Poker Hand 解题报告

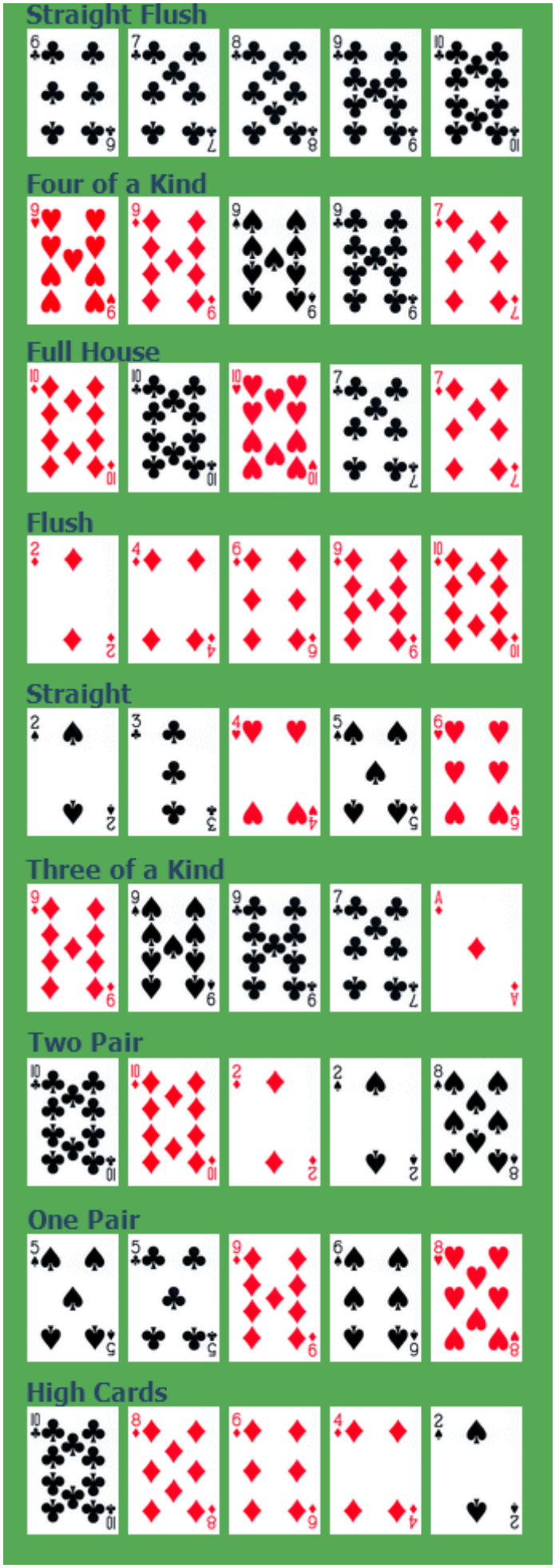
主题

模拟香港电影中梭哈的游戏，由黑方和白方的两手牌决定双方最终的胜负关系，胜负关系为胜、平、负。 有52张扑克牌，每张牌有4种花色：梅花(club)、方块(diamond)、红桃(heart)和黑桃(spade)。 牌面根据2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K, A的顺序排列且不考虑花色顺序。

黑白双方的牌面，先比较等级，等级按照如下规则排列，注意皇家同花顺就是同花顺，不做区分。

牌面分为如下等级，等级大小依次排序如下：

同花顺 > 四条 > 葫芦 > 同花 > 顺子 > 三条 > 两对 > 对子 > 高牌



如果双方的牌面具有相同的等级，有如下的判定规则来判断手牌的大小：

- **同花顺**： 同花顺比较的是最大一张牌的大小，如果相等就是和局（比如：AD KD QD JD TD 和 AS 2S 3S 4S 5S）结果为Tie。
- **四条**：四条的比较规则是比较牌面最大的牌的大小，四条是没有和局的。
- **葫芦**：葫芦的比较规则和三条类似，比的是3张同样牌的牌面的大小，葫芦也是没有和局的。
- **同花**：同花的比较规则和高牌一样
- **三条**：三条比较的是3张相同的牌的牌面大小，三条间的比较同样没有和局。
- **两对**：两对比较的是较大的一对牌面的大小，如果相同的话比较较小的一对牌面的大小，在相同的话比较剩余牌面的大小。
- **对子**：对子首先比较的是对子的大小，如果对子大小相等，比较的是剩余牌，每张牌的大小，如果都相等就打和。
- **高牌**：依次比较每张牌面的大小，如果都相等就打和。

解题思路

很多网上的答案就是根据牌面计分来计算最终的大小，看过源代码后感觉面条似的代码让人瞧着呕吐，根本无法读。这也是国内很多程序员的通病，写出的代码没有几个人能够看得懂……

这里阅读了 The Psychic Poker Player的源代码， Card这个基础类是用枚举来表示牌面和花色。

其中的两个枚举，enum Rank和enum Suite分别代表了牌面和花色，每一张牌就代表了Card类实例，此类设计的非常清晰。

enum HandType是**策略枚举**的集中体现，根据等级之间的规则来判定一副牌是属于哪个等级。

```
public enum HandType implements IHandType{

    HIGH_CARD("HIGH_CARD", 0) {
     ……
    },

    PAIR("PAIR",1) {
       ……
    },
    ……
    ……
    protected abstract boolean isValid(Card[] hand);

 }
```

想通了以上几步后同等级的排序规则可以用**ShowHandType策略枚举**来实现。

在判定输出过程中，可以用简单工厂方法模式来选择ShowHandType中的判定方法。

在此类繁复的模拟程序设计过程中，通过组织相关的测试，可以理清楚每个判断规则的设计的思路。

比如判定HandType的测试用例的设计：

```
@Test
    @Parameters(method = "dataProvider")
    public void testGetHandType(String cards, HandType handType) {
        IHandType machine = new IHandTypeImp(cards);
        Assert.assertEquals(machine.getHandType(), handType);
    }

    private Object[] dataProvider() {
        return $(
                $("3C KD 2S QC AH", HandType.HIGH_CARD),
```

```
                $("2C KD 2S QC AH", HandType.PAIR),
                $("KS KD 2S 2D 3H", HandType.TWO_PAIRS),
                $("JS JD KS KD QS", HandType.TWO_PAIRS),
                $("AS AH AC KS JD", HandType.THREE_OF_A_KIND),
                $("2C 3D 4S 5H 6H", HandType.STRAIGHT),
                $("AH 3H 2H KH JH", HandType.FLUSH),
                $("KS KC AH AC AD", HandType.FULL_HOUSE),
                $("2C 2D 2H 2S KS", HandType.FOUR_Of_A_KIND),
                $("TH JH QH KH AH", HandType.STRAIGHT_FLUSH)
        );
    }
```

经验总结

1. 这是一道非常好的模拟题，只要遵循GOF的设计原则和清晰的基础类的设计，就可以设计出可读性很高的牌面判定规则的逻辑设计。
2. 设计判定牌面判定规则中，我遵循的是测试先行策略，通过测试用例的组织合适，来影响牌面判定规则的逻辑设计，总计了36条测试用例，覆盖所有牌面的判定规则，所以最后提交代码一次AC。
3. 设计模式方面，**简单工厂模式**和**策略模式**特别是策略枚举重复使用2次，给我的感受是优秀设计模式减少了很多冗余的代码的编写。
4. 代码还是略显冗余有1000行之多！？程序是用来读的，但是这些代码比"面条"代码易读的多。

题后思考：

1. 否设计一个发牌程序，这样组织测试代码的数据就容易一些?
2. 可否再精简一下。
3. 能否设计一个Spring-boot的Web端程序，包括发牌，判断等接口?

关联问题

天眼模式判断最大的牌

梭哈