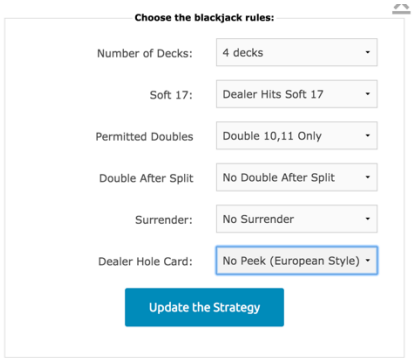
物件導向期末戰略報告

資管二乙 B0544243 鄒涵如

（一）物件導向要牌機制：



<附圖一>

4 decks, H17, D10, No DAS, No Surrender, No Peek

Estimated casino edge for these rules: 1.01 %

Hard Totals

Your Hand	Dealer 2	3	4	5	6	7	8	9	T	A
Hard 5	H	H	H	H	H	H	H	H	H	H
Hard 6	H	H	H	H	H	H	H	H	H	H
Hard 7	H	H	H	H	H	H	H	H	H	H
Hard 8	H	H	H	H	H	H	H	H	H	H
Hard 9	H	H	H	H	H	H	H	H	H	H
Hard 10	D	D	D	D	D	D	D	D	H	H
Hard 11	D	D	D	D	D	D	D	D	H	H
Hard 12	H	H	S	S	S	H	H	H	H	H
Hard 13	S	S	S	S	S	H	H	H	H	H
Hard 14	S	S	S	S	S	H	H	H	H	H
Hard 15	S	S	S	S	S	H	H	H	H	H
Hard 16	S	S	S	S	S	H	H	H	H	H
Hard 17	S	S	S	S	S	S	S	S	S	S
Hard 18+	S	S	S	S	S	S	S	S	S	S

<附圖二>

Soft Totals

Your Hand	Dealer 2	3	4	5	6	7	8	9	T	A
Ace + 2	H	H	H	H	H	H	H	H	H	H
Ace + 3	H	H	H	H	H	H	H	H	H	H
Ace + 4	H	H	H	H	H	H	H	H	H	H
Ace + 5	H	H	H	H	H	H	H	H	H	H
Ace + 6	H	H	H	H	H	H	H	H	H	H
Ace + 7	S	S	S	S	S	S	S	H	H	H
Ace + 8	S	S	S	S	S	S	S	S	S	S
Ace + 9	S	S	S	S	S	S	S	S	S	S

<附圖三>

以上機制為參考 BLACKJACK BASIC STRATEGY ENGINE，最接近此次遊戲規則的要牌策略。

以下轉換為程式碼後：

```
if(hasAce()==false){
    /*if(getTotalValue()==12&& (table.get_face_up_card_of_dealer().getRank()==4||table.get_face_up_card_of_dealer().getRank()==5||table.get_face_up_card_of_dealer().getRank()==6)){
    return false;
    }
    if(getTotalValue()==13&&(table.get_face_up_card_of_dealer().getRank()==2||table.get_face_up_card_of_dealer().getRank()==3||table.get_face_up_card_of_dealer().getRank()==4||table.get_face_up_card_of_dealer().getRank()==5||table.get_face_up_ca
    return false;
    }
    if(getTotalValue()==14&&(table.get_face_up_card_of_dealer().getRank()==2||table.get_face_up_card_of_dealer().getRank()==3||table.get_face_up_card_of_dealer().getRank()==4||table.get_face_up_card_of_dealer().getRank()==5||table.get_face_up_ca
        return false;
    }*/
    if(getTotalValue()==15&&(table.get_face_up_card_of_dealer().getRank()==2||table.get_face_up_card_of_dealer().getRank()==3||table.get_face_up_card_of_dealer().getRank()==4||table.get_face_up_card_of_dealer().getRank()==5||table.get_face_up_ca
    return false;
    }
    else if(getTotalValue()==16&&(table.get_face_up_card_of_dealer().getRank()==2||table.get_face_up_card_of_dealer().getRank()==3||table.get_face_up_card_of_dealer().getRank()==4||table.get_face_up_card_of_dealer().getRank()==5||table.get_face_u
    return false;
    }
    else if(getTotalValue()>=17){
    return false;
    }
    else{
        return true;
    }
}
else { //hasAce()==true
    if(getTotalValue()>=19){
        return false;
    }
    else if(getTotalValue()==18&&(table.get_face_up_card_of_dealer().getRank()==2||table.get_face_up_card_of_dealer().getRank()==3||table.get_face_up_card_of_dealer().getRank()==4||table.get_face_up_card_of_dealer().getRank()==5||table.get_face_u
        return false;
    }
    else
        return true;
}
```

邏輯為假如沒有 A ，原本依照附圖二，從總數 12 就必須要停牌，但覺得太過消極，且試驗後從總數為 15 開始停牌，贏得局數的比例感覺較高。

假如手牌中有 A 的話，則依照附圖三的要牌原則要牌。判斷 A 為 1 或 11，對於要牌策略並沒有影響很大，加上測試過後發現，若分開判斷要牌的話，跟同一個人比的話反而會輸，最後決定保持原樣。

在 hit_me()這個程序之中，因為可以傳 Table 值，所以除了判斷是否要牌外，也另外做了很多事。

1. 找出對手的位置

   找對手位置可以知道他的下注金額或他現在擁有的籌碼。

```
public boolean hit_me(Table table){
    Player[] MyPlayers = table.get_player();
    for(int i=0;i<2;i++){
        if(MyPlayers[i].get_name()!="鄒涵如_B0544243"){
            Pset=i;
        }
    }
    OtherChip=MyPlayers[Pset].get_current_chips();
```

只有 2 位玩家，所以名字不是自己即是對手。變數 OtherChip 存對手的籌碼，以供下注使用，雖然是上一局的籌碼，但有一定參考價值，待下注方法再解釋。

2. 計算是否為最後一局(不採用)

```
if (table.getPercentofUsedCard() >= 0.72115385){//用到0.75時會停一場會有6張牌
     LastGame= true;}
```

在牌剩下 0.75 的時候會停牌，六張牌在 4 副的機率是 0.2884615(=6/208)，所以假如牌只剩最後一局的話，就會全下。發現因為 hitme 跟 makebet 有時間差，當發現為最後一場時早就已下注完畢。

3. **計算 TrueCount**(不採用)

算牌的方法其中一樣稱為 High Low 法：根據數學統計，當牌盒中剩的牌佔多時，莊家若為 12～16 便容易爆牌，玩家平均下來會贏。

將十三張牌分為三組。高點牌（Hi）T、J、Q、K 及 A (1 deck 20 張)，中性牌七、八、九，及低點牌（Lo）二至六 (1 deck 20 張)。高點牌之參數為減一（-1），低點牌為加一（+1），中性牌為 0。將這些點數加總起來被稱為流水數（Running Count），將流水數除以剩下的牌數就為真數（TrueCount）這個真數可以判斷目前勝率大概是多少？

在 hit_me()中的程式碼：

```
MyUsedCard=table.getOpenedCards();
count();
TrueCount=RunningCount/((208-table.getOpenedCards().size())/52);
```

計算流水數的另外一個程序：

```
public void count(){
    for(Card card:MyUsedCard){
        if(card.getRank()==1||card.getRank()==10||card.getRank()==11||card.getRank()==12||card.getRank()==13){
            RunningCount--;
        }
        else if(card.getRank()==2||card.getRank()==3||card.getRank()==4||card.getRank()==5||card.getRank()==6){
            RunningCount++;
        }
```

原本有打算使用此算牌方法，考慮下注，但這個方法因此次遊戲規則，並不準確，所以不採用，建議下注時也傳入 table 值會較為準確。

（二）物件導向下注機制：

1.所使用到的變數：

```
private int PreChip =1000;
private int winGame=0;
private int b=200;
private int OtherChip=0;
```

PreChip 為上一場的籌碼，與這場籌碼相比，判斷上一場是贏或輸。

WinGame 判斷是否連贏。

B 為每次下注的基本籌碼。

OtherChip 為對手的籌碼。

程式碼為簡單判斷是否連贏：

```
if(get_current_chips()>=PreChip){
    winGame++;//一進場就=1
    lostGame=0;
}
else{
    lostGame++;
    winGame=0;
}
```

若這場籌碼大於上一場則贏否則輸。

下注戰略：

```
if(winGame<=1){
    setBet(b);
}
else if(winGame==2){//贏第一場
    setBet(b*2);
}
else if(winGame==3){//贏第二場
    setBet((int)(b*2.5));
    winGame=1;//若繼續贏則重置
}
```

最一開始會下 b=200 若連勝則為 b*2=400 若三連勝則 b*2.5=500，三連勝後則重置，但因為能夠重置表示前面已經三連勝過，所以會希望籌碼加大，故 winGame 會由 2 開始（重置 1+這場勝贏 1）但上一場輸 winGame 會變為 0，則由最一開始 b=200 下注。

比例的想法是，連贏 3 場的機率較低，第三場輸的機率較高，前兩場贏的籌碼為 b*3，第三場輸的籌碼為 b*2.5，理論上還是籌碼還是正的。

輸太多就賭一把：

```
if(OtherChip>(get_current_chips()+2500)){
    b=(OtherChip-get_current_chips());
    while(b>get_current_chips()){
        b=b/2;
    }
    setBet(b);
}
else{
    b=200;
    }
```

差超過 2500，對我第三場才下 500 來說是很大一筆數字，很難挽回，下一把與對手差距的籌碼賭賭看，超過自身籌碼的話就除以 2，若贏了則可以填補與對手之間籌碼的差距。在 make_bet()的最上方會讓 b=200，重新跑原本的判斷。

最後檢查：

```
    if(get_current_chips()<400||getBet()==get_current_chips()||getBet()>get_current_chips()){
        setBet((int)(get_current_chips()*0.1));
        return getBet();
    }
    else{
        return getBet();
    }
}
```

假如籌碼小於 400 塊，若一次下 200 很有可能輸光籌碼，下注也不能高於現有的籌碼，並且不是最後一局的話，就改每次下籌碼的 10%。

其他則照原本的判斷下注。

補充：下注程式碼由上到下，若後面的程式碼會覆蓋前面的設定結果。