扑克牌游戏程序设计思路文档

**1. 功能需求**

* 随机发牌：将52张牌随机分配给4个玩家；
* 手牌排序：实现每个玩家分到的牌，按照花色和点数从小到大排列；
* 扑克游戏：基于已有的功能，写一个计算21点的小游戏（Black Jack）。

**2. 系统架构**

2.1 总体架构

本程序采用模块化结构设计，对设计内容进行拆分，可抽象出三个主要模块：

* 扑克牌模块：

负责生成52张牌的集合；

实现将牌随机分配给4个玩家的方法；

* 玩家模块：

记录玩家的有关信息（姓名）；

实现玩家接收牌的方法；

实现对玩家手中的牌按照花色和点数进行排序的方法。

* 21点游戏模块：

负责生成52张牌，并初始化的方法；

实现对玩家和电脑进行随机分牌的方法；

实现按照21点游戏规则计算点数和的方法；

实现游戏交互和最终结果判定的方法。

2.2 具体实现

通过使用面向对象的编程方法，从问题的需求中找到对象并抽象出对应的类。本程序具体包括以下的类和方法：

**Face**: 定义牌面属性的类，使用列表存储构成牌的所有花色和点数取值。

**Card**: 定义牌类，表示一张扑克牌，包含有花色和点数两个属性；而Card在初始化时，需由牌面属性的Face类里的元素才能进行合法初始化。

**PorkerGame**: 定义扑克游戏类, 包含进行一次扑克牌游戏的属性和方法。主要有：

PorkerGame.cards: 初始化的一副扑克牌，即由4 x 13=52个Card型元素组成的列表；

PorkerGame.current: 用于记录每次发牌后最上面的一张牌编号。初始化为0，每一次发完牌就+1；

PorkerGame.shuffle(): 洗牌操作，对上面初始化好的牌组cards，调用random.shuffle()函数进行随机打乱；

PorkerGame.deal(): 发牌操作，将打乱好的牌组发给玩家。每发一张牌，就使得current+1，始终确保current的值指向牌组最上方的牌；

PorkerGame.recevie(): 结合发牌方法PorkerGame.deal()，和用户得到一张牌的方法player.receive\_card()，以及利用循环，实现对所有玩家的取牌操作。

**Player**: 定义玩家类, 包含玩家的属性以及一些方法，主要有：

Player.name: 玩家姓名信息；

Player.cards: 玩家的手牌；

Player.receive\_card(): 玩家摸取一张牌；

Player.hand\_sort(): 实现对玩家手中的牌按照花色和点数进行排序的方法。主要是对Python中列表的库函数list.sort( key=None, reverse=False)，对其参数key指定为当前的牌花色和点数的索引值，这样的话就可以实现对牌类型Card按照其花色存储在列表的索引值和点数存储在列表的索引值进行排序。

**Blackjack**: 定义21点扑克牌游戏类，继承了已有的PokerGame类中一部分属性和方法，并实现了进行21点游戏的功能。

Blackjack.house：庄家（电脑）的手牌；

Blackjack.user: 玩家（用户）的手牌；

Blackjack.calculate(): 计算点数的方法；

Blackjack.start(): 游戏的运行方法。

备注：21点游戏部分借鉴参考了CSDN上的一篇文章，链接如下

原文链接 https://blog.csdn.net/Zhangguohao666/article/details/103948545

**原文是用函数的形式实现了21点游戏, 而我将原文的功能以类(Blackjack)的形式进行了封装。此外, 在实现Blackjack类中, 继承了已有的PokerGame类中的一部分属性和方法, 尽可能实现了代码的复用。**

**3. 游戏流程**

* 扑克发牌和手牌排序：（1）创建四位玩家；（2）创建一副完整的扑克牌并洗牌；（3）依次为每位玩家发牌，直到所有牌发完；（4）对每位玩家的手牌进行排序；（5）打印每位玩家的手牌信息。
* 21点游戏：（1）初始化一次21点游戏：创建一个Blackjack对象；（2）调用start()方法执行一次游戏；（3）按照交互提示，用户选择要牌或停牌；（4）系统判定胜负结果。

**4. 后续改进**

可基于现在所设计的核心玩法，对程序进行可视化处理，从而增强用户的游戏交互体验。