项目名称 Project Name		密级 Confidentiality Level
		仅供收件方查阅
项目编号 Project ID	版本 Version	文档编号 Document Code

Fight the Landlord Game

详细设计说明书

符岗

汪钇丞

刘力玮

Revision Record

修订记录

Date 日期	Revision Version 修订版本	CR ID /Defect ID CR/ Defect 号	Sec No. 修改章节	Change Description 修改描述	Author 作者

1 Introduction 简介

1.1 Purpose 目的

本文档是对斗地主这款游戏的详细设计说明书，主要包含了详细的类的设计以及结构

1.2 Scope 范围

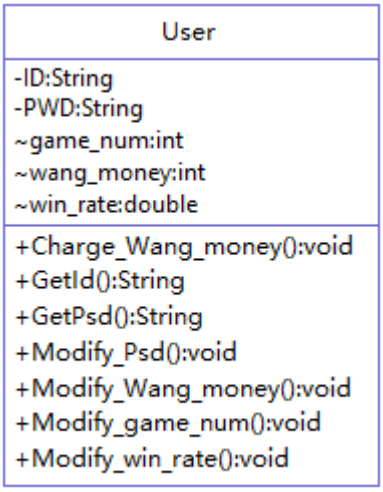
2 类详细设计

2.1 User Class 的设计

2.1.1 Overview 简介

用户类：主要是管理用户的相关信息，包括ID，密码，游戏信息等

2.1.2 Class Diagram 类图



2.1.3 Status Design 状态设计

无

2.1.4 Attributes 属性

private String ID 用户账号；

private String PSD 密码；

int Game_num 游戏总局数；

int Wang_money 汪币（欢乐豆）；

double WinningRate 胜率；

2.1.5 Methods 方法

2.1.5.1 charge_wang_money()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void charge_wang_money()
Description 功能描述	充值汪币
Calls 调用函数	无
Input 输入参数	int money
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

表 1 charge_wang_money()

2.1.5.2 GetId()

（1）Method Descriptions 方法描述

Prototype 函数原型	public string GetId()
Description 功能描述	获得用户的ID
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	String(Id)
Exception 抛出异常	无

表 2 GetId()

2.1.5.3 GetPsd()

（1）Method Descriptions 方法描述

Prototype 函数原型	public string GetPsd()
Description 功能描述	获得用户的密码(password)
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	String(psd)

Exception 抛出异常	无

表 3 GetPsd()

2.1.5.4 Modify_Psd()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void Modify_Psd()
Description 功能描述	修改用户的密码(password)
Calls 调用函数	无
Input 输入参数	String psd
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

表 4 Modify_Psd()

2.1.5.5 Update_Wang_money()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void Update_Wang_money()
Description 功能描述	更新用户的汪币
Calls 调用函数	无
Input 输入参数	int times,int base_money
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

表 5 Update_Wang_money()

2.1.5.6 Update_game_num()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void Update_game_num()
Description 功能描述	更新用户的游戏局数（每调用一次加一）
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

表 6 Update_game_num()

2.1.5.7 Update_win_rate()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void Update_win_rate()
Description 功能描述	更新用户的游戏胜率
Calls 调用函数	无
Input 输入参数	boolean win_or_fail
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

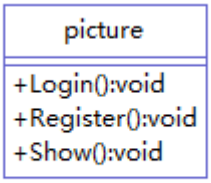
表 7 Update_win_rate()

2.2 Picture Class 的设计

2.2.1 Overview 简介

画面类：主要包括登录画面，注册画面，以及游戏画面的实现

2.2.2 Class Diagram 类图



2.2.3 Status Design 状态设计

无

2.2.4 Attributes 属性

无

2.2.5 Methods 方法

2.2.5.1 Login()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void Login()
Description 功能描述	用户登录
Calls 调用函数	无

Input 输入参数	public void String ID,String Psd
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

2.2.5.2 Register()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void Register()
Description 功能描述	用户注册
Calls 调用函数	无
Input 输入参数	String ID,String Psd
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

2.2.5.3 Show()

（1）Method Descriptions 方法描述

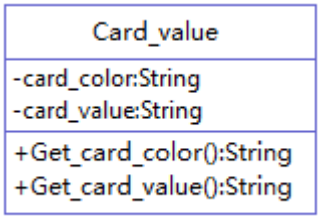
Prototype 函数原型	public void Show()
Description 功能描述	界面展示
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

2.3 Card_value Class 的设计

2.3.1 Overview 简介

牌值类：主要包含了牌的值以及花色

2.3.2 Class Diagram 类图



2.3.3 Status Design 状态设计

无

2.3.4 Attributes 属性

private String CardValue 牌值；

private String CardColor 花色；

2.3.5 Methods 方法

2.3.5.1 Get_card_value()

（1）Method Descriptions 方法描述

Prototype 函数原型	public string Get_card_value()
Description 功能描述	获得牌的值
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	String (card_value)
Exception 抛出异常	无

2.3.5.2 Get_card_color()

（1）Method Descriptions 方法描述

Prototype 函数原型	public string Get_card_color()
Description 功能描述	获得牌的值
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	String (card_color)

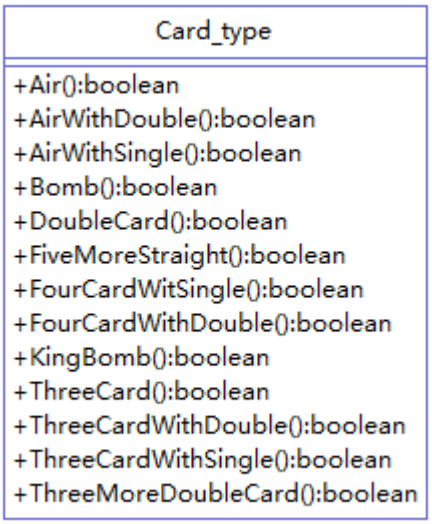
Exception 抛出异常	无

2.4 Card_type Class 的设计

2.4.1 Overview 简介

牌型类：主要是用来判断各种牌型的。（如对子，连对，飞机等）

2.4.2 Class Diagram 类图



2.4.3 Status Design 状态设计

无

2.4.4 Attributes 属性

无

2.4.5 Methods 方法

2.4.5.1 Air()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean Air()
Description 功能描述	判断是否是飞机(不带)
Calls 调用函数	无

Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.2 AirWithDouble()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean AirWithDouble()
Description 功能描述	判断是否是飞机（带的全为对子）
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.3 AirWithSingle ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean AirWithSingle ()
Description 功能描述	判断是否是飞机（带的全为）
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.4 Bomb ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean Bomb ()
Description 功能描述	判断是否是炸弹
Calls 调用函数	无
Input 输入参数	String card, int length

Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.5 DoubleCard ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean DoubleCard ()
Description 功能描述	判断是否是对子
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.6 FiveMoreStraight ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void FiveMoreStraight ()
Description 功能描述	判断是否是顺子
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.7 FourCardWitSingle ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean FourCardWitSingle ()

Description 功能描述	判断是否是四带二(带单)
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.8 FourCardWithDouble ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean FourCardWithDouble ()
Description 功能描述	判断是否是四带二(带对子)
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.9 KingBomb ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean KingBomb ()
Description 功能描述	判断是否是王炸
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean

Exception 抛出异常	无

2.4.5.10 ThreeCard ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean ThreeCard ()
Description 功能描述	判断是否是三个(不带)
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.11 ThreeCardWithDouble ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean ThreeCardWithDouble ()
Description 功能描述	判断是否是三带对子
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.12 ThreeCardWithSingle ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean ThreeCardWithSingle ()
Description 功能描述	判断是否是三带单

Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.4.5.13 ThreeMoreDoubleCard ()

（1）Method Descriptions 方法描述

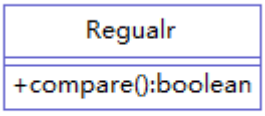
Prototype 函数原型	public boolean ThreeMoreDoubleCard ()
Description 功能描述	判断是否是连对
Calls 调用函数	无
Input 输入参数	String card, int length
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.5 Regular Class 的设计

2.5.1 Overview 简介

规则类：比较上家出的牌和本家即将出的牌的大小，防止出错

2.5.2 Class Diagram 类图



2.5.3 Status Design 状态设计

无

2.5.4 Attributes 属性

无

2.5.5 Methods 方法

2.5.5.1 compare()

（1）Method Descriptions 方法描述

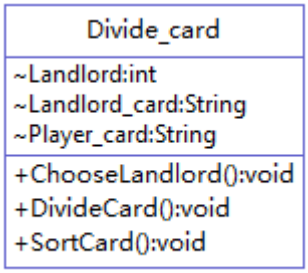
Prototype 函数原型	public boolean compare()
Description 功能描述	比较上家出的牌和本家即将出的牌的大小（根据牌型比较大小，除炸弹外，同类型牌才能比较）；
Calls 调用函数	无
Input 输入参数	Type card1, Type card2
Output 输出参数	无
Return 返回值	Boolean
Exception 抛出异常	无

2.6 Divide_card Class 的设计

2.6.1 Overview 简介

发牌类：实现每一句游戏开始的发牌操作，将一副牌随机分成三份，并得出底牌

2.6.2 Class Diagram 类图



2.6.3 Status Design 状态设计

无

2.6.4 Attributes 属性

`String Player_card;` 每个人的牌（随机分三份17张）;

`String Landlord_card;` 底牌

`int Landlord;` 谁是地主（随机选择）

2.6.5 Methods 方法

2.6.5.1 DivideCard ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void DivideCard ()
Description 功能描述	分牌（随机分三份17张）
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

2.6.5.2 SortCard ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void SortCard ()
Description 功能描述	对分的牌排序

Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

2.6.5.3 ChooseLandlord ()

（1）Method Descriptions 方法描述

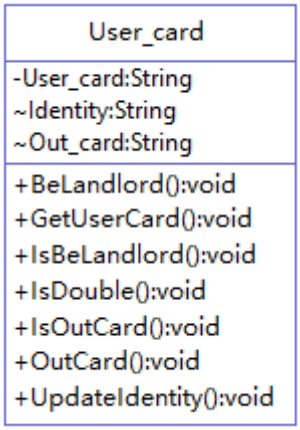
Prototype 函数原型	public int ChooseLandlord ()
Description 功能描述	开始随机选一个人作为地主，然后再由玩家进行选择
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	int landlord;
Exception 抛出异常	无

2.7 User_card Class 的设计

2.7.1 Overview 简介

用户牌类：记录用户的牌以及身份，由玩家选择是否成为地主并选择是否加倍，同时实现用户的出牌操作

2.7.2 Class Diagram 类图



2.7.3 Status Design 状态设计

无

2.7.4 Attributes 属性

private `String User_card;` 用户的牌（得到的以及出牌后实时更新的牌）

`String Identity;` 身份（地主或农民）

`String Out_card;` 用户要出的牌

2.7.5 Methods 方法

2.7.5.1 GetUserCard ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public string GetUserCard ()
Description 功能描述	获得用户的牌
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	String card
Exception 抛出异常	无

2.7.5.2 IsBeLandlord ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean IsBeLandlord ()
Description 功能描述	玩家选择是否成为地主
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	Boolean Landlord
Exception 抛出异常	无

2.7.5.3 BeLandlord ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void BeLandlord ()
Description 功能描述	玩家成为地主(最终)
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

2.7.5.4 UpdateIdentity ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void UpdateIdentity ()
Description	更新玩家的身份（地主或是农民）

功能描述	
Calls 调用函数	无
Input 输入参数	String player
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

2.7.5.5 IsDouble ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public void IsDouble ()
Description 功能描述	玩家选择是否将游戏倍数加倍
Calls 调用函数	无
Input 输入参数	int Times
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

2.7.5.6 IsOutCard ()

（1）Method Descriptions 方法描述

Prototype 函数原型	public boolean IsOutCard ()
Description 功能描述	玩家选择是否出牌
Calls 调用函数	无
Input 输入参数	无

Output 输出参数	无
Return 返回值	Boolean out
Exception 抛出异常	无

2.7.5.7 OutCard ()

（1）Method Descriptions 方法描述

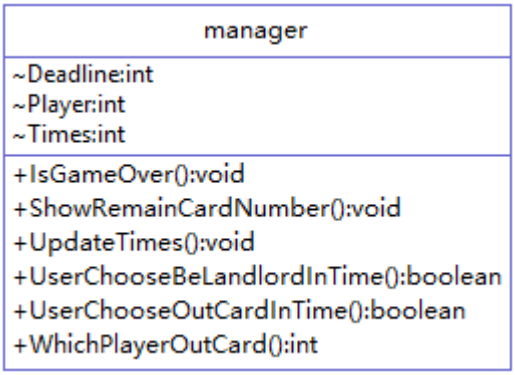
Prototype 函数原型	public void OutCard()
Description 功能描述	出牌（并更新牌）
Calls 调用函数	无
Input 输入参数	无
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

2.8 Manager Class 的设计

2.8.1 Overview 简介

管理类：主要管理整个游戏的运行，包括开始的选地主，以及轮流出牌，以及游戏结束后的结算等操作

2.8.2 Class Diagram 类图



2.8.3 Status Design 状态设计

无

2.8.4 Attributes 属性

Int Times 倍数；

Int Player 当前轮次的玩家；

Int Deadline 当前轮次的截止时间；

2.8.5 Methods 方法

2.8.5.1 UserChooseBeLandlordInTime()

（1）Method Descriptions 方法描述

Prototype 函数原型	Public boolean UserChooseBeLandlordInTime()
Description 功能描述	让当前轮次的玩家在规定时间内选择是否成为地主
Calls 调用函数	*sBeLandlord(),UpdateIdentity(),BeLandlord(),UpdateTimes(), WhichPlayerOutCard()*
Called By 被调用函数	无
Input 输入参数	int deadline
Output 输出参数	
Return 返回值	Boolean choose
Exception 抛出异常	无

2.8.5.2 UserChooseOutCardInTime()

（1）Method Descriptions 方法描述

Prototype 函数原型	Public boolean UserChooseOutCardInTime()
Description 功能描述	让当前轮次的玩家在规定时间内选择是否出牌
Calls 调用函数	*IsOutCard() OutCard() UpdateUserCard() WhichPlayerOutCard()*
Called By 被调用函数	无
Input 输入参数	int deadline， int player
Output 输出参数	card OutCard
Return 返回值	Boolean Out
Exception 抛出异常	无

2.8.5.3 UpdateTimes()

（1）Method Descriptions 方法描述

Prototype 函数原型	Public void UpdateTimes()
Description 功能描述	更新当前玩家的倍数
Calls 调用函数	*无*
Called By 被调用函数	IsDouble() OutCard() sBeLandlord() UpdateIdentity()
Input 输入参数	int addtimes, int player
Output 输出参数	Int Times
Return 返回值	无
Exception 抛出异常	无

2.8.5.4 IsGameOver()

（1）Method Descriptions 方法描述

Prototype 函数原型	Public void IsGameOver()
Description 功能描述	判断游戏是否已经结束
Calls 调用函数	*无*
Called By 被调用函数	UpdateUserCard()
Input 输入参数	无
Output 输出参数	Int player
Return 返回值	*Boolean end*
Exception 抛出异常	无

2.8.5.5 WhichPlayerOutCard()

（1）Method Descriptions 方法描述

Prototype 函数原型	Public int WhichPlayerOutCard()
Description 功能描述	判断下一个出牌的玩家
Calls 调用函数	*无*
Called By 被调用函数	UserChooseOutCardInTime(), UserChooseOutCardInTime()
Input 输入参数	Int player
Output 输出参数	无
Return 返回值	Int player
Exception 抛出异常	无

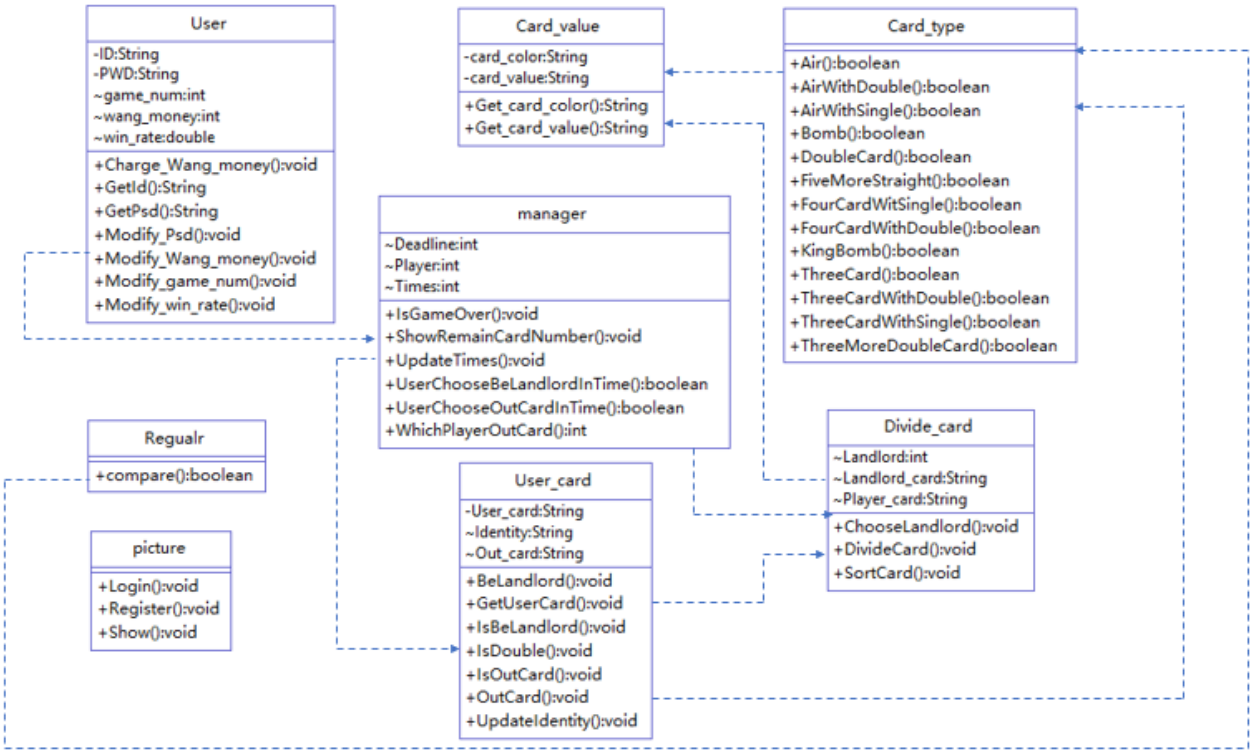
2.8.5.6 ShowRemainCardNumber()

（1）Method Descriptions 方法描述

Prototype 函数原型	Public void ShowRemainCardNumber()
Description 功能描述	在界面上显示每位玩家剩余的手牌
Calls 调用函数	无
Called By 被调用函数	UpdateUserCard()
Input 输入参数	Card usercard
Output 输出参数	无
Return 返回值	无
Exception 抛出异常	无

3 类关系

如下图所示：

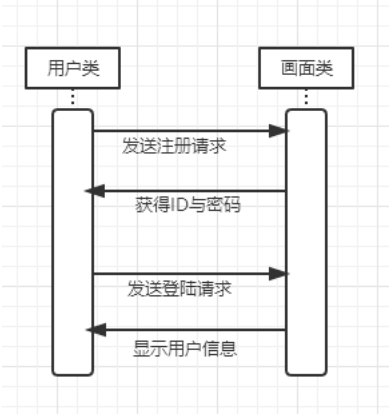


一共 8 个类，其中用户类在更新信息时与每一局游戏的结果有关，所以 User 类与

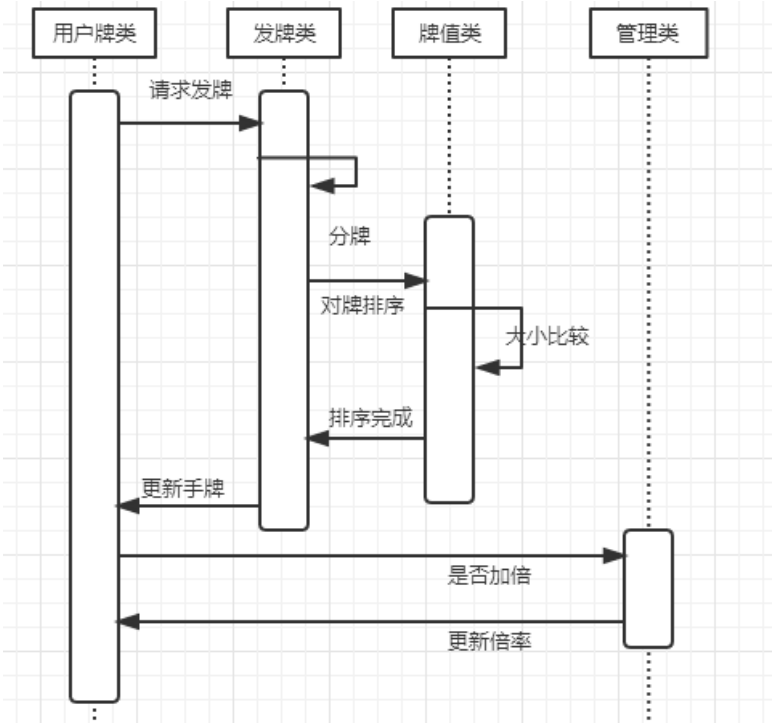
manager 类之间有依赖关系，manager 类管理游戏的进行，其中需要分牌，和更新游戏倍数，所以 manager 类和 User_card 类、Divide_card 类之间有依赖关系，然后，比较（compare）和出牌（OutCard）需要对牌型判断，所以 Regular 类、User_card 类与 Card_type 类有依赖关系，然后牌型判断以及牌的排序(SortCard)都需要获得牌值，所以 Divide_card 类，Card_type 类与 Card_value 类有依赖关系。

4 顺序图，活动图

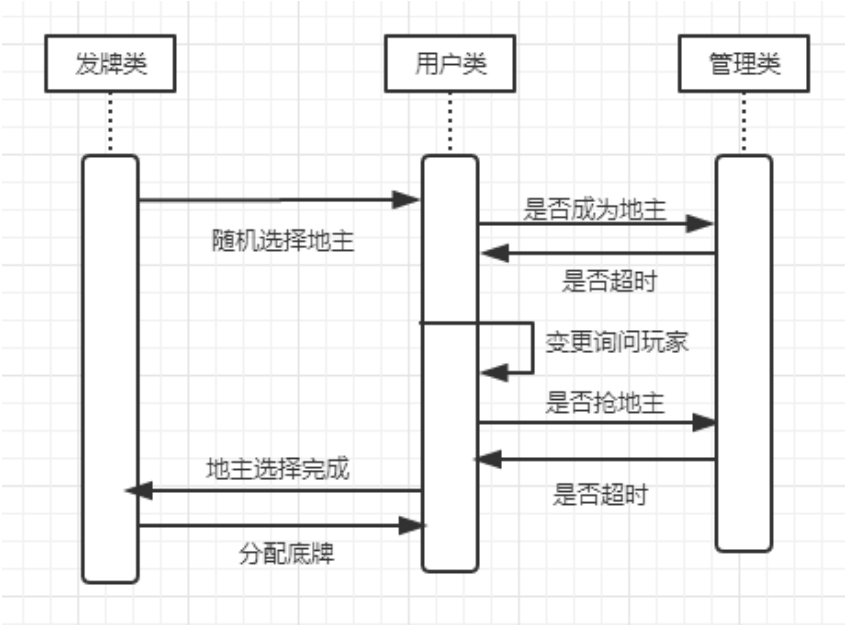
4.1 用户登录与注册

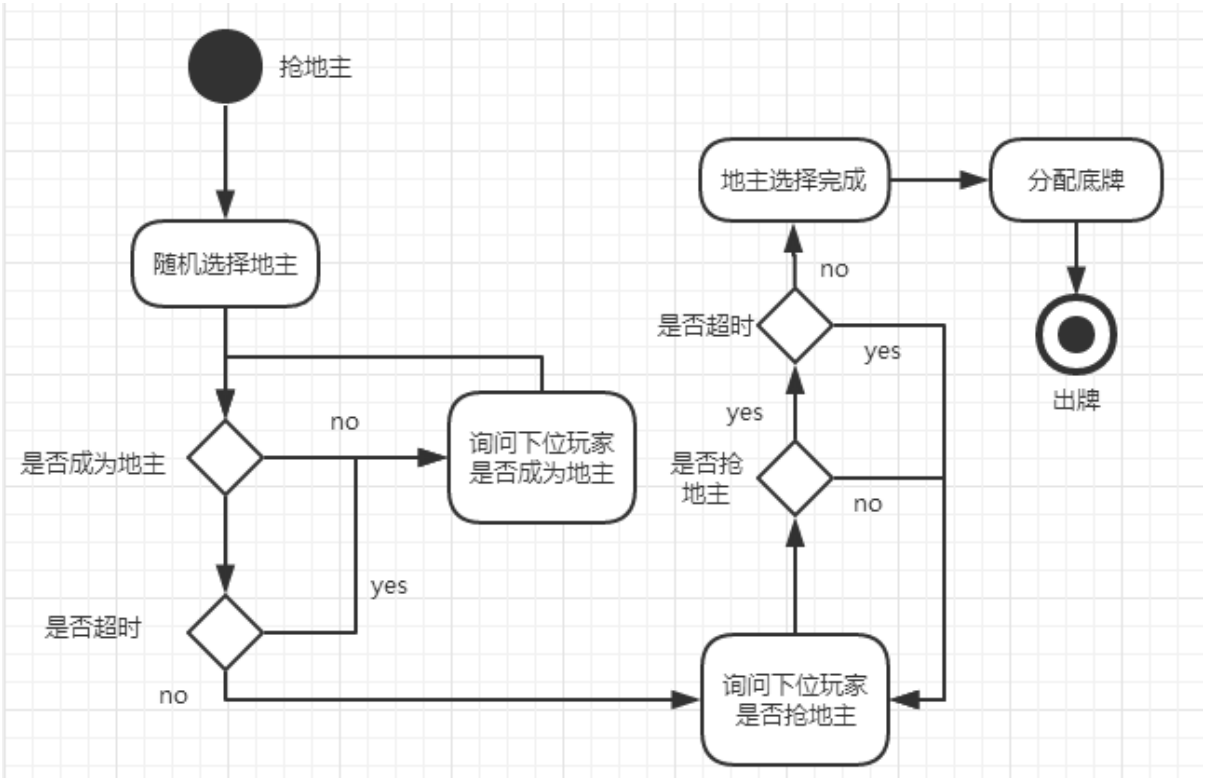


4.2 发牌

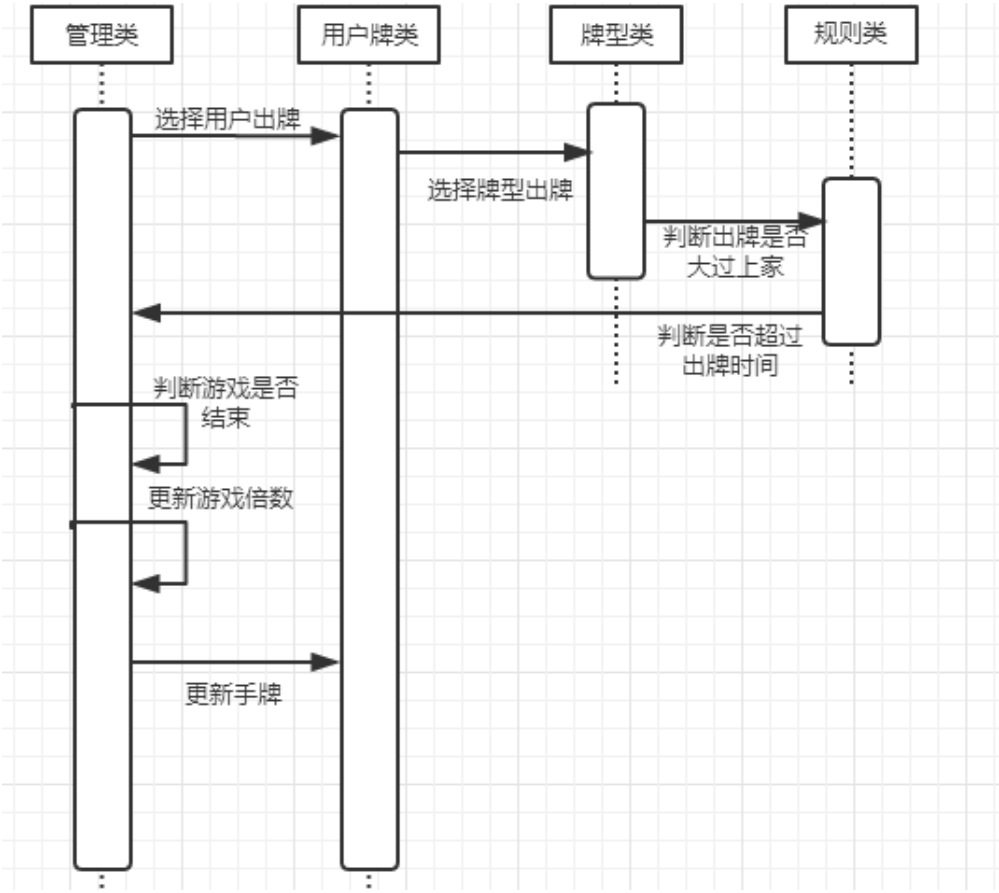


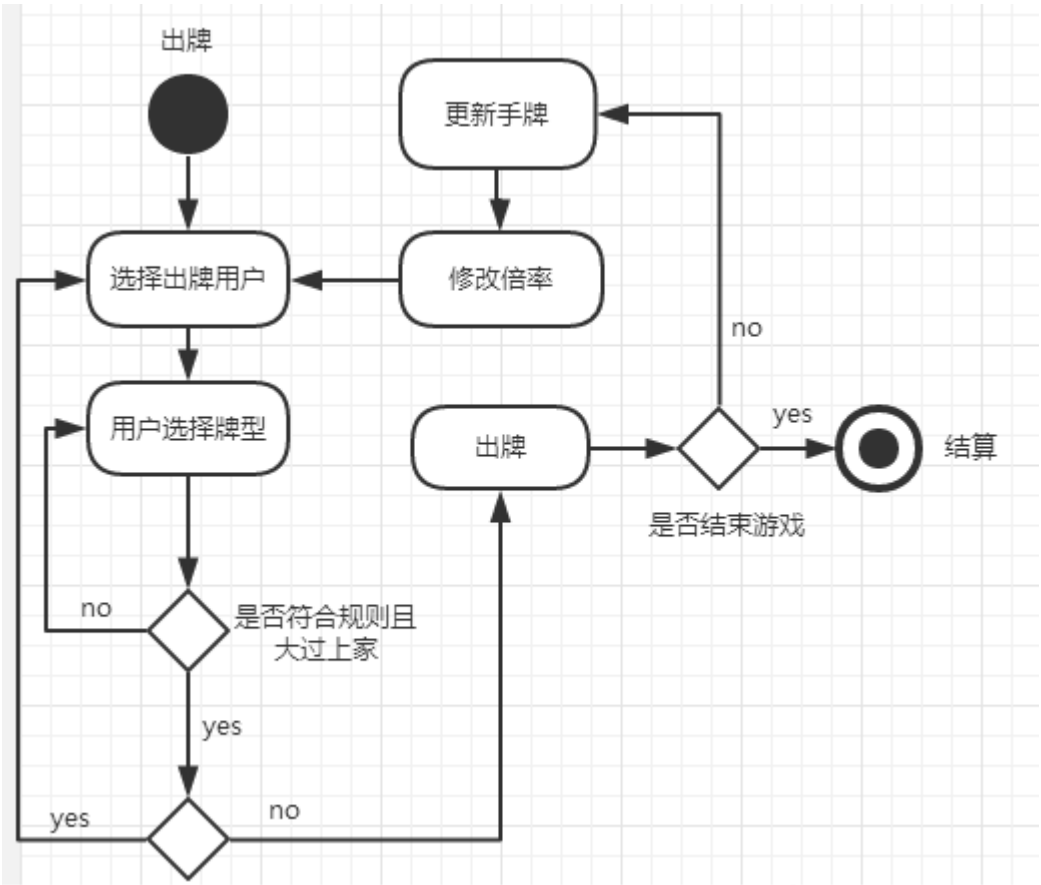
4.3 抢地主


发牌类
用户类
管理类
随机选择地主
是否成为地主
是否超时
变更询问玩家
是否抢地主
是否超时
地主选择完成
分配底牌

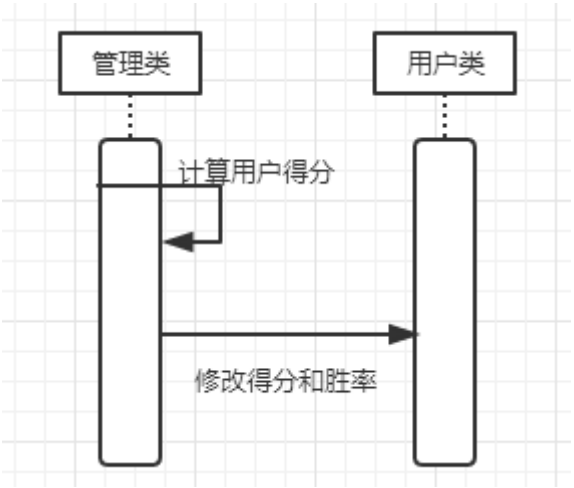

抢地主
随机选择地主
是否成为地主
no
询问下位玩家
是否成为地主
yes
是否超时
no
地主选择完成
分配底牌
出牌
no
是否超时
yes
yes
是否抢
地主
no
询问下位玩家
是否抢地主

4.4 出牌


管理类
用户牌类
牌型类
规则类
选择用户出牌
选择牌型出牌
判断出牌是否
大过上家
判断是否超过
出牌时间
判断游戏是否
结束
更新游戏倍数
更新手牌


出牌
更新手牌
选择出牌用户
修改倍率
no
用户选择牌型
出牌
yes
结算
是否结束游戏
no
是否符合规则且
大过上家
yes
yes
no

4.5 结算


管理类
用户类
计算用户得分
修改得分和胜率

5 设计模式

5.1 工厂方法模式

注：由于之前设计的斗地主玩法单一，太过简单，不够有新意，所以此次在之前的基础上又新加了几种玩法，以便丰富游戏的内容，增加游戏的趣味性。不过新增的玩法，有一点始终不变，那就是始终是农民斗地主，两个农门，一个地主，只是不同的玩法有不同的规则而已，下面分别简单介绍一下：

（1） 玩法一：就是之前设计的最普通，最经典的玩法，玩家根据手中的牌，合理运用自己的智慧，先出完者为胜者。（经典场）

（2） 玩法二：增加一张癞子牌，这张牌可谓是很厉害了，他可以替换成你想要的任何牌，包括王牌，但是一局只有一张癞子，而且不能单出。（单癞子场）

（3） 玩法三：同样有癞子，但不是增加的癞子牌，而是每一局发完牌地主确定后，随机选择一个值对应的 4 张牌最为癞子，即 4 张牌全为癞子，但是这里的癞子不能替换成王牌，而且可以单出。（多癞子场）

（4） 玩法四：去掉王牌，并且出牌规则是只能出单牌或对子，其他一律不准出。（单双场）

另外，根据不同玩家所拥有的财富不同，设计低级场，中级场，高级场，每位玩家根据自己的财富选择适合自己的场次进行游戏。

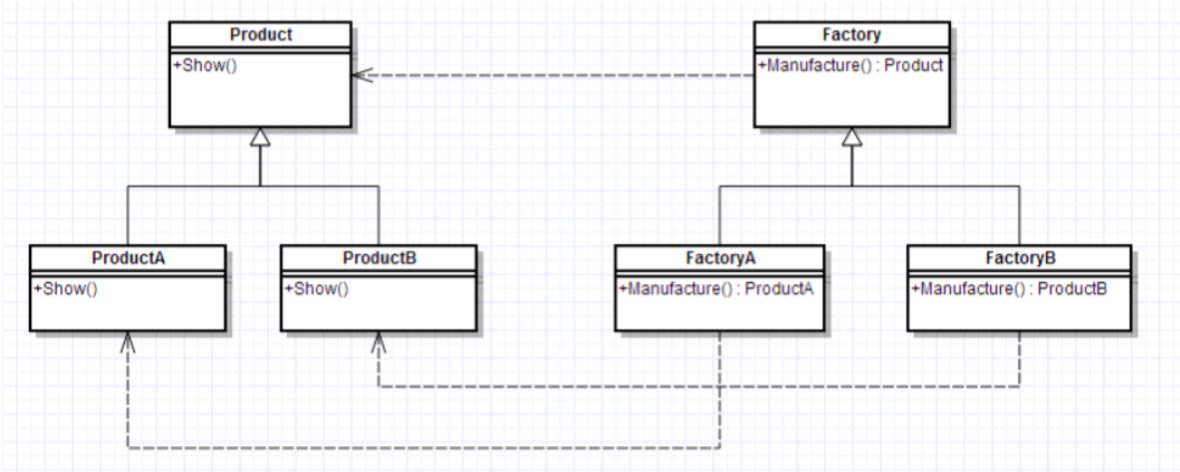
下面再来谈一下工厂方法模式：

工厂方法模式，又称工厂模式、多态工厂模式和虚拟构造器模式，通过定义工厂父类负责定义创建对象的公共接口，而子类则负责生成具体的对象。主要作用是将类的实例化（具体产品的创建）延迟到工厂类的子类（具体工厂）中完成，即由子类来决定应该实例化（创建）哪一个类。

模式组成如下：

组成	关系	作用
抽象产品（product）	具体产品的父类	描述具体产品的公共接口
具体产品（concrete product）	抽象产品的子类；工厂类创建的目标类	描述生产的具体产品
抽象工厂（creator）	具体工厂的父类	描述具体工厂的公共接口
具体工厂（concrete creator）	抽象工厂的子类：被外界调用	描述具体工厂，实现 Factory Method 工厂方法创建产品的实例

工厂模式类图：



如上图所示，该模式的具体工作流程是：

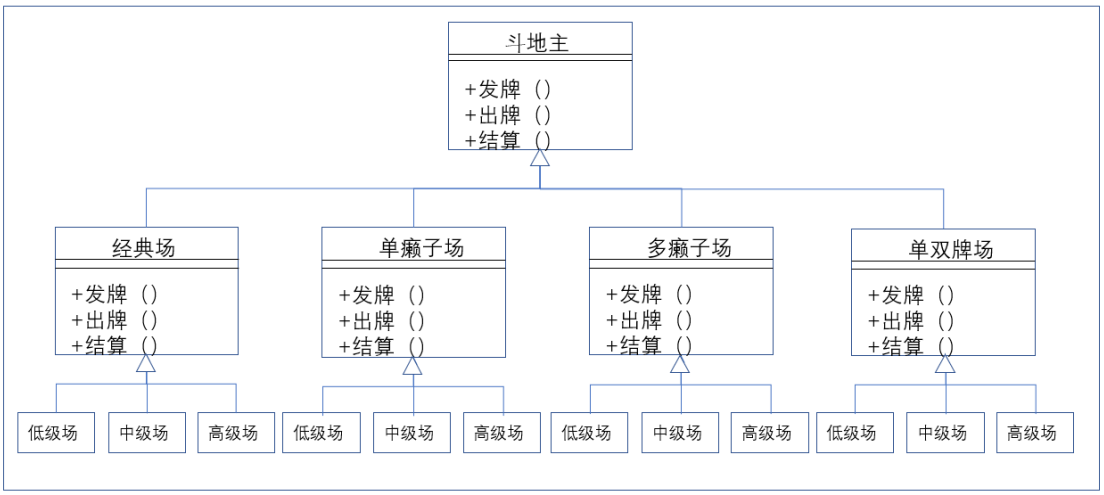
1：创建抽象工厂类，定义具体工厂的公共接口；

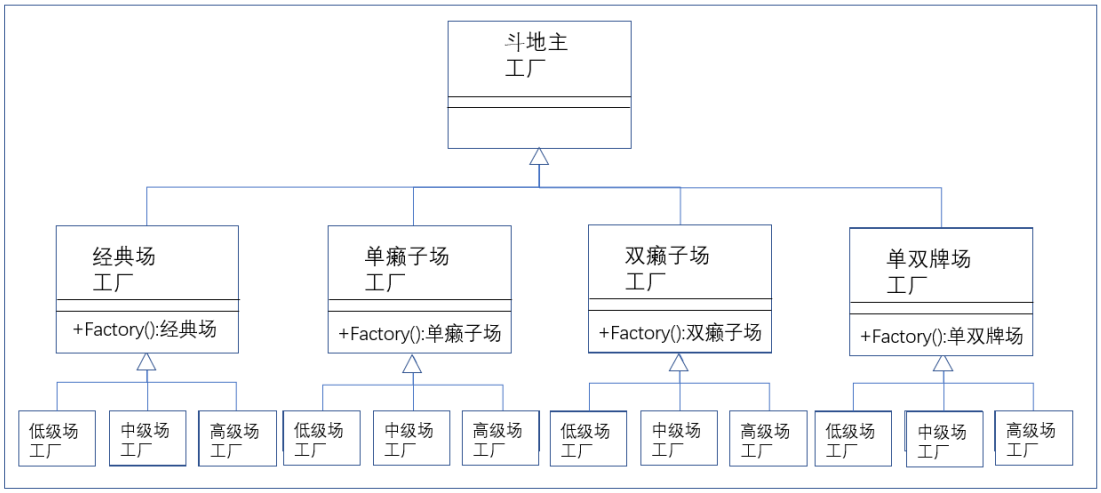
2：创建抽象产品类 ，定义具体产品的公共接口；

3：创建具体产品类（继承抽象产品类） & 定义生产的具体产品；

4：创建具体工厂类（继承抽象工厂类），定义创建对应具体产品实例的方法；

5：外界通过调用具体工厂类的方法，从而创建不同具体产品类的实例





在上面的两张简略的类图中，简单说明了在斗地主这个游戏中工厂方法模式是如何体现的，第一幅图中是具体的产品，最顶层的斗地主是一个抽象类，他定义了三个抽象的方法：发牌，出牌，结算，但是没有具体实现，具体实现在各个子类中，子类具体实现了发牌和出牌，而结算又在各个子类的子类中，因为不同等级场次的结算规则不同，所以结算在子类的子类中才具体实现。

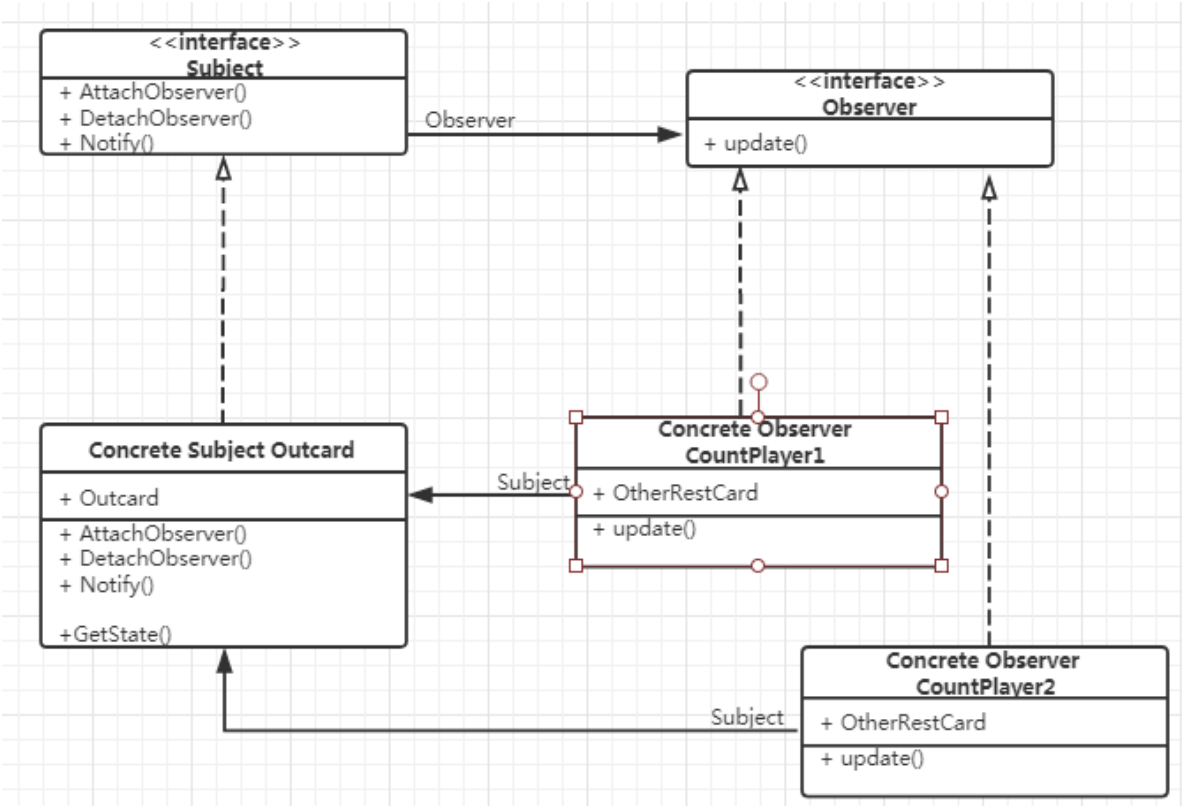
在这里，系统是一个工厂，而对应的产品就是不同玩法所对应的模式，当玩家选择相应想要玩的游戏模式后，系统会自动实例化相对应的产品，而不需要每次都去判断选择实例化哪个类，在调用方法时，会直接调用相应类的方法，而不用每次还要判断对象属于哪个类，然后再调用对应的方法，也不需要单独为每个产品创建单独的类（如果这样，会显得很麻烦，而且不利于扩展），在这种模式下，如果开发者想要增加游戏的模式，也是十分简单，因为增加一个模式，相当于增加一个产品，这对于一个工厂而言是再简单不过的事了。

5.2 观察者模式

所谓观察者模式就是一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。当这个主题对象在状态上发生变化时，会通知所有观察者对象，使它们能够自动更新自己。

由于在斗地主游戏中存在着很多的关联关系和依赖关系，各个玩家作为观察者需要对一个主题对象的状态进行同时监听，并且观察者根据主题对象的状态需要实时更新，所以采用观察者模式在斗地主游戏中是合适的。

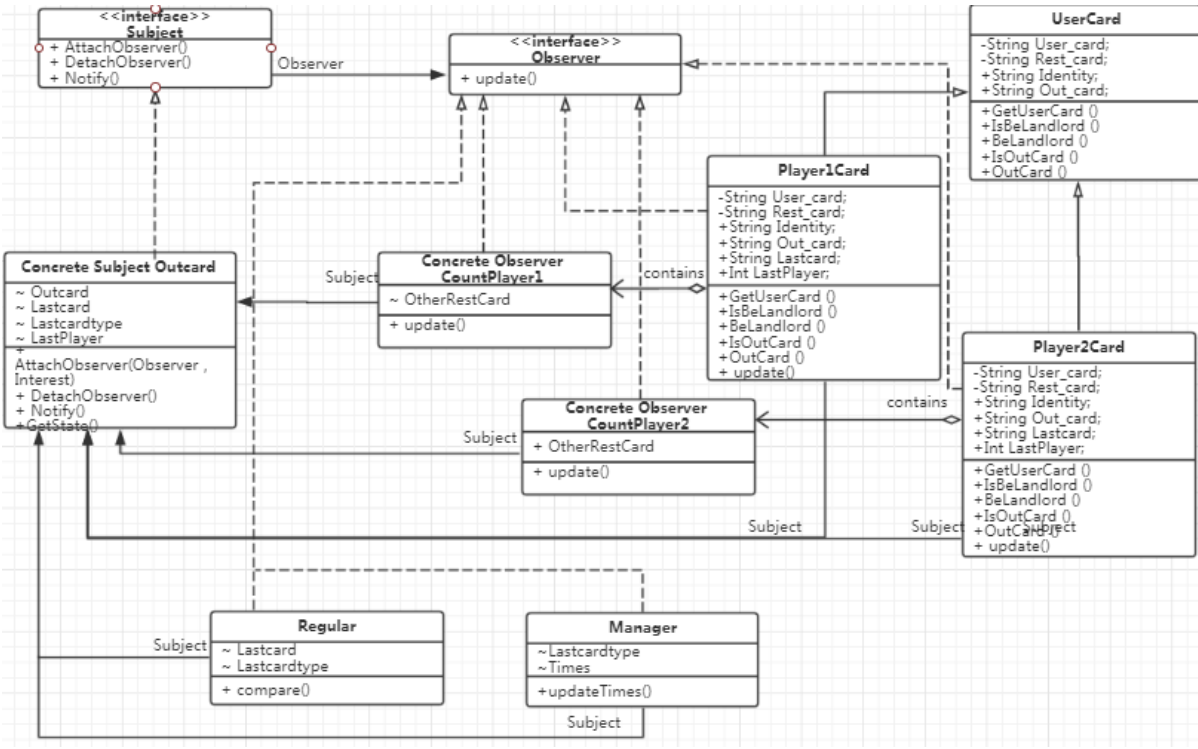
首先使用道具记牌器就需要用到观察者模式，记牌器记录和展示还留在其他玩家手中的牌，或者是已经打出去的牌，帮助玩家推测其他玩家可能持有的牌，来判断局势进行出牌。观察者模式具体到这个道具上来说的主题对象就是牌堆，即已经被出过的牌，而观察者就是使用了记牌器的玩家，每当牌堆里有其他玩家的牌新增，就需要通知各个记牌器，然后由记牌器做出更新，所以记牌器和牌堆的关系就是观察者和被观察者的关系，如下图所示：



当然，不止是记牌器，各个玩家的对象也需要观察牌堆，根据刚才上一次出的牌和出牌的对象，以决定是否出牌和跳过；规则的类也需要观察上一次刚出的牌，只是更进一步需要观察牌型和牌值；计分器也需要观察上一次刚出的

牌，只不过观察的是刚出的牌是否为炸弹需要更新倍数。

此时依然需要采用观察者模式，主题对象还是牌堆，但是观察者感兴趣的内容就不是这个牌堆里的所有牌，而是其他的主题对象的信息，所以 AttachObserver 方法可以具体到两项参数：（Observer，Interest），这样一来，玩家观察的 Interest 就是刚出的牌和出牌的玩家，规则的类的 Interest 就是刚出的牌的牌值和牌型，计分器的 Interest 就是刚出的牌是否为炸弹。具体的观察者模式 UML 类图为：



当然，由于部分的观察者，不需要每一次主题对象更新状态都更新自己的状态，例如计分器只有在出牌为炸弹时才更新倍数，可以加一个 ChangeManager 来进行调度和维护映射，并定义一个特定的更新策略，这里不再赘述。