

# Telecom Sudparis



**Software Engineering**  
**MAT 7003 : Mathematical Foundations**

## Project: Poker Hands

NGUYEN Van Luong  
[nguyen\\_v@telecom-sudparis.eu](mailto:nguyen_v@telecom-sudparis.eu)

NGUYEN Thi Mai  
[nguye\\_tm@telecom-sudparis.eu](mailto:nguye_tm@telecom-sudparis.eu)

Professor: J Paul Gibson  
[paul.gibson@it-sudparis.eu](mailto:paul.gibson@it-sudparis.eu)

*TSP, 12/2012*

## CONTENT

1. POKER HANDS PROBLEM
  - 1.1 Poker hands problem describe
  - 1.2 The rules
2. SOLUTION
  - 2.1 Represent a card
  - 2.2 Represent a set cards
  - 2.3 Represent a hand
  - 2.4 Represent a fighting
3. APPLY IN RODIN AND IMPLEMENT IN JAVA
  - 3.1 Apply in Rodin
  - 3.2 Implement in Java programing language
  - 3.3 Test
  - 3.4 Mini program

## 1 POKER HANDS PROBLEM.

### 1.1 Poker hands problem description

**Poker** is a family of card games involving betting and individualistic play whereby the winner is determined by the ranks and combinations of their cards.

There are 52 cards in the pack, and the rank of the individual cards, from high to low (ace, king, queen, jack, 10, 9, 8, 7, 6, 5, 4, 3, 2). There is no rank between suits - for example, the king of heart and the king of spade are equal.

A poker hand consists of five cards. The categories of hand, from highest to lowest, are listed below. Any hand in a higher category beats any hand in a lower category (for example, any three of a kind beats any two pairs). Between hands in the same category the rank of the individual cards decides which is better, are described in more detail below.

In games where a player has more than five cards and selects five to form a poker hand, the remaining cards do not play any part in the ranking. Poker ranks are always based on five cards only.

**The Poker hands problem:** We have two hands. We have to find which is the winner.

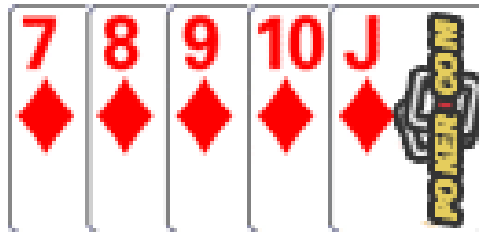
### 1.2 The rules

#### 1.2.1 Royal flush



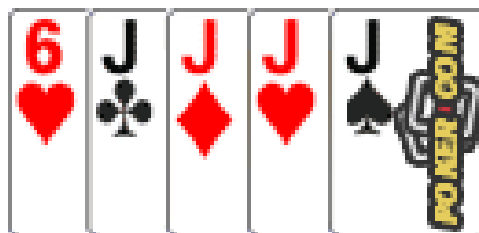
This is the highest poker hand. It consists of ace, king, queen, jack, ten, and all in the same suit. Because all suits are equal, all royal flushes are equal.

#### 1.2.2 Straight flush



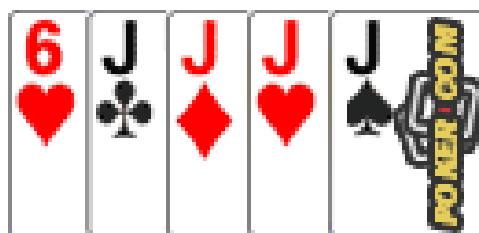
Five cards of the same suit in sequence - such as Heart (J- 10- 9- 8- 7). Between two straight flushes, the one containing the higher top card is higher. Moreover, The cards cannot "turn the corner": Spade( 4- 3- 2- A- K) is not valid.

### 1.2.3 Four of a kind



Four cards of a kind - such as four queens. The fifth card can be anything. Between two fours of a kind, the one with the higher set of four cards is higher - so 3-3-3-3-A is beaten by 4-4-4-4-2.

### 1.2.4 Full house



This consists of three cards of one kind and two cards of another kind - for example three sevens and two tens (colloquially known as "sevens full" or more specifically "sevens on tens"). When

comparing full houses, the rank of the three cards determines which is higher. For example 9-9-9-4-4 beats 8-8-8-A-A.

### 1.2.5 Flush



Five cards of the same suit. When comparing two flushes, the highest card determines which is higher. If the highest cards are equal then the second highest card is compared; if those are equal too, then the third highest card, and so on. For example Heart(K- J- 9- 3- 2) beats Spade(K- J- 7- 6- 5) because the nine beats the seven.

### 1.2.6 Straight



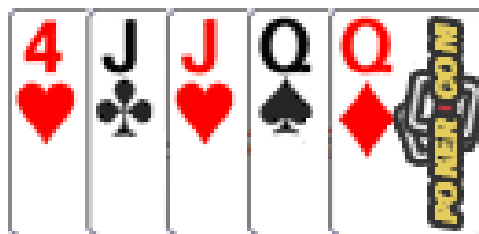
Five cards of mixed suits in sequence - for example {Heart(Q- J) - Diamond(10)- Club(9- 8)}. When comparing two sequences, the one with the higher ranking top card is better.

### 1.2.7 Three of a kind



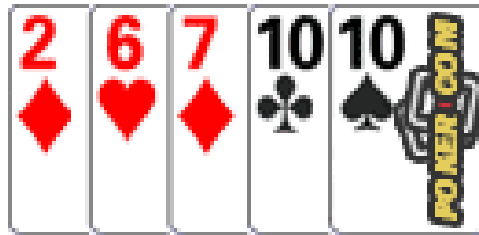
Three cards of a kind plus two other cards. When comparing two threes of a kind the hand whereby the three are higher value is better. So for example 5-5-5-3-2 beats 4-4-4-K-Q.

### 1.2.8 Two pairs



A pair is two cards of equal value. In a hand with two pairs, the two pairs are of different value (otherwise you would have four of a kind), and there is an odd card to make the hand up to five cards. When comparing hands with two pairs, the hand with the highest pair wins, irrespective of the rank of the other cards - so J-J-2-2-4 beats 10-10-9-9-8 because the jacks beat the tens. If the higher pairs are equal, the lower pairs are compared, so that for example 8-8-6-6-3 beats 8-8-5-5-K. Finally, if both pairs are the same, the odd cards are compared, so Q-Q-5-5-8 beats Q-Q-5-5-4.

### 1.2.9 Pair



A hand with two cards of equal value and three other cards which do not match these or each other. When comparing two such hands, the hand with the higher pair is better - so for example 6-6-4-3-2 beats 5-5-A-K-Q. If the pairs are equal, compare the highest ranking odd cards from each hand; if these are equal compare the second highest odd card, and if these are equal too, compare the lowest odd cards. So J-J-A-9-3 beats J-J-A-8-7 because the 9 beats the 8.

#### 1.2.10 High card



Five cards which do not form any of the combinations listed above. When comparing two such hands, the one with the better highest card wins. If the highest cards are equal, the second cards are compared; if they are equal too, the third cards are compared, and so on. So A-J-9-5-3 beats A-10-9-6-4 because the jack beats the ten.

## 2 SOLUTION

### 2.1 Represent a card

Each card is represented by an id number. There is 52 cards in a poker pack, so the id number is from 1 to 52.

Each card has the value and suite. The suit of card is from 1 to 4 correspond the suits:

1 – Heart, 2 – Diamond, 3 – Club, 4 – Spade.

To define the suit of card, we based on the id of card.

---

**GetSuit()**

---

```
if id>0 and id <14
    then return 1;
if id>13 and id <27
    then return 2;
if id>26 and id <40
    then return 3;
if id>39 and id <53
    then return 1;
```

---

The value of card is from 1 to 13 correspond the card from ACE, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King.

To define the value of card, we based on the result of operator id mode 13.

---

**getValue()**

---

```
if (id mod 13 != 0)
    then return (id mod 13);
else return 13;
```

---

If the id of card is outside of from 1 to 52, it is not an valid id. Then the suit and value are 0.



## 2.2 Represent a set cards

To represent a set cards, we use the set id of cards. A set of cards can:

- Be an empty set.
- Be an invalid set cards if it has two cards are the same id.
- Has two cards are the same value.
- Has three cards are the same value.
- Has an ACE card.
- Has a King card.

The algorithms for defining the properties of a set cards A:

---

**isEmpty()**

---

```
if card(A) = 0
    then return TRUE;
else return FALSE;
```

---

---

**isSetCard()**

---

```
if A is empty, return TRUE;
else {
    x,y belong A;
    if (x = y)
        then return FALSE;
}
return TRUE;
```

---

---

**isPair()**

---

```
if card(A) <2 or A is an invalid set, return FALSE;
else {
    x,y belong A;
    x is different y;
    if x and y are the same value, return TRUE;
}
```

---

---

**isThree()**

---

```
if card(A)<3 or A is an invalid set, return FALSE;
else {
```

---

```
x,y,z belong A;  
x,y,z are different with other card  
if x,y,z are the same value, return TRUE;  
}
```

---

---

**isA()**

```
if A is empty set or A is an invalid set, return  
FALSE;  
else {  
    x belong A;  
    if the value of x is 1, return TRUE;  
}
```

---

---

**isK()**

```
if A is empty set or A is an invalid set, return  
FALSE;  
else {  
    x belong A;  
    if the value of x is 13, return TRUE;  
}
```

---

Moreover, we can get the max value of set cards and the card has the max value.

---

**GetMax()**

```
if A is empty set or A is an invalid set, return 0;  
else{  
    x belong A;  
    if the value of x is the highest in A, return the  
value of x.  
}
```

---

---

**getMaxCard()**

```
if A has the max value,  
x belong A;  
if the value of x is the highest in A, return x.
```

---

Finally, we can compare two set cards A and B. The result of comparing can be: -1, 0, 1, 2 are Error, fair, A win, B win.

---

**compareSetCard(A,B)**


---

```

if A and B are different size, return -1;
if A or B is an invalid set, return -1;
if A and B are empty set, return 0;
if A and B are the valid sets and the same size,
    if A has an ACE card but B has not, return 1;
    if A has not an ACE card but B has , return 2;
    if both A and B has or has not an ACE,
        if the max value of A is higher than the max
value of B, return 1;
        if the max value of A is less than the max value
of B, return 2,
    else {
        C = A \ {the max card};
        D = B \ {the max card};
        return compareSetCard(C,D);
    }

```

---

## 2.3 Represent a hand

If a valid set card has exactly 5 cards, it is a valid hand.

---

**IsHand()**


---

```

if A is a valid set and card(A) = 5,
    then return TRUE;

```

---

Beside the properties of a set cards, a hand can :

- Has four of a kind.
- is flush: all cards are the same suit but are not in sequence.
- is straight: Five cards of mixed suits but are in sequence.

---

**isFour()**


---

```

if A is an invalid hand, return FALSE;
else{
    x,y,z,t belong A;
    x,y,z,t are difference with other card,
    if x,y,z,t are the same value,
        then return TRUE;}

```

---

**isFlush()**

---

```
if A is an invalid hand, return FALSE;
else {
    x,y belong A;
    x is difference y,
    if x is not the same suite with y,
        then return FALSE;
}
```

---

---

**isStraight()**

---

```
if A is an invalid hand, return FALSE;
else {
    A ={x,y,z,t,w}
    if x,y,z,t,w are in sequence,
        then return TRUE;
    if the value of cards in A is {1,10,11,12,13},
        then return TRUE;
}
```

---

We base on the properties of a hand to define the rank. The rank of hand can be from 1 to 10 correspond from High card to Royal Flush.

---

**GetRank()**

---

```
if A is an invalid hand, return -1;
else{
    if A has an ACE card, a King card and A is
straight, flush,
    then A is Royal Flush, return 10;
    if A is straight, flush and has not an ACE card or
a King card,
    then A is Straight Flush, return 9;
    if A is four of a kind, then return 8;
    if A consist of three cards of one value and two
cards of another value, then A is Full house,
return 7;
    if A is straight but A is not flush,
```

```

    then A is straight, return 6;
    if A is flush but A is not straight,
        then A is flush, return 5;
    if A consist of three cards of one value, and two
other cards,
        then A is three of a kind, return 4;
    if A has two pairs of a kind difference value, and
other card,
        then A is two pair, return 3;
    if A has a pair of a kind, and three other cards,
        then A is one pair, return 2;
    if A has not a pair of a kind and A is not flush,
straight,
        then A is high card, return 1;
}

```

---

## 2.4 Represent a fighting

A fighting has two valid hands. Also, there is no two cards are same id. To compare two hand, we compare the rank of them, if they are the same rank, we compare them follow the rules are presented above.

To check a fighting is or is not valid:

---

**validFighting(A,B)**

---

```

if A or B is not a valid hand, return FALSE;
if the set cards consist of A and B is not unique,
return FALSE;
else return TRUE

```

---

In the first , we compare the rank of two hand. The result of the comparision can be -1, 0, 1, 2 correspond to cases error, fair, A win, B win.

---

**beatRank ()**

---

```

if validFighting(A,B) return -1;
else {
    if the rank of A is higher than the rank of B,

```

```
    then A win B, return 1;
  if the rank of A is lower than the rank of B,
    then B win A, return 2;
  if A and b have the same rank,
    then A equal to B, return 0;
}
```

---

If the rank of A and B are the same, we continue the comparison follow the rules are represented above. After that, we have the final result of the fighting.

---

**beat (A,B)**

---

```
if beatRank(A,B) != 0,
  then return beatRank(A,B);
else {
  if A and B are Royal,
    then A equal to B, return 0;
  if A and B are Straigh Flush,
    the result of beating is the result of comparison
    between the max card of A with the max card of B.
  if A and B are Four of a kind,
    the result of beating is the result when
    comparing the value of the four of A with the
    value of the four of B.
  if A and B are full house or three of a kind,
    the result of fighting is the result when
    comparing the value of the three of A with the
    value of the three of B.
  if A and B are straight,
    the result of beating is the result of comparison
    between the max card of A with the max card of B.
  if A and B are flush or high card,
    the result of fighting is the result when
    comparing the set A with the set B.
  if A and B are two pair of a kind,
    if the two pairs of A are the same value with two
    pairs of B, then the winner is the hand has
    the last card with the value higher.
    else then the winner is the hand has the value of
    a pair higher.
```

```
    if A and B are one pair,  
        if pair of a is the same value with pair of b,  
            then the result of fighting is the result  
            when comparing the rest of cards of a with  
            the rest of cards of b.  
        else then the winner is the hand has the pair  
            with the value higher.  
}
```

---

In addition, we can compare two hands by the transitivity of ordering between hands. If A beats B and B beats C, then A beats C.

---

**beat(A,B,C)**

---

```
if A beats B, and B beats C, and A and C compose a  
valid fighting,  
    then A beats C, return 1;
```

---

### 3 APPLY IN RODIN AND IMPLEMENT IN JAVA

#### 3.1 Apply in Rodin

Based on the part 2. SOLUTION, we apply in Rodin platform. We use 4 Event-B context:

- Card: describe a card.
- Poker: describe set cards.
- Ranking: describe a hand card.
- Fighting: describe a fighting.

All Event-B context specifications are represented detail in the documentations of Rodin.

#### 3.2 Implement in Java programming language

We use 4 Java class to represent for 4 Event-B context in Rodin.

- The Card class correspond with the Card Event – B context.
- The SetCard class correspond with the Poker Event-B context.
- The Hand class correspond with the Ranking Event-B context.
- The Fighting class correspond with the Fighting Event-B context.

All class specifications are represented detail in the java doc of Poker project.

#### 3.3 Test

We use the JUnit test to test the implementation in java programming language. There are four test classes correspond to the four implemented classes:

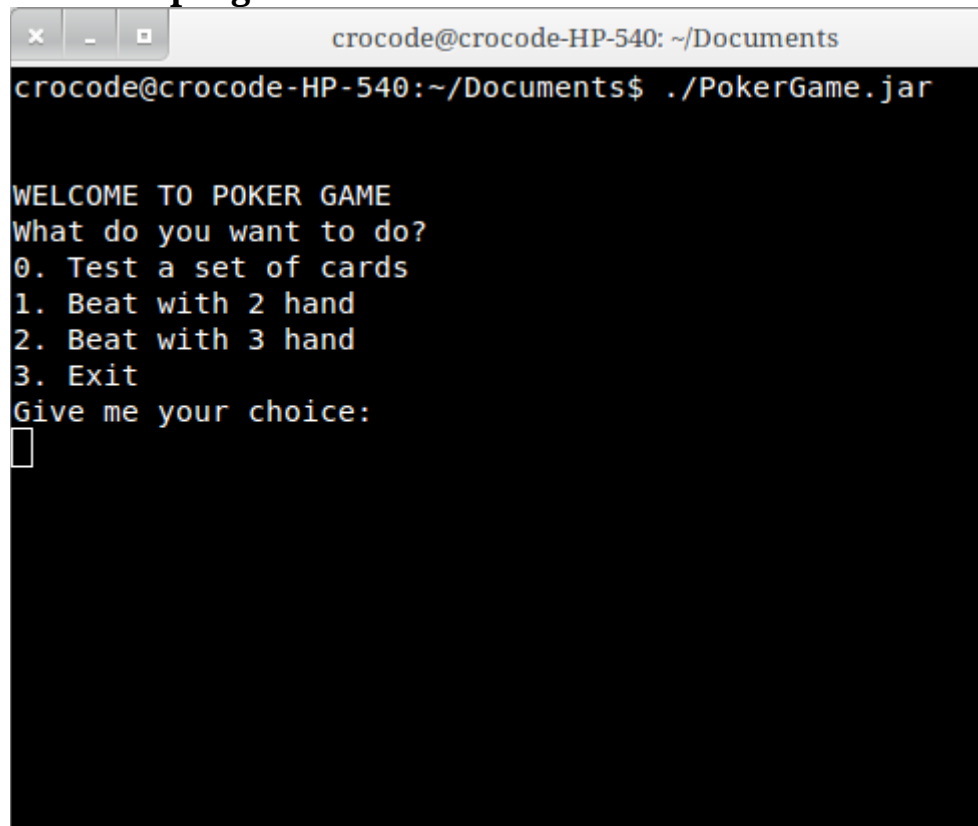
- testCard.java - Card.java
- testHand.java – Hand.java
- testSetCard.java – SetCard.java
- testFighting.java – Fighting.java

Moreover, we have a test suite called testSuit.java include four test case classes.

All the result of testing are represented in the excel file **TestPoker.ods**



### 3.4 Mini program



```
crocodile@crocodile-HP-540: ~/Documents
crocodile@crocodile-HP-540:~/Documents$ ./PokerGame.jar

WELCOME TO POKER GAME
What do you want to do?
0. Test a set of cards
1. Beat with 2 hand
2. Beat with 3 hand
3. Exit
Give me your choice:
█
```

We have a mini program **PokerGame.jar** . We can check the properties of a set cards, or do a fighting with two or three hands.