

# 2014 级面向对象程序设计 Project2

--说明文档

14302010031 陆周涛

## 一、 图形界面设计

### 1. 欢迎页面 start pane

主要用于显示该游戏的关键信息（标题、版本号、确认进入游戏的选项）

#### i. 封面（显示游戏的名字和开始按钮）

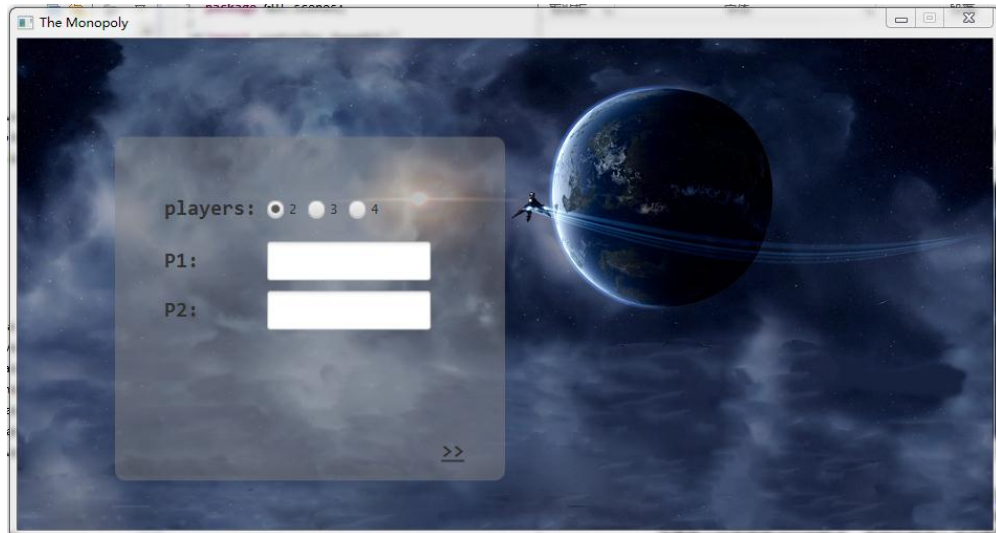


点击 start 按钮，即可进入玩家信息设置界面。（如下）

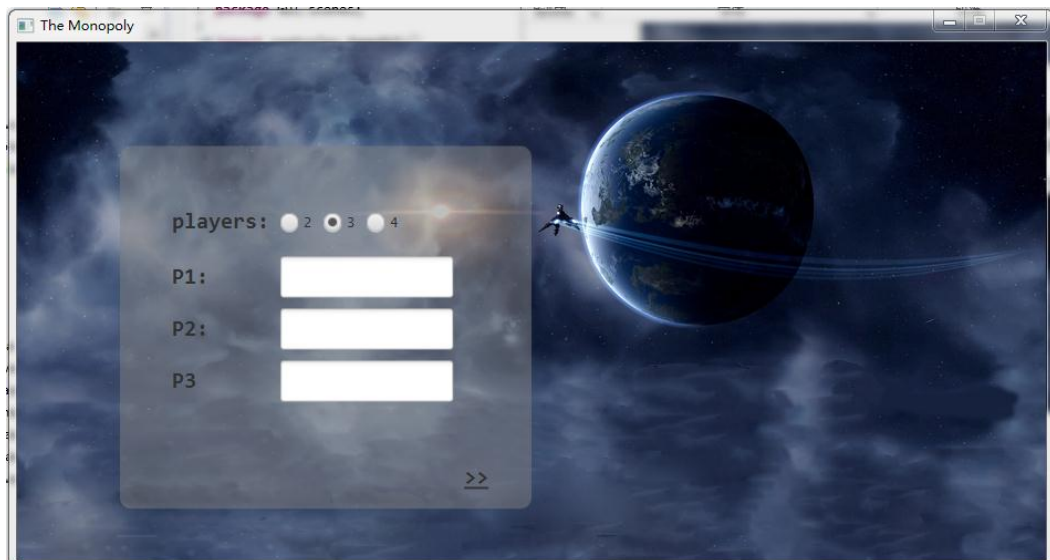
#### ii. 设置玩家

主要用于设置各个玩家的姓名，首先是四个单选框用于确定玩家的人数，姓名输入框会随设置的玩家人数动态显示。“>>”按钮表示确认输入，此时对于输入的要求是每个玩家都有对应的姓名，否则不能进入游戏界面。

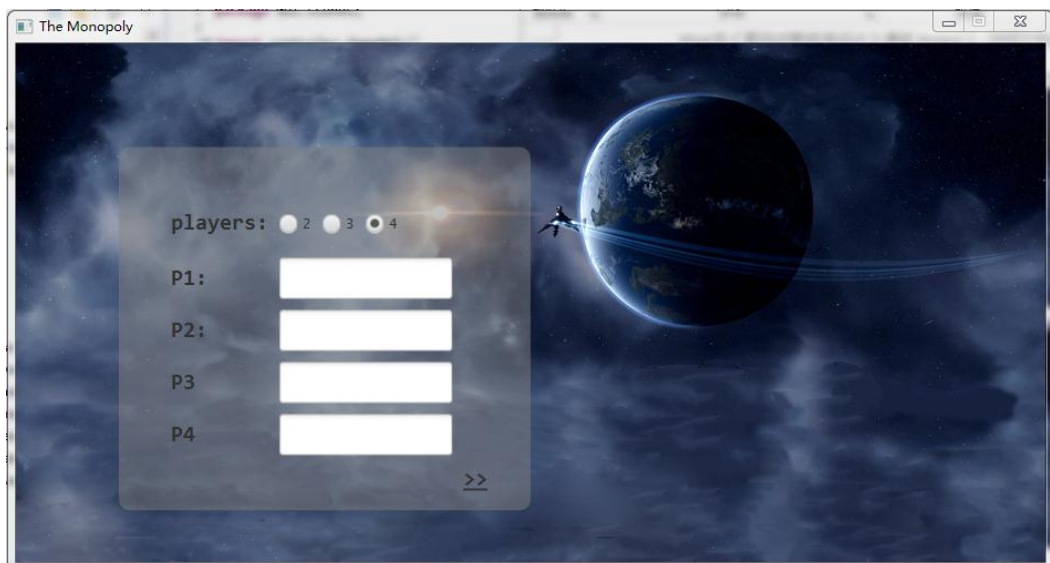
（两个玩家）



(三个玩家)



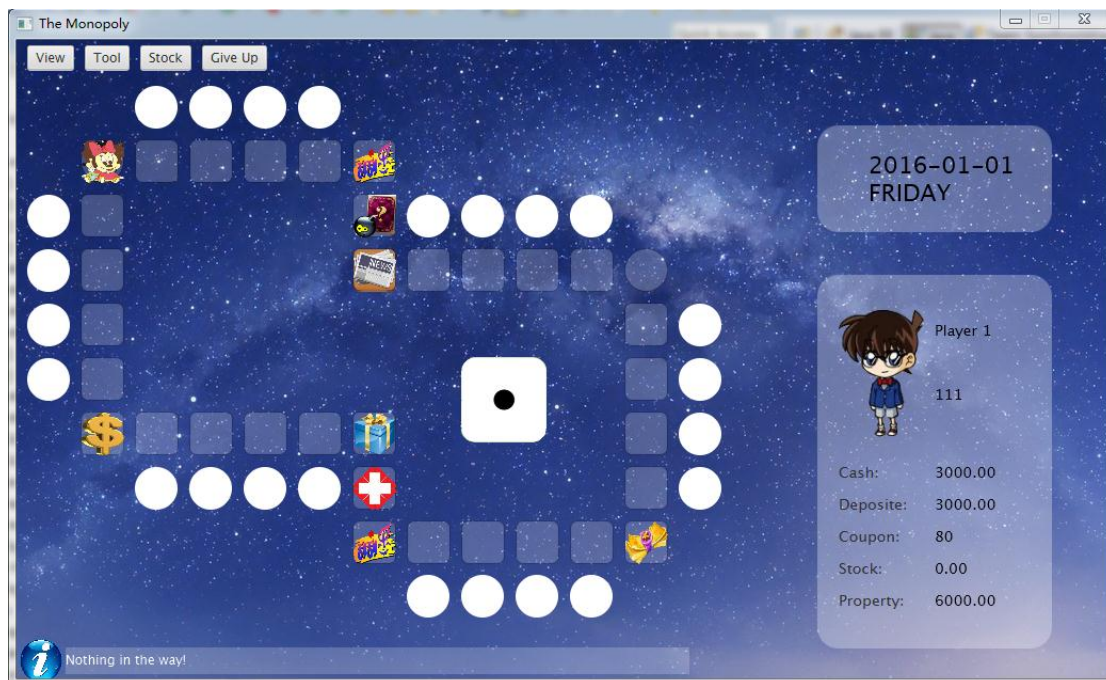
(四个玩家)



填写对应玩家数量的名字，并点击">>", 即可进入游戏界面。

## 2. 游戏界面 play pane

整个游戏界面主要分为如下 5 个部分，整体继承自 HBox 水平盒模型，左边部分是一个垂直和模型（VBox）从上到下有菜单面板、地图面板、消息面板，右边也是垂直盒模型从上到下有时间面板、玩家面板。



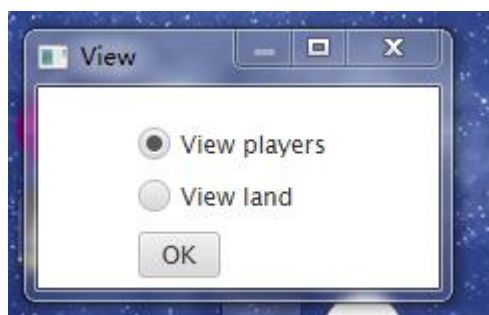
（当前可点击的区域有骰子区域以及菜单区的四个按钮）

### i. 菜单 menu pan



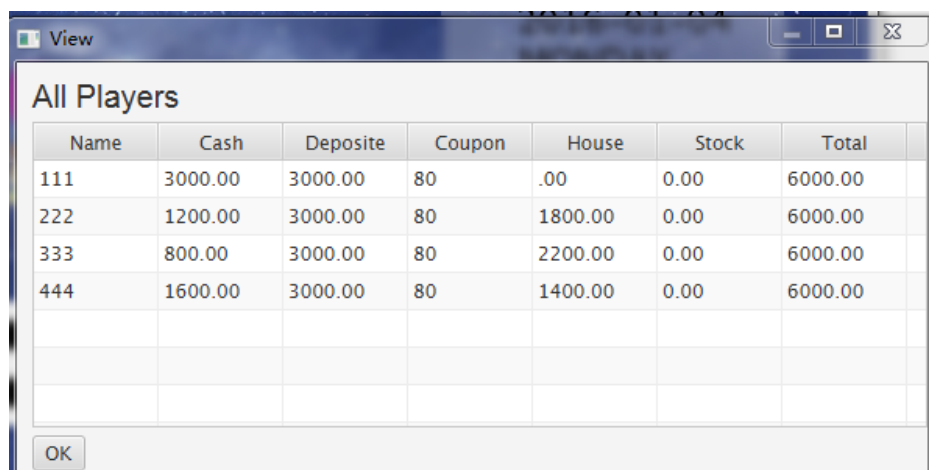
菜单栏共有四个按钮，分别对应于 part1 命令行版本的菜单中的“查看玩家信息”、“查看指定步数信息”、“使用道具”、“进入股市”、“认输”等命令。

a) View 按钮包括查看玩家信息和查看地图信息。



若选择查看玩家的信息，即会显示对应的玩家信息表格。



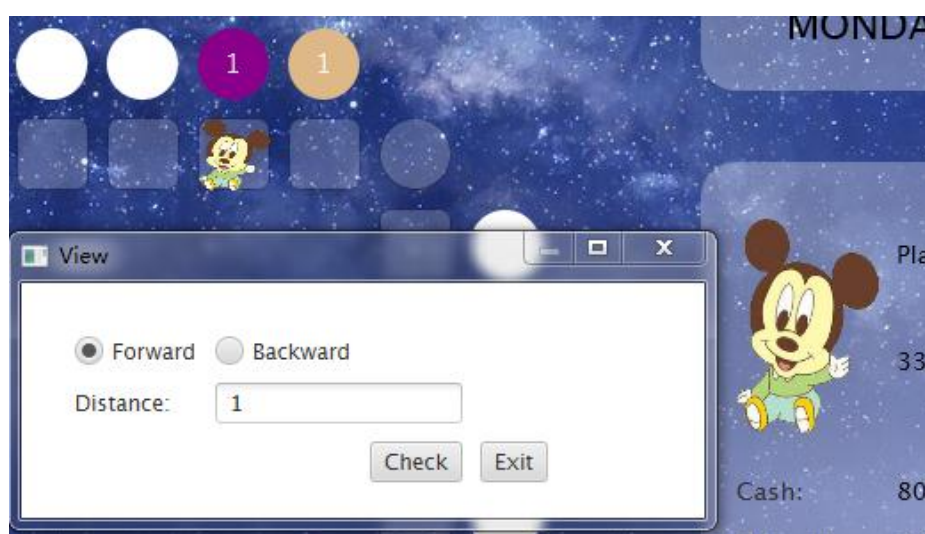


The screenshot shows a window titled "View" with a sub-header "All Players". It contains a table with the following data:

Name	Cash	Deposite	Coupon	House	Stock	Total
111	3000.00	3000.00	80	.00	0.00	6000.00
222	1200.00	3000.00	80	1800.00	0.00	6000.00
333	800.00	3000.00	80	2200.00	0.00	6000.00
444	1600.00	3000.00	80	1400.00	0.00	6000.00

At the bottom of the window is an "OK" button.

若选择查看地图信息，还需要确定查看的具体位置。

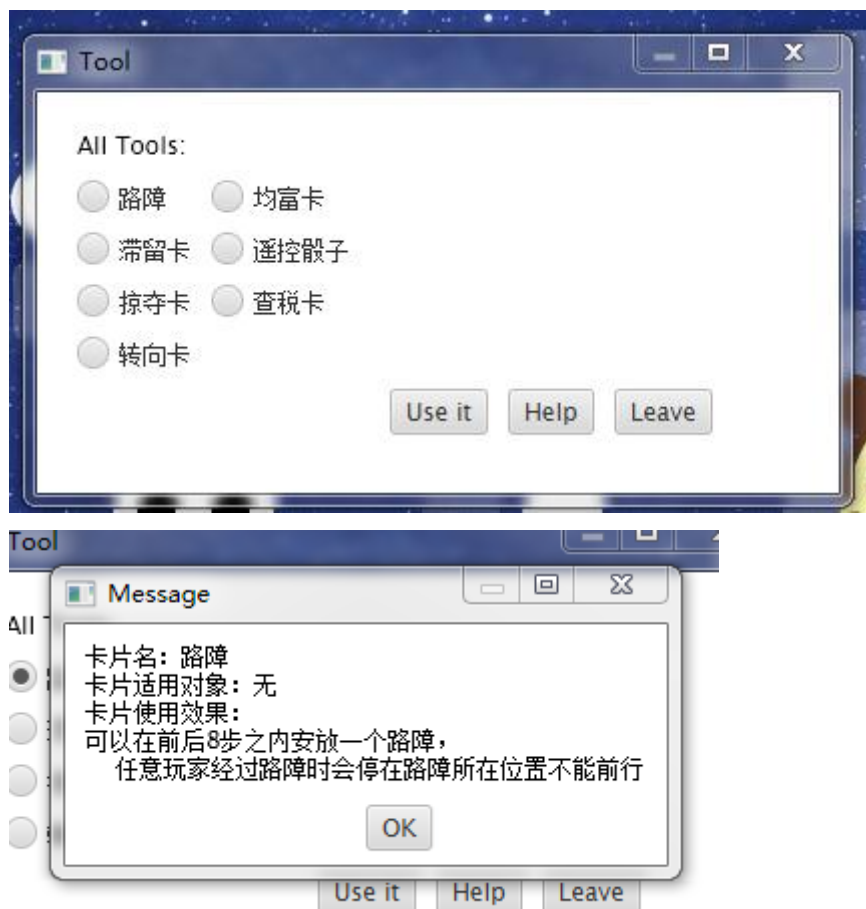


点击 check 即会出现相应位置信息的窗口。



- b) Tool 按钮用于使用道具，选择想要使用的道具，并确定使用即进入相应道具的响应界面，进行相应的处理（有些道具直接发动不需另外再设置）；同时也

提供了 help 选项，可以查看道具的具体信息（名称，使用限制，使用效果等）



- c) Stock 按钮用于进入股市（这里没有实现 bonus 部分的折线图）。进入股市之后双击相应的股票即可进入购买或者售出的界面进行相应的交易。Ok 按钮用于退出股市。

股票显示信息如下：

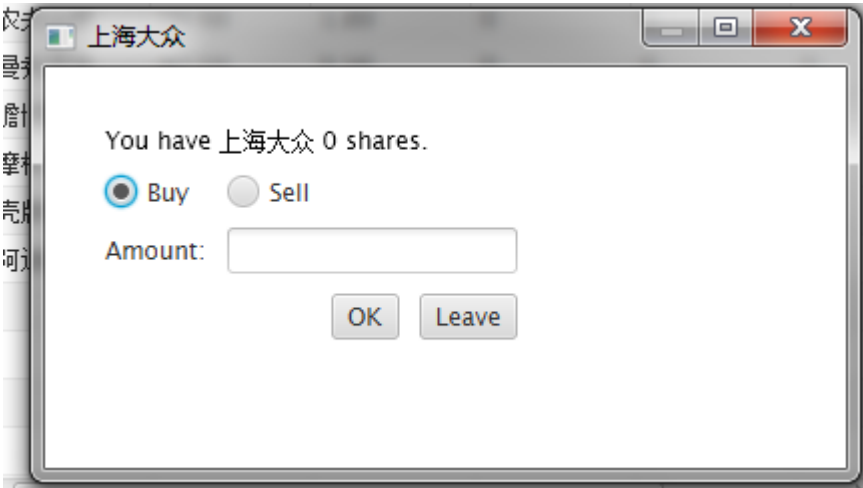


The screenshot shows a window titled "Stock" with a table titled "All Stocks". The table has six columns: Name, Price, Scope, 111 owns, 222 owns, and 333. The data is as follows:

Name	Price	Scope	111 owns	222 owns	333
壳牌石油	38.58	1.45	0	0	0
上海大众	142.19	-0.38	0	0	0
菲诗小铺	26.2	4.14	0	0	0
联合利华	17.71	-1.53	0	0	0
广州恒大	101.21	2.02	0	0	0
农夫山泉	56.98	1.89	0	0	0
曼秀雷敦	87.09	0.16	0	0	0
瞻博网络	34.25	-8.4	0	0	0
摩根大通	28.16	-1.87	0	0	0
壳牌石油	38.58	-2.85	0	0	0
阿迪达斯	112.77	-7.41	0	0	0

Below the table is a horizontal scrollbar and an "OK" button.

双击之后，进入选择业务界面。



The screenshot shows a dialog window titled "上海大众" (Shanghai Volkswagen). It contains the text "You have 上海大众 0 shares." and two radio buttons: "Buy" (selected) and "Sell". Below these is a text field labeled "Amount:". At the bottom are "OK" and "Leave" buttons.

输入相应数量并选择业务，确认后即可完成交易。

- d) **Give up** 按钮用于确认退出游戏。退出之后玩家将从活跃玩家的列表中移除，并将游戏的控制权转移到下一个玩家（若只剩下意味玩家则游戏结束），但是地图依然显示玩家的最后位置。

## ii. 地图 map pane



地图界面整体是一个 **grid pane**，每一个 **cell** 各自是一个 **stack pane**，可以类似栈的方式在容器中添加组件，这些组件包括每一个位置图形化显示，玩家的图片等。从而可以实现玩家在不同位置之间的移动。

玩家可以点击骰子的区域，从而进行移动，到达相应的位置便会发生相应的响应。响应完成之后就自动将游戏的控制权转移到下一个玩家。

## iii. 信息提示 message pane



信息框主要用于提示玩家前方 10 步以内可能遇到的路障。相当于 **part1** 菜单中“前方 10 步预警”命令。如果存在路障则会显示“Attention! A barrier is in 3 steps before!”。否则便显示“Nothing in the way!”。

信息提示框会随着玩家设置路障或者游戏控制权转换而发生更新。

iv. 时间 time pane



当每一个玩家完成一次移动之后，就会自动更新时间信息。

v. 玩家信息 player pane



用于展示当前游戏控制玩家的基本信息，包括姓名、现金、存款、点券、股票和总资产，其中总资产是现金、存款和房产的总和，不包括股票。

玩家信息的更新基本是玩家发生信息发生变化时立即更新。因此可以随时看到相关信息的变更。

## 二、 代码改动和新增（benefit from OO）

1. 新增的【医院】功能。



对于这个新增的功能，主要需要解决的问题包括将医院点添加到地图上，在新闻点的响应事件中增加【关进医院】的新闻，以及控制玩家在两回合之内不能移动。

- 首先，对于将医院点添加到地图中，由于之前在设计地图的属性时，在每个 cell 中存放的地点类是以基类的声明类型（即 Land）存储的，因此对于 cell 无需做任何改动。只要修改最原始的字符地图，并添加继承自 Land 类的 Hospital 类，以及增加 Landtype 中的枚举类型，即可完成显示时的修改。这里用到的 OO 思想主要是继承的抽象封装，以及通过继承实现的多态。
- 在新闻事件中增加【关进医院】的新闻，并进行具体的操作。由于在玩家类中已经提供了设置玩家所在 cell 以及移动方向的接口。因此当玩家确认之后就可以调用接口设置 cell 到医院，并将运动方向设成逆时针。这里用到的 OO 思想主要是 Player 类对修改玩家属性的封装和隐藏。
- 最后，是如何控制玩家两回合内不能移动。我在 player 类中添加了用于标识需要等待回合数的属性。每次在将控制权换到下一个玩家的时候，检查玩家是否还有需要等待的回合数，若不为 0 则将等待回合数减 1 并直接将控制权再移给下一个玩家，若为 0 则允许玩家移动。

修改或增加的代码如下：

增加医院的枚举类型：

```

2
3 public enum LandType {
4     house,
5     propShop,
6     bank,
7     news,
8     cardGift,
9     emptyLand,
0     couponGift,
1     lottery,
2     hospital;
3
4 public static LandType getLandType(char c){
5     switch (c){
6         case 't': return LandType.house;
7         case 'c': return LandType.lottery;
8         case 'q': return LandType.couponGift;
9         case 'x': return LandType.news;
0         case 'k': return LandType.emptyLand;
1         case 'y': return LandType.bank;
2         case 'd': return LandType.propShop;
3         case 'n': return LandType.cardGift;
4         case 'h': return LandType.hospital;
5         default: return null;
6     }
7 }
8 }
9

```

增加医院类（代码结构）

```

1 package entity.land;
2
3 import entity.map.Cell;
4
16
17 public class Hospital extends Land{
18
19     /*
20      * GUI version
21      */
22     public Hospital(Cell cell,GameGUI game) {
23         super(cell,LandType.hospital,game);
24         name = "爱心医院";
25         this.image = new ImageView(new Image("file:icons/hospital.gif"));
26     }
27
28     @Override
29     public void _response(Player controlPlayer) {
30         Game.getGame().getMenuController().go_on();
31     }
32
33     /*
34      * general part
35      */
36     @Override
37     public String getDiscription() {
38         return "类型: 医院\n"
39             + "名称: "+name;
40     }
41 }
42

```

添加新闻事件

```

        _cardQuit(); break;
    case 5:
        _putIntoHospital(controlPlayer); break;
    }
}

private void _putIntoHospital(Player cp) {
    MessageDialog md = new MessageDialog(game.getStage(),"Sorry to inform you that you need
    md.getBtn().addEventHandler(ActionEvent.ACTION, new EventHandler<ActionEvent>(){
        @Override
        public void handle(ActionEvent arg0) {
            cp.setCell(cell.getMap().getHospital());
            cp.setWaitingRound(2);
            cp.setDirection(Direction.anticlockwise);
            Game.getGame().getMenuController().go_on();
        }
    });
}
}

```

确定是否移动（Player 类中）

```

    public boolean checkNeedWaiting(){
        if(this.waitingRound == 0)
            return false;
        else {
            this.waitingRound--;
            System.out.println("距离出院还有"+this.waitingRound+"回合! ");
            return true;
        }
    }
}

```

## 2. 用户界面扩展。

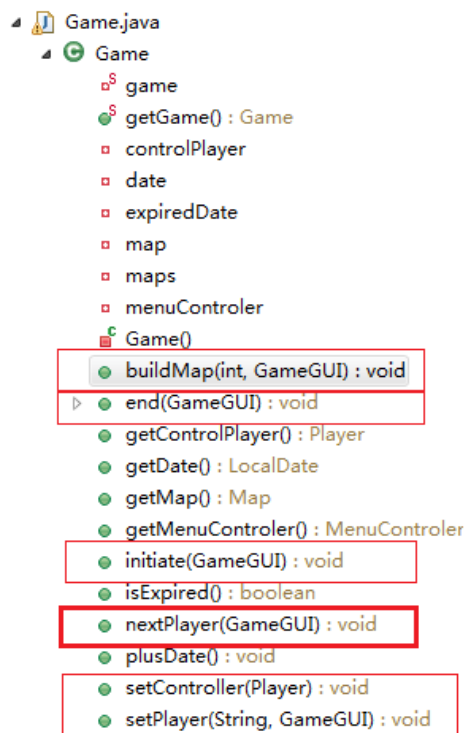
针对扩展用户的界面，新建的类主要有 GameGUI, PlayScene, StartScene, MapPane, MenuPane, MessagePane, PlayerPane, StockPane, TimePane, MessageDialog, PlayerData, StockData。

发生修改的类有：Game，MenuControler，Land 及其子类，Cell，Prop 及其子类。

下面就新增和修改的具体部分谈谈其中的 OO 思想。

首先，从修改的类入手。在 PJI 中设计的大致逻辑流程是由 Game 类作为整个游戏的控制类用于调用接口进行游戏的初始化以及回合制的控制；由 MenuController 来作为玩家在自己的回合内未发生移动之前的处理控制；另外 Land 的各个子类和 Prop 的各个子类用于处理各自的响应事件。

但是，当转成 GUI 时遇到的一个最大的问题，就是游戏逻辑的控制流程。由于在命令行版本，程序的执行可以在等待输入的时候中断，等到玩家键入回车的时候继续响应，然而 GUI 不同，程序的现实不可能再通过等待命令行输入的时候中断，而是要将那些处理输入的函数（或者说过程语句）放入相应控件的事件处理函数中，等到用户触发这个事件的时候进行异步的响应。因此需要将 Game 中原本的顺序控制流解散，即这个类将不再控制整个游戏的逻辑流程（将这个职责转到 GameGUI 这个类中）。Game 类的结构如下（其中被框起来的部分是指参与整体游戏流程控制的函数）：



另一个由于 GUI 的控制流程引起的比较大的改变是菜单的控制器，虽然因为图形界面的原因，许多菜单的选项已经移植到相应控件的事件监听函数中，但是还是姑且将其称为菜单控制器，现在它的职责只剩下让玩家在地图上移动、获得预警信息和认输的功能。其中让玩家在地图上移动的部分还是因为之前说的 GUI 的限制的原因，没办法让玩家在移动过程中中断（为了应对银行将在运行途中出发响应事件），因此我将 go 这个函数拆成了两部分 go() 和 go\_on()，并将玩家将要移动的步数

存入到菜单控制器的实例变量中，每次响应完之后，通过调用 `go_on` 来判断是继续移动还是转到下一个玩家。

然后第三组进行了修改的类是所有土地类型。第一，是增加对于土地显示的处理。由于大部分的土地在显示的时候都有一个矩形的背景框，除了对于“空地”类型显示的是一个圆（可以通过重写父类的 `toGraphics` 实现自己的背景显示方式，这里用到了**多态**的思想），因此将 `toGraphics` 操作放在了基类 `Land` 类中。然后对于每一块土地都有自己对应的显示图片。第二，是各个土地响应时间的重写，响应事件不可能再放在命令行中了，因此需要将其转化为图形界面的处理，修改的函数中只是将原来的循环处理流程转换成了事件监听处理方式。

接着修改的是所有的道具类。因为道具类不涉及显示问题，因此，只需要修改响应的函数，将其转化为图形化显示并将相应的事件处理放到监听器中。

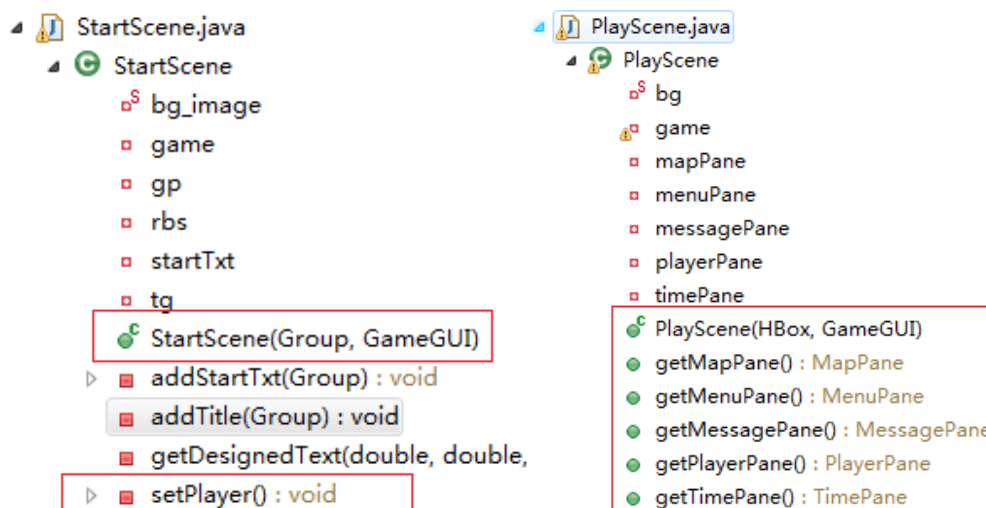
最后要解释的修改类是 `Cell` 类，为了方便将每个 `Cell` 布局到 `GridPane` 类型地图面板中，我让 `Cell` **继承** `StackPane`，从而可以直接将原来的 `cell` 实例 `add` 到地图格子面板中（**多态**）。同时也在 `Cell` 类中添加了 `toGraphics` 类似之前的 `toTexture`，然后一步步调用 `land` 的 `toGraphocs`，`player` 的 `toGraphics` 最后组装成一个小的完整的面板。

下面在简单解释一下新增的几个用于显示和控制游戏的类以及其中的控件之间消息发送和接受的机制。

其中 `GameGUI` 类是整个项目的主类。由于我的整个 GUI 开发用的是 `javafx` 的框架，由它对于主类的规定，我在 `start()` 函数设置了 `stage` 的第一个 `scene` 即 `start scene`，而在 `start scene` 中会通过相应的按钮调用该类的设置游戏场景接口将 `stage` 的 `scene` 重设为 `play scene`。另外在该类中还提供了对整个界面进行操作的接口，比如刷新玩家信息展示面板，刷新时间面板，刷新消息提示框。（这里需要提到 `javafx` 框架的优点，它的抽象实现让整个项目 OO 的思想更加突出，将每一个界面作为一个个场景的实例，整个窗口是一个 `stage` 的实例，从而能够更好地用面向对象的思路看待整个界面）。

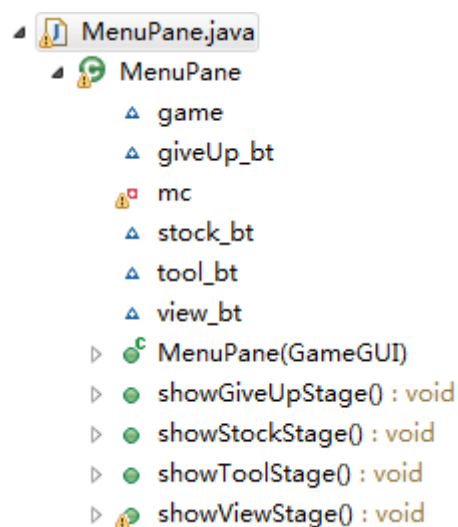
然后是游戏的两个场景 `start` 和 `play`。两者是实现的界面及其功能已经在第一部分中解释。这里主要提一下各自的结构，`startPane` 比较简单，用两组 `root` 来显示封面和玩家设置界面。而 `playPane` 则包含了之前提到的 5 个子面板（菜单、地图、提示、时间、玩家信息）。具体结构如下：





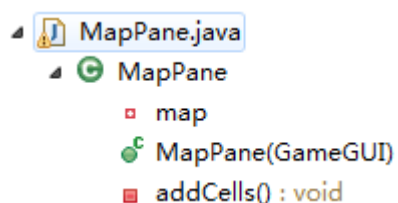
接下来是游戏场景下的每一个面板。

菜单面板。



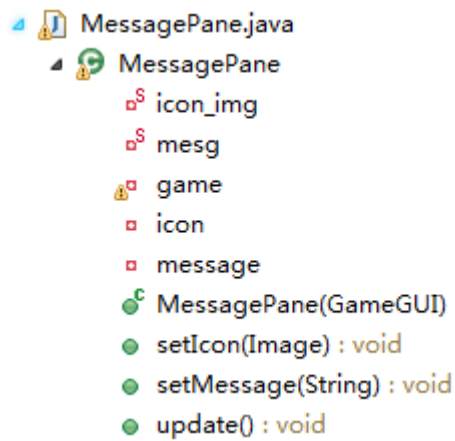
菜单面板有 4 个 button 分别会触发相应的处理窗口。其中，view\_bt, tool\_bt, stock\_bt, giveUp\_bt 的时间处理分别对应于 showViewStage(), showToolStage, showStockStage, showGiveUpStage 四个处理的函数。

地图面板。



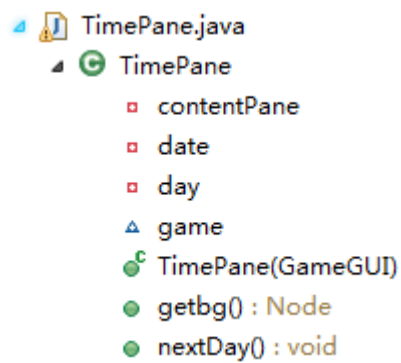
地图面板比较简单。它是继承自 GridPane，因此只要将原来数据层的地图中的每个 cell 利用循环逐个添加到相应位置即可。（之前提过，因为每个 cell 就是应小 pane）

消息面板。



消息面板提供了两个设置接口，设置消息内容和更新，其中外部调用设置消息内容接口主要用在遇到一些需要临时显示的信息的时候，而更新接口是通过进一步呈现给 `GameGUI` 来调用的，一般在设置路障或者更换游戏控制玩家的时候会调用。

时间面板。



用 `stakePane` 叠放背景和包含时间字符串的 `GridPane`。提供一个接口用于更新时间面板为下一天。

玩家面板。



同样，也是用 GridPane 来展示各个信息，并提供 refresh 接口用于更新玩家信息的显示。

### 三、 发现的设计问题

1. 在进行图形化扩展的时候，遇到了许多将原来的类进行拆解的操作，因为之前做 part1 的时候，让一个类或者一个函数承担了过多的操作和职责，这就加大了扩展的难度。对应的处理方法就是将相应的函数进行分解，分解成不同的几个操作。
2. 另一个最大的问题，就是 GUI 导致游戏逻辑流程的控制出现了问题。GUI 不能通过等待输入来中断函数的执行，因为 GUI 中用户的输入要么是通过触发控件的事件，要么是通过输入到文本框中，因此需要将一大部分的代码移动到相应控件的监听处理函数中，而这在某些场合下会在造成许多的赘余代码，比如银行事件的响应、游戏回合的轮转等。对于这个问题，应该可以通过多线程来解决。比如在银行响应的时间上，当遇到银行的时候，就开一个新的线程用于进行银行时间的响应，而暂时中断游戏主界面的线程，等到银行响应事件完成之后再通过关闭银行响应事件和恢复主界面线程，让玩家继续移动。而且这些改动，会让整个项目更加 OO，也更加便于维护和扩展。