华东师范大学软件学院实验报告

实验课程：OOAD　　**年级：**2020 级软件方向　　**实验成绩：**

实验名称：blackjack　　**姓名/学号：**

实验日期：2020 年 10 月 21 日　　10192100578 任璐

实验编号：No.12　　10205101530 赵晗瑜

目录

一、实验目的

1. 练习 blackjack 游戏概念模型的设计过程；
2. 进一步掌握 OOAD 的方法，通过系统的实现，体会 OOAD 的思想。

二、实验内容与实验步骤

1. 游戏规则：

1) 有关点数计算：J、Q、K 为 10，A 为 1 或 11，系统可自动判断对玩家或庄家最有利（最有可能赢）的情况确定 A 为 1 或 11；

2) 游戏初始化：

① 如果玩家是 A 和 10，则玩家即获得 blackjack，玩家赢，游戏结束；

② 如果玩家不是 blackjack，系统判断庄家是否为 A 和 10，若是，则翻开暗牌，庄家获得 blackjack，玩家赢，游戏结束。

3) 游戏继续进行：

① **玩家选择是否 double**（即选择是否将赌注加倍），再选择 hit 或 stand 或 split；

② **玩家选择 hit**：系统为玩家随机发一张牌，系统检查总点数是否超过 21 点，同时检查是否有两张相同的牌；

a. 玩家总点数超过 21 点，则庄家赢，游戏结束；

b. 玩家总点数未超过 21 点，玩家可以继续选择是否 double，再选择是否继续 hit。

③ **每一次 hit 后，**系统都会判断玩家是否有两张牌相同，若存在两张牌相同，系统的应用界面上即出现“split”按钮。如果玩家选择 split（分牌），并且玩家的总钱数足够将赌注加倍，则只保留两张相同的牌的某一张，并将赌注加倍。如果玩家点击了 split，但是总钱数不够加倍，则系统会提示不可以 split。（若为玩家发的牌中有 A，A 的点数按照使玩家的总点数最大且不爆的原则来计算）；

④ **玩家选择 stand：**当玩家点击 stand 按钮后，庄家首先翻开初始的暗牌，系统会持续为庄家发牌直至点数不小于 17（若为庄家发的牌中有 A，A 的点数按照使庄家的总点数最大且不爆的原则来计算）

a. 为庄家发完牌后，如果庄家的牌的总点数超过了 21 点，则玩家赢，游戏结束；

b. 为庄家发完牌后，如果庄家的牌的总点数没有超过 21 点，则比较此时玩家和庄家的牌数的大小，大方赢，相等则平局，游戏结束。

2．实验步骤：

1) 对问题进行细致的分析，描述分析过程。

2) 给出完整的用例设计：参与者、用例图、用例描述，非功能性约束（如果有的话）

3) 给出概念模型、设计类图，系统交互图

4) 用 C++或 Java 实现，并且需要一个用户界面（UI），把界面及问题的答案复制到实验报告中。

三、实验环境

1．计算机一台

2．UML 工具 Enterprise Architect

3．Java 开发框架 Springboot2.7.4

4．Vue.js 开发框架 uni-app

四、实验过程与分析

1. 分析过程

① 首先，我们对游戏规则描述进行分析，找出描述中的概念名词，根据 OOAD 的思想，我们将其抽象出六个类，分别是 Player（玩家）、BookMaker（庄家）、Poker（扑克

牌）、Game（一轮游戏）、GameRoom（游戏的主入口）、Common（玩家和庄家的公共类）。如下表是类的职责的简单描述（下文附有类图）：

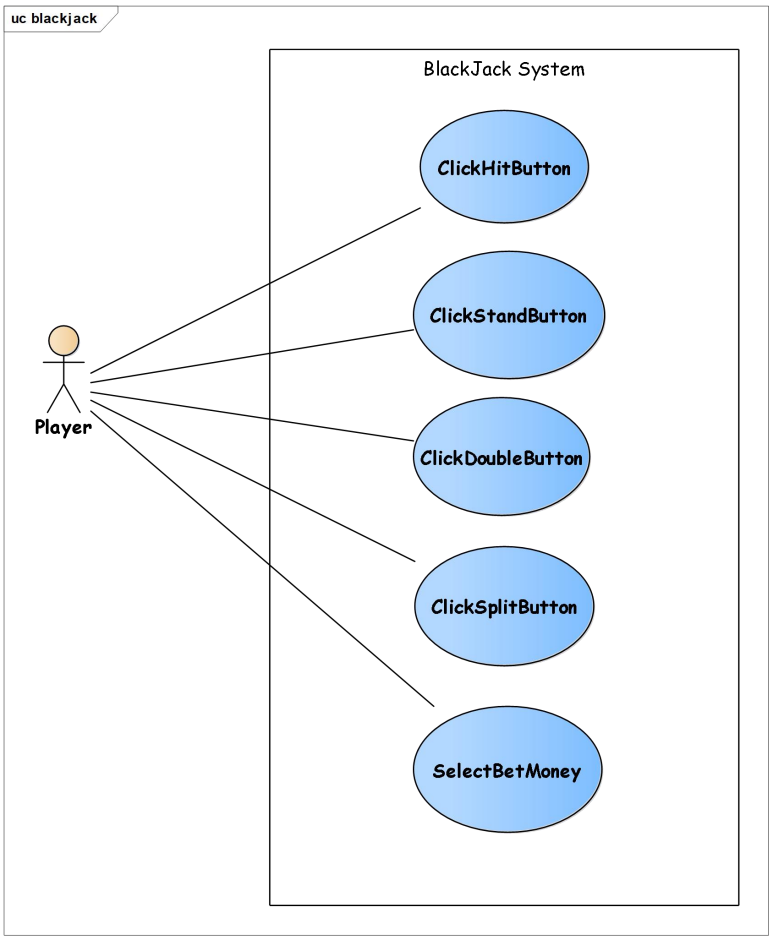
类	类的职责
Common	作为玩家和庄家的公共类，该类抽取了玩家和庄家的公共属性和行为，使得玩家和庄家可以直接沿用。
Poker	该类提供了扑克牌的各种信息
Player	该类提供了剩余的赌注数信息并作为游戏的一部分参与到游戏中
BookMaker	该类作为游戏的一部分参与到游戏中
Game	该类作为游戏的推动器，提供了系统实现的各种操作，如要牌、停牌、分牌、加倍等
GameRoom	该类是程序的主入口，可以实例化 Game 类

② 其次，我们对每个类的属性和方法都有哪些进行了分析，即实现层面。

a. 对于 Player 类，其应该具有 hit（出牌）、stand（分牌）、double（加倍）、

2. 用例设计

1) 用例图



2) **用例描述**,选择其中的三个用例进行描述，分别是：ClickHitButton，ClickStandButton，ClickDoubleButton

① 用例 ClickHitButton

Section Title	Section Content
Summary	The use case allows a Player to click hit button on the interface.
Actor list	Player
Pre-condition	The user selected certain bet money and started the game correctly, and the system allowed the user to select the hit button
Description	1. The user clicks the hit button on the UI interface. 2. The system randomly deals a poker to the user and displays it on UI interface. 3. The system determines whether the total number of the pokers is greater than 21 3.1 Total points greater than 21

	3.1.1 The system displays that the player has lost. 3.1.2 The use case ends. 3.2 Total points less than or equal to 21 3.2.1 The system waits for the player to perform the next operation. 3.2.2 The use case ends.
Post-condition	The system deals the pokers successfully for the player and determines whether the number of points was greater than 21.
Exception	1.a The system has crashed and the player fails to click the button. 2.a The system has crashed,so it fails to hand out a poker to the player. 3.a The system has crashed and it fails to determine whether the point is greater than 21

② 用例 ClickStandButton

Section Title	Section Content
Summary	The use case allows a Player to click stand button on the interface.
Actor list	Player
Pre-condition	The user selected certain bet money and started the game correctly, and the system allowed the user to select the stand button
Description	1. The user clicks the stand button on the UI interface. 2. The system continues to hand out pokers to the bookMaker until his total number of points is at least 17. 3. After handing out the pokers,the system determines whether the points of bookMaker is greater than 21 3.1 Total points greater than 21 3.1.1 The system displays that the bookMaker has lost. 3.1.2 The use case ends. 3.2 Total points less than or equal to 21 3.2.1 The system compares the points of the player with points of the bookMaker and determine that the larger stand is the winner. 3.2.2 The use case ends.

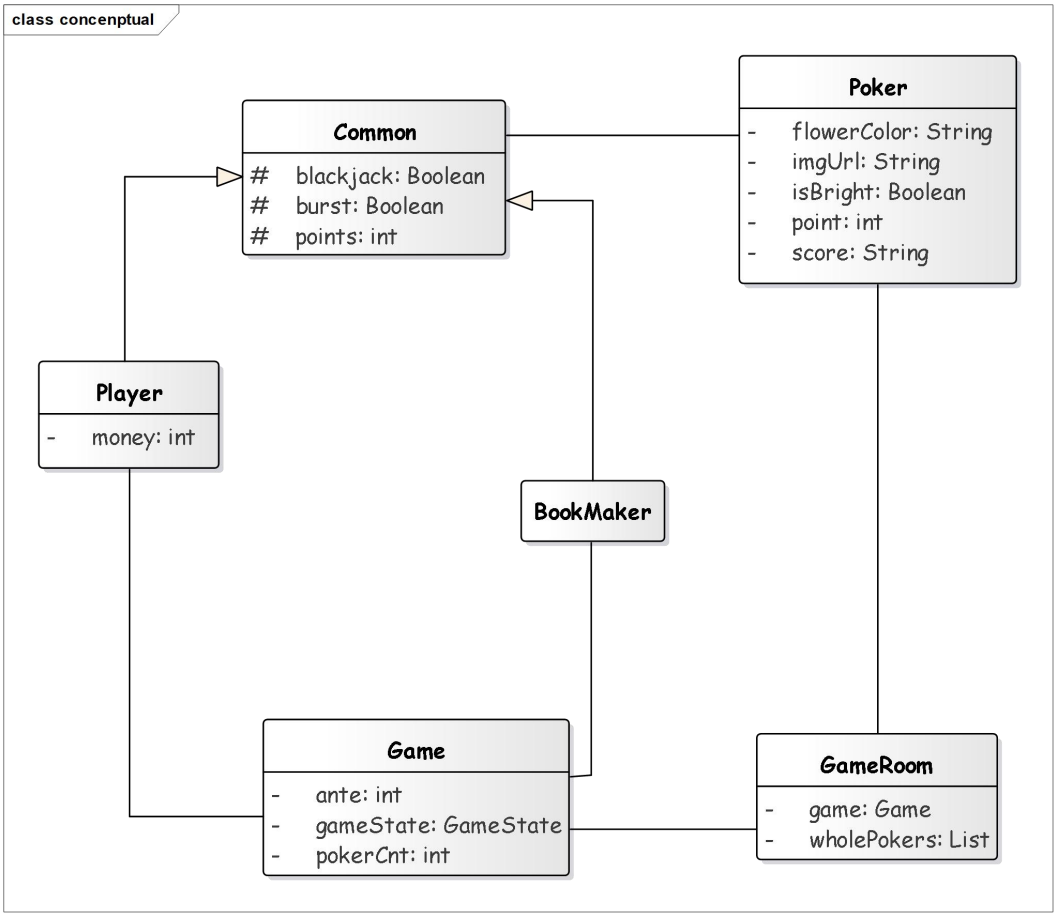
Post-condition	The system deals the pokers successfully for the bookMaker continuously and determines which one is the winner by the points of the player and the bookMaker.
Exception	1.a The system has crashed and the player fails to click the button. 2.a The system has crashed,so it fails to hand out pokers to the bookMaker. 3.a The system has crashed and it fails to determine whether the point is greater than 21 and which one is the winner.

③ 用例 ClickDoubleButton

Section Title	Section Content
Summary	The use case allows a player to click double button on the interface to double the bet money.
Actor list	Player
Pre-condition	The system initializes the UI interface correctly.
Description	1. The user clicks the double button on the UI interface. 2. The system determines whether the player' s money is enough to double. 2.1 Enough 2.1.1 The system doubles the bet money of player and displays it on interface. 2.1.2 The use case ends. 2.2 Not enough 2.1.1 The system notes that money is not enough. 2.1.2 The use case ends.
Post-condition	The system doubles the bet money of player or a mode box is displayed indicating that the user is short of money.
Exception	1.a The system has crashed and the player fails to click the button. 2.a The system has crashed,so it fails to double the bet money.

3. 概念模型

概念图中共有六个概念类，分别是 Player，BookMaker，Common，Poker，Game，GameRoom。



4. 类图

类图中共有 6 个类，对类的属性、方法、关系、类的职责的解释描述：

1) Common 类：

① 属性

burst：是否爆掉；points：玩家或庄家的总点数；blackjack：玩家或庄家是否获得黑杰克；pokers：扑克牌列表

② 方法：

hit：要牌操作

③ 类的职责

作为玩家和庄家的公共类，该类抽取了玩家和庄家的公共属性和行为，使得玩家和庄家可以直接沿用。

2) Player 类

① 属性

money：剩余的赌注数

② 类的职责

该类提供了剩余的赌注数信息并作为游戏的一部分参与到游戏中

3) BookMaker 类

① 类的职责

该类作为游戏的一部分参与到游戏中

4) Poker 类

① 属性

imgUrl：图片路径地址；score：牌上显示的点数；isBright：是不是明牌；point：实际的点数；flowerColor：花色

② 类的职责

该类提供了扑克牌的各种信息

5) Game 类

① 属性

bookMaker：庄家，游戏的参与者之一；player：玩家：游戏的参与者之一；pokerList：扑克牌列表；ante：当前赌注；pokerCnt：当前剩余扑克牌数；gameState：当前游戏状态

② 方法

init（）：初始化一局游戏；stand（）：停牌操作；playerHit（）：要牌操作；split（）：分牌操作；doubleAnte（）：加倍赌注

③ 类的职责

该类作为游戏的推动器，提供了系统实现的各种操作，如要牌、停牌、分牌、加倍等

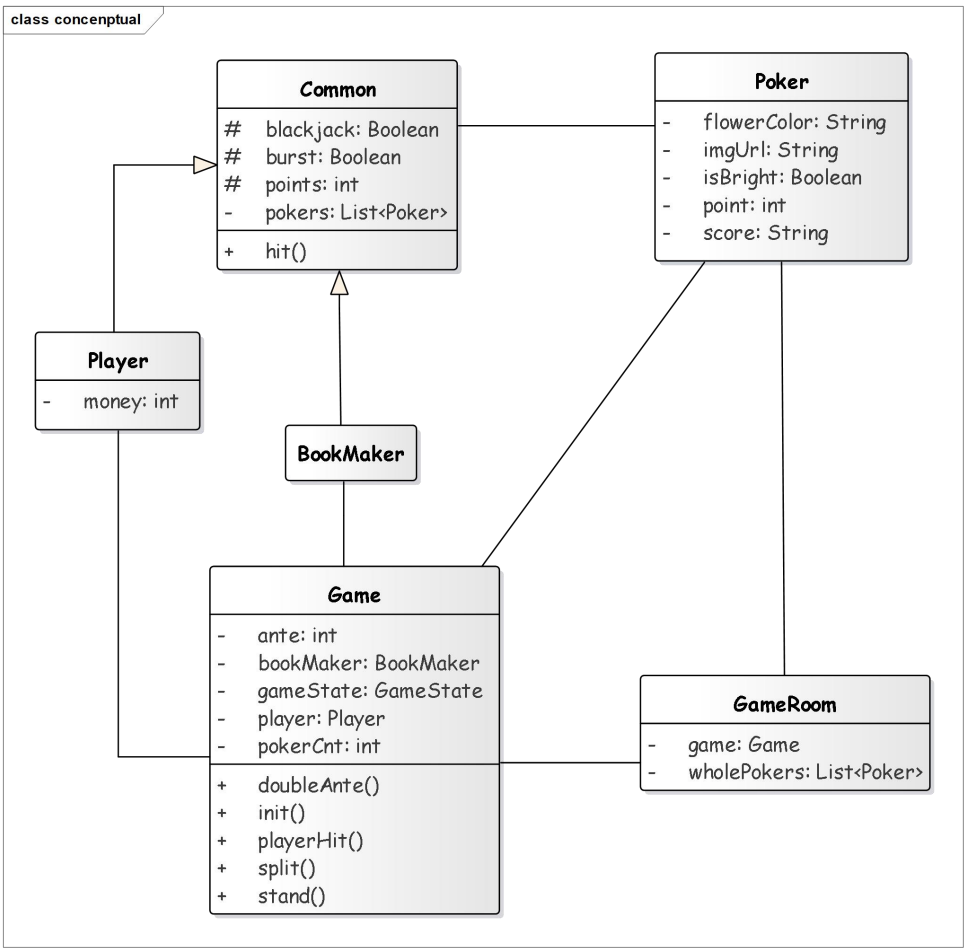
6) GameRoom 类

① 属性

game：Game 类的一个实例；wholePokers：所有的扑克牌列表

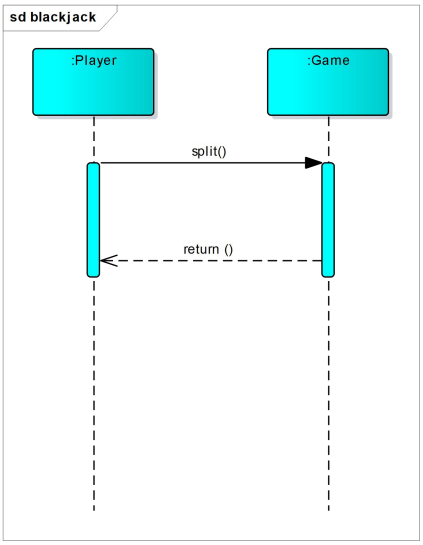
② 类的职责

该类是程序的主入口，可以实例化 Game 类

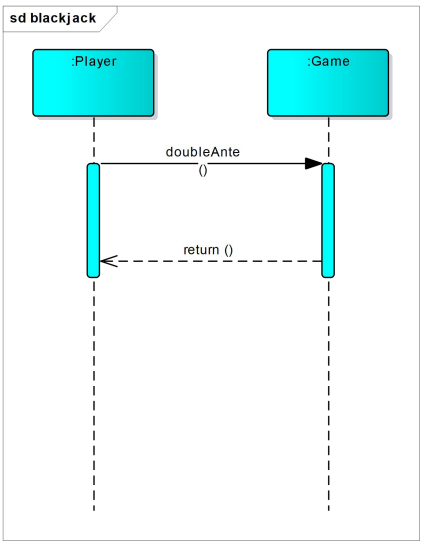


5. **交互模型**（因为涉及到类图中的方法较少，故交互图较为简单）

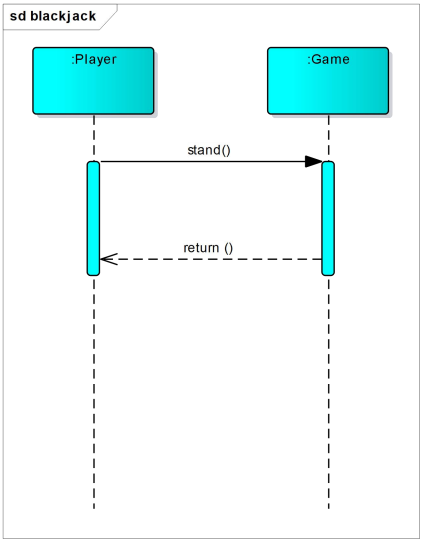
1) **玩家点击 split 按钮时的交互图**



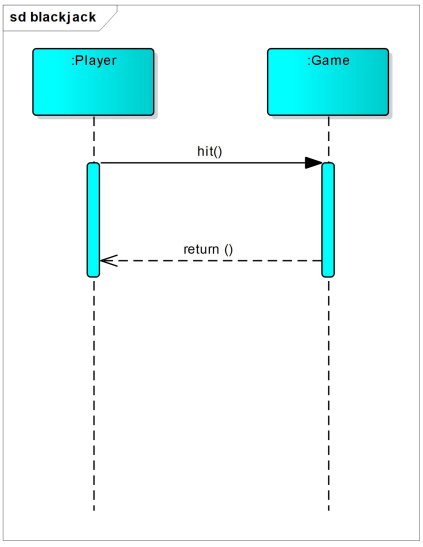
2) **玩家点击 double 按钮时的交互图**



3） 玩家点击 stand 按钮时的交互图

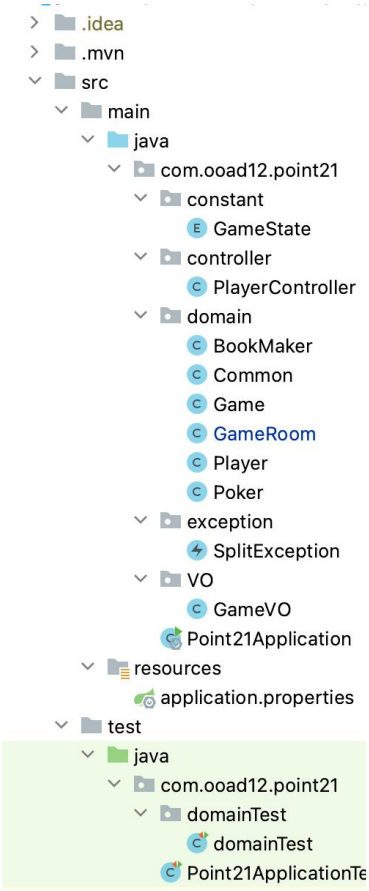


4） 玩家点击 hit 按钮时的交互图



6. 后台功能实现

1）工程目录



左图是是后台代码目录。

constant 包：游戏涉及到的常量

domain 包：游戏所涉及到的类，包含属性和方法（具体参见类图）

controller 包：与前台进行交互的接口

VO：传递给前端页面的对象，仅包含数据。

exception 包：包含游戏所涉及到的异常类

2）功能实现

① 由于玩家和庄家具有相同的属性和操作，例如当前拥有的牌、点数总和、要牌等，所以我们将这些共同属性和操作放在 Common 类中，由 Player 和 BookMaker 去继承他，然后各自再写各自特有的的属性和操作（例如只有庄家能翻开自己暗牌）。

② Game 类里放一轮游戏应该具备的属性和操作（我们将玩家可以选择操作放在 Game 类中而不是放在 Player 中，是因为玩家的操作会改变游戏的状态和数据，而不仅仅是玩家的数据和状态）：

一轮游戏应该具备玩家、庄家、扑克、赌注，为了方便记录游戏进行的状态，利用 gameState 去记录游戏的进行（GO_ON）、平局（PUSH）、庄家赢（BOOKMAKER_WIN）、玩家赢（PLAYER_WIN）， 用 pockerCnt 记录这幅扑克牌还剩几张。

在创建游戏时，游戏状态为 GO_ON;

当初始化发牌 init()时，给玩家和庄家各发两张牌之后（调用 bookMaker.closePocker()——设置暗牌），玩家和庄家去调用 isBlackJack()判断是否是黑

杰克，如果是就分出胜负（改变游戏状态）并对玩家的 monmey 进行增删，否则游戏状态保持不变；

当玩家选择要牌（playerHit()），就调用 player.hit()给玩家发牌，并检测玩家是否爆牌了，如果爆了，就改变 gameState = BOOKMAKERr_WIN，玩家 money 减去赌注 ante。否则游戏状态保持不变。

当玩家选择停牌(stand()) 就调用 bookMaker.disclosePocker（）翻开暗牌，当 bookMaker.getPoints()<17 让庄家一直 hit。然后检查庄家有没有爆牌，如果爆了游戏状态设置为 PLAYER_WIN，玩家增加 money，否则，就去比较点数大小判断游戏状态，对玩家 money 进行相应的增删。

如果玩家选择 split()，优先会去检查是否符合分牌要求 checkSplitCondition ，不符合的话，就抛出 SplitException 异常，否则就去掉最后一张牌，减去相应的点数 point，让赌注加倍，

③我们每轮游戏可以以全新的一副牌、新的玩家开始，也可以以之前的牌、之前的玩家重新继续游戏，所以我们设计 GameRoom 来操作这两种开始方式，前者设计为 startGame(ante)，后者设计为 continueGame(int ante)

④ 将前端玩家进行的操作传递到 PlayerController 中（开始游戏、继续游戏、要牌、分牌、停牌、赌注加倍），并返回相应的响应数据。

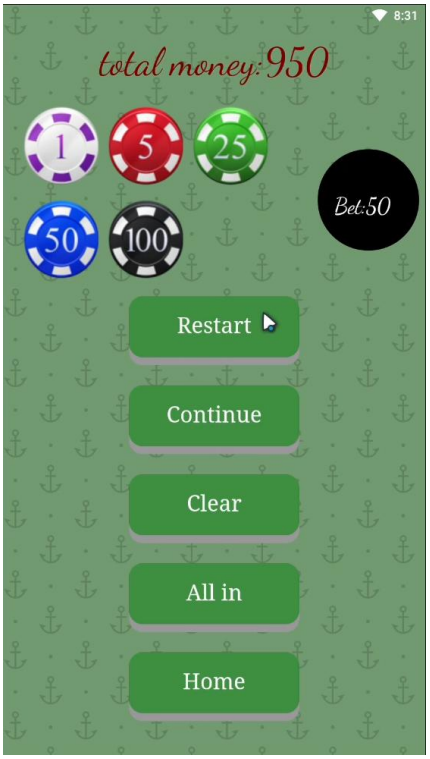
7. 前台功能实现

前台的实现采用了 uni-app 开发框架，使用 MuMu 模拟器模拟在真机上的运行，采用 uni.request 函数对后台提供的 API 进行调用。下面是对 app 界面和功能的展示和交互功能介绍。

1） 开始界面

左图是游戏的开始界面，点击“开始游戏”按钮即可进入游戏赌注选择的准备页面。

2） 游戏待开始界面



左图是游戏待开始界面，可点击图中的“1 5 25 50 100”按钮选择对应的赌注，选择时，“Bet”（将要注入的赌注）和 total money（总赌注）的数目将随之对应变化。

如，点击 50，Bet 会变为 50，而 total money 减少了 50；再点击 100，Bet 会变为 150，而 total money 又减少了 100。

选择 Restart 可让游戏重新开始（新开一副 52 张的牌）；

选择 Continue 可继续游戏（使用同一副牌）；

选择 Clear 可将将要注入的赌注清除，变为 0；

选择 All in 可将全部赌注一次性注入；

选择 Home 可返回游戏主界面。

3） 游戏进行界面

页面介绍：

① 左上角显示当前剩余的牌数量。

② 上面一部分是 Dealer（庄家）的 points 和扑克牌，下面一部分是 Player（玩家）的 points 和扑克牌。

③ 中间的按钮是将赌注加倍的按钮，同时旁边显示了当前的赌注。

④ 下面的 hit 按钮是玩家出牌，stand 按钮是玩家停牌。当玩家可以分牌时，会出现 split 的按钮。

API 调用：

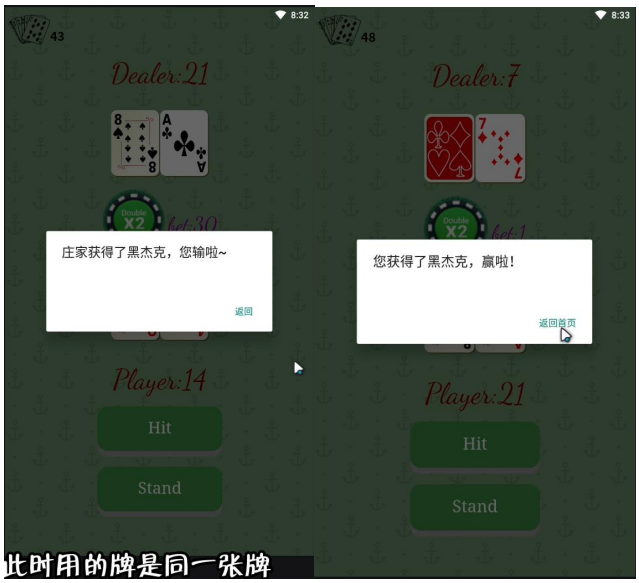
⑤ 在进入该页面时（初始化），系统调用后台/startGame 接口，传入 ante（赌注），返回玩家和庄家的所有信息，前台进行渲染；

⑥ 当用户点击 hit 按钮时，调用后台/hit 接口，返回玩家出牌后玩家和庄家的所有信息，前台进行渲染；

⑦ 当用户点击 stand 按钮时，调用后台/stand 接口，返回玩

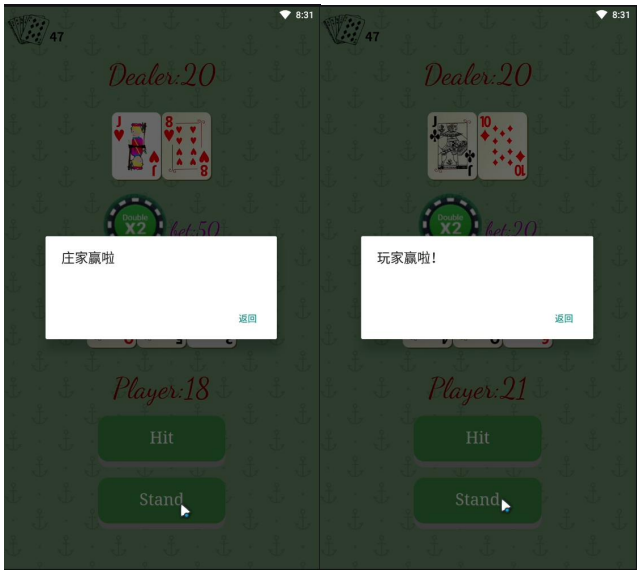
家出牌后玩家和庄家的所有信息，前台进行渲染。

4） 提示输赢和平局界面



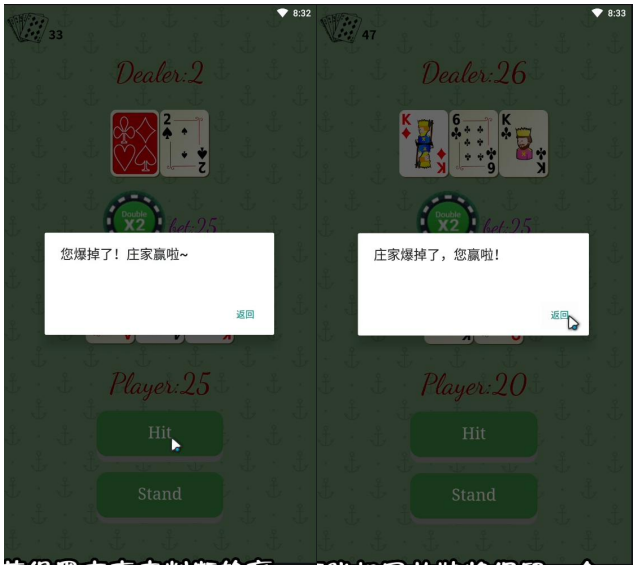
左图 1 是游戏刚开始时，系统判断庄家的分数是 21，则直接将其暗牌翻开，游戏结束，庄家赢了。

左图 2 是游戏刚开始时，系统判断玩家的分数是 21，游戏家属，玩家赢了。



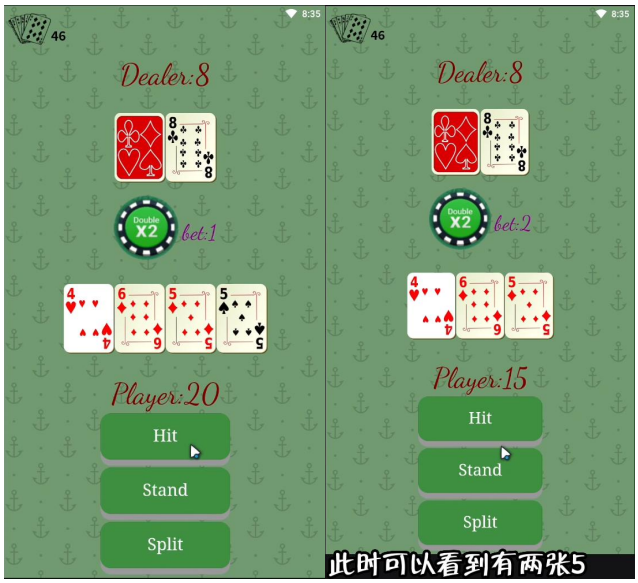
左图 1 是庄家和玩家开始都没有获得黑杰克，在游戏的进行过程中，玩家选择 Stand 并打完牌后，判断庄家的分数大于玩家的分数后的显示结果（20>18）。

左图 2 是玩家的分数大于庄家的分数后的显示结果(21>20)。



左图 1 是**玩家**的分数大于 21 时的游戏结果，庄家赢，游戏结束。

左图 2 是**庄家**的分数大于 21 时的游戏结果，庄家赢，游戏结束。



左图 1 中有两张点数相同的牌（两张 5），此时界面上出现了 split 按钮，玩家可以选择分牌操作。

左图 2 是玩家点击分牌后的情况，其中只有一张 5 被保留下来，并且赌注由 1 变为了 2（加倍）。

左图是庄家和玩家的分数相同时的平局情况。

五、实验结果总结

1. 通过本次练习 blackjack 游戏的设计过程和系统的实现，进一步掌握了 OOAD 的方法，体会了 OOAD 的思想；
2. 通过本次分工合作，加强了合作意识和合作时的沟通能力；
3. 完成了 blackjack 游戏的设计和开发。

六、附录

接口文档链接：开始游戏（新一副牌） - 21points (apifox.cn)

后台代码仓库：point21: ooad 21 点扑克牌游戏 (gitee.com)

前台代码仓库：hot-zhy/blackjack at master (github.com)