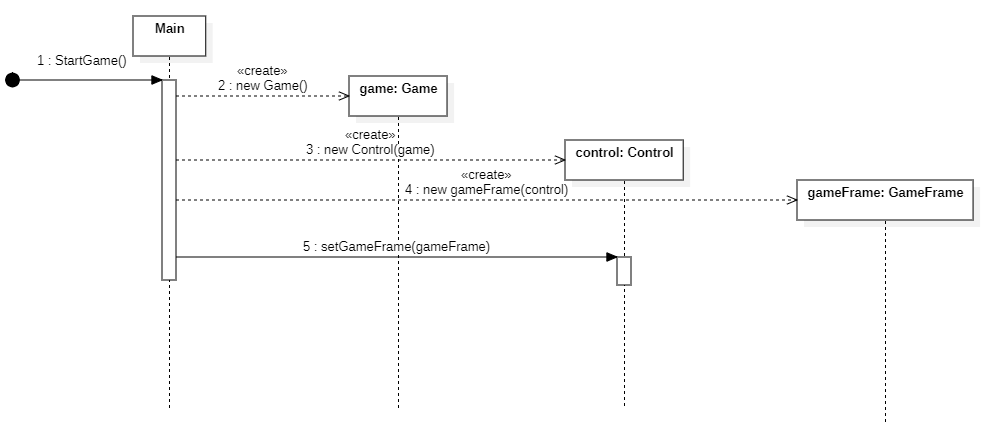
#### 3.1 启动游戏功能

用户启动游戏，生成界面原型中的界面，界面生成使用JavaSwing中的组件。

**3.1.1 类图**



**3.1.2 功能时序图**

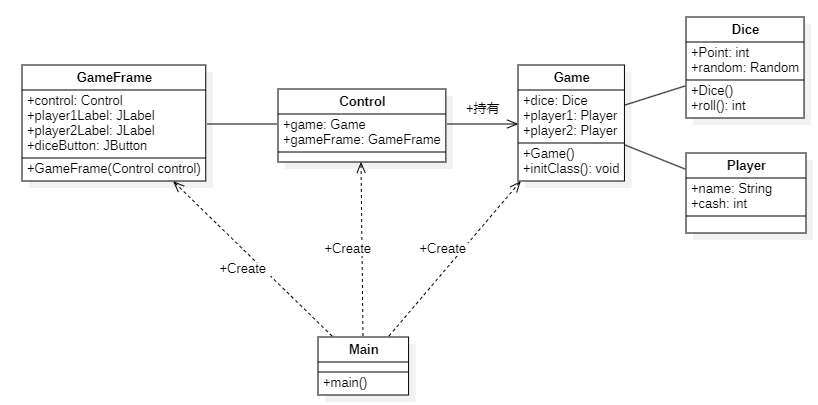


**3.1.3 代码实现**

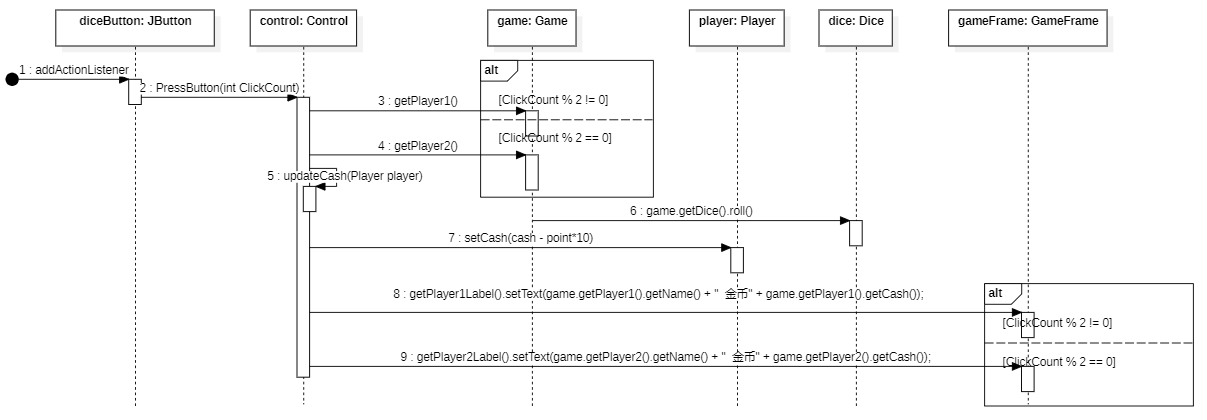


#### **3.2 投掷骰子功能**

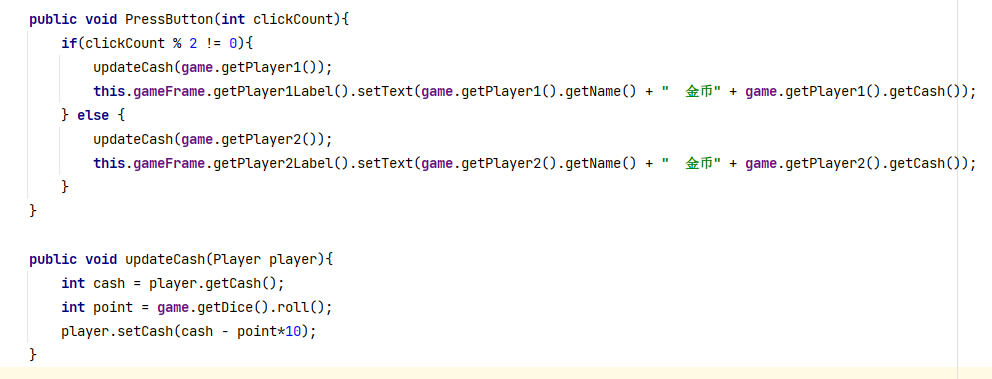
**3.2.1 类图**



**3.2.2 功能时序图**



**3.2.3 代码实现**

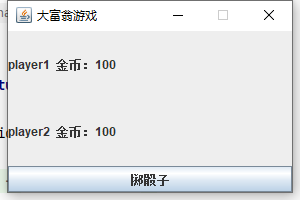


**4、概念解释**

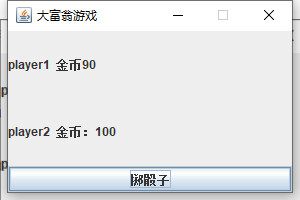
[4]:为什么要对实体层与视图层进行模型分离呢？ 大家可以思考一下。

[5]:View需要取得Model层的数据来进行展示。

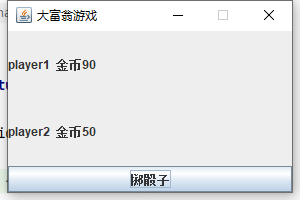
## **五、实际效果演示**



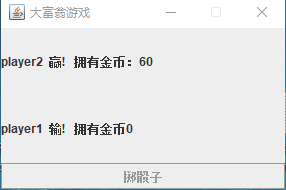
初始界面



点击一次之后 Player1金币减少



点击两次之后 player2金币减少



游戏结束

## **下个目标**

1. 增加图片效果，使得页面更加好看。
2. 增加游戏地图以及格子的概念，丰富游戏玩法。