2021 OOPI

Group39_Project3

高雄大學資訊管理學系 113 級

A1093355 彭郁庭　　A1093367 陳俐卉

目錄

壹、 需求描述 1

貳、 功能邏輯說明 2

- Class Diagram 2
- 流程圖 ..... 2~5
- 程式碼 ..... 6~13

參、 系統架構 14

肆、 使用說明 15

伍、 實際執行結果 16

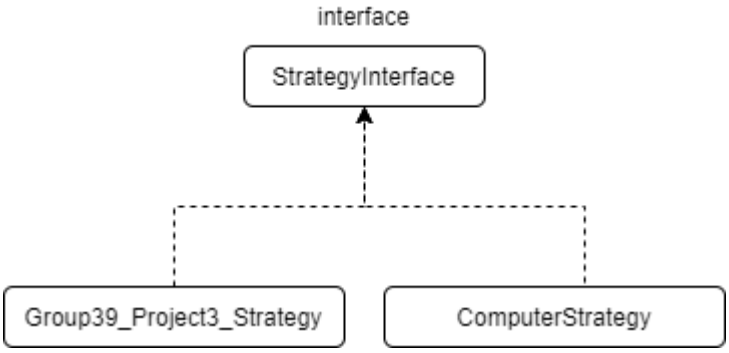
陸、 其他(延伸討論)..... 17~18

壹、 需求描述

此次作業要求將出牌邏輯實作成介面(StrategyInterface.java)並於遊戲進行流程(Threads.java)使用，並印出要求之遊戲資訊，另外需將出牌邏輯設計得更加完整。用 Threads.java 讓 RoundPlayer 保持在 0~3 之間循環，跳過已經死亡的玩家，並利用 Strategy.java 中的 auto_run()執行每個 Roundplayer 的行動，整合 fold()與 useCard()對選定的對象出牌。

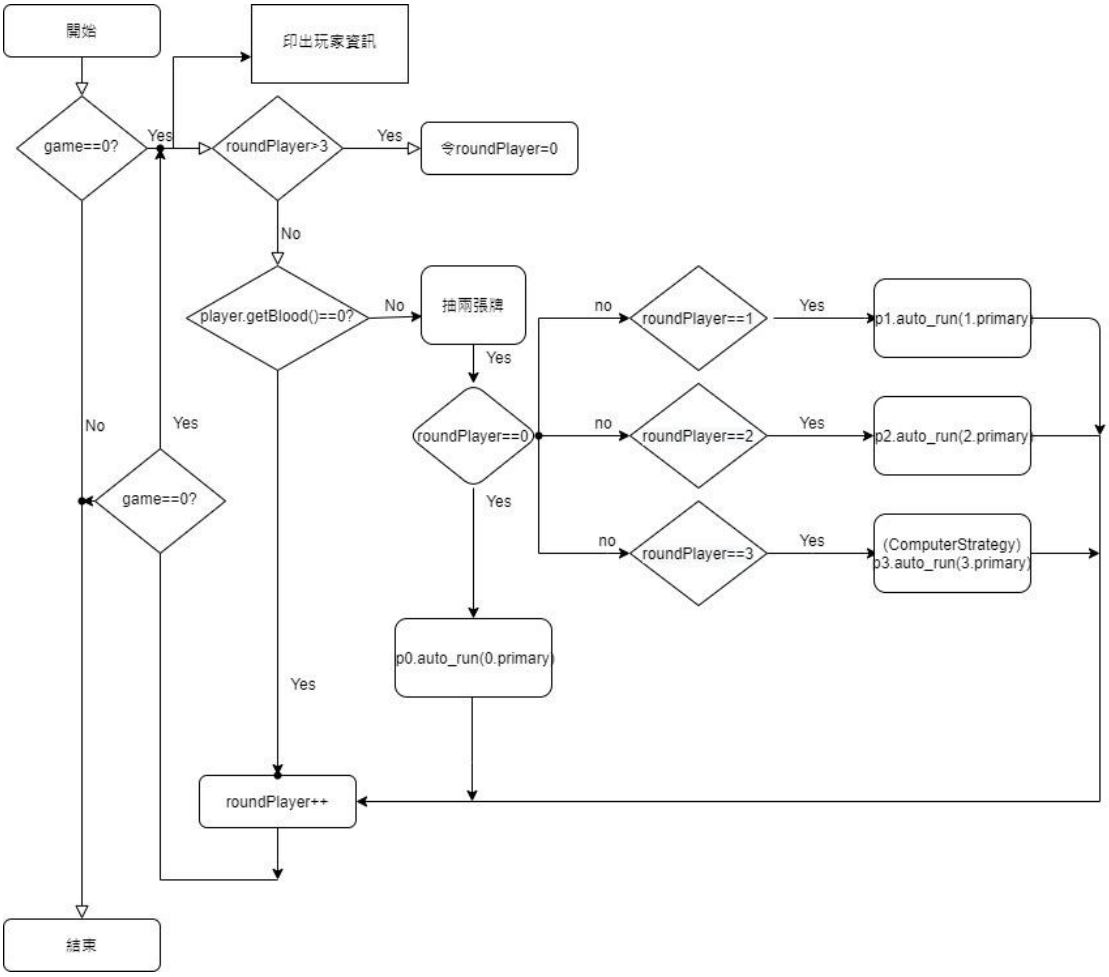
貳、 功能邏輯說明

Class Diagram

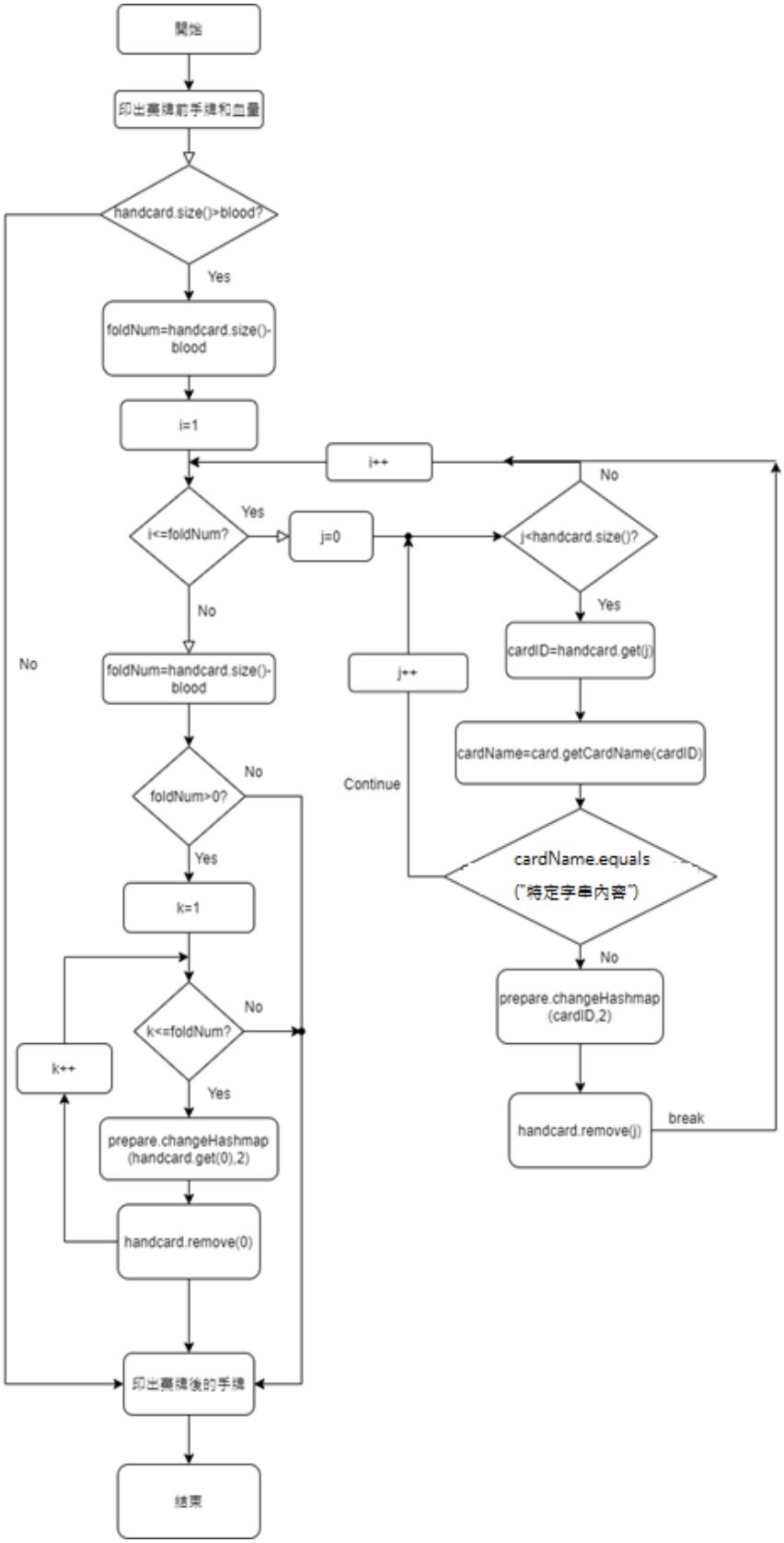


流程圖

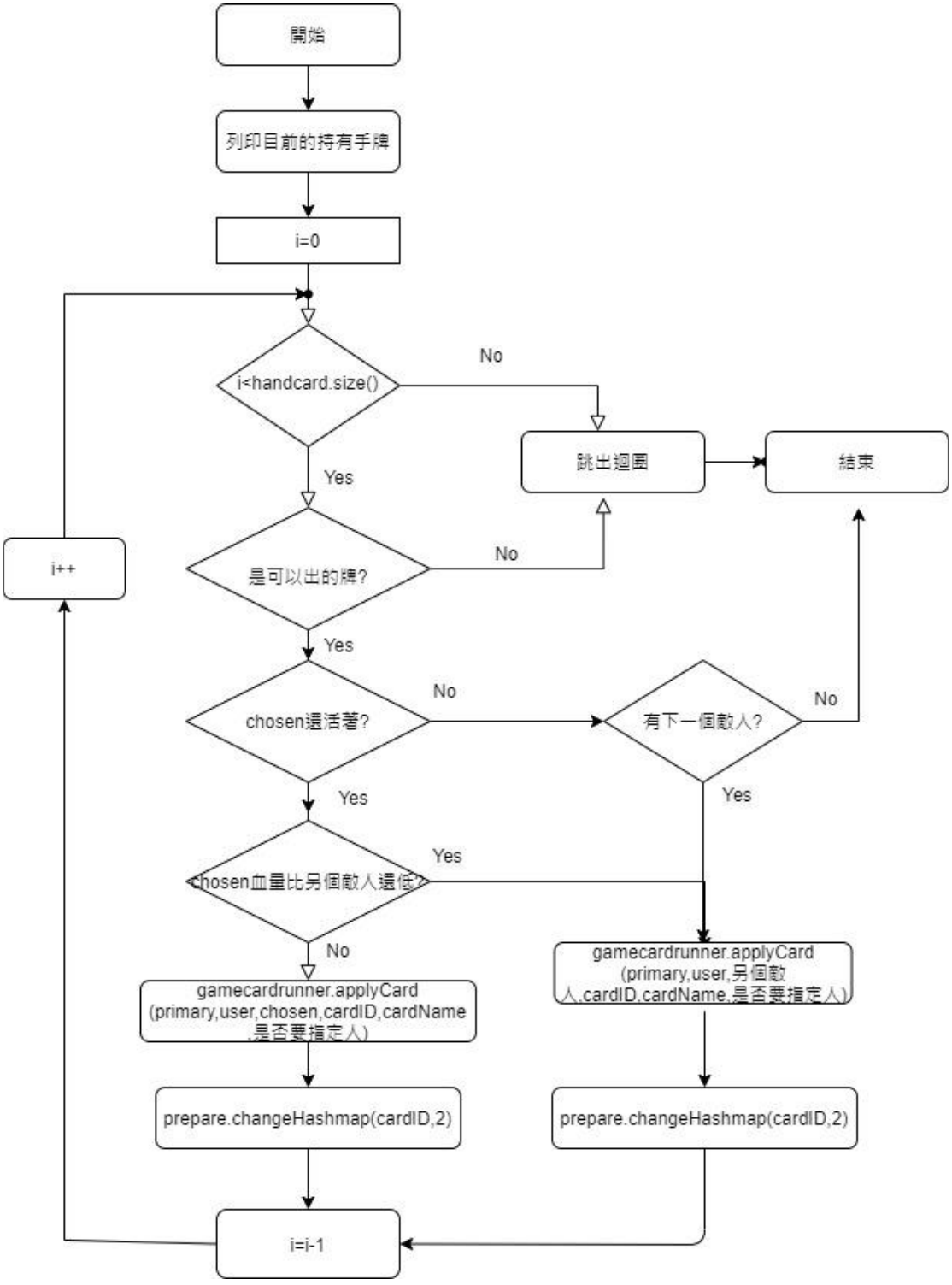
Threads.java



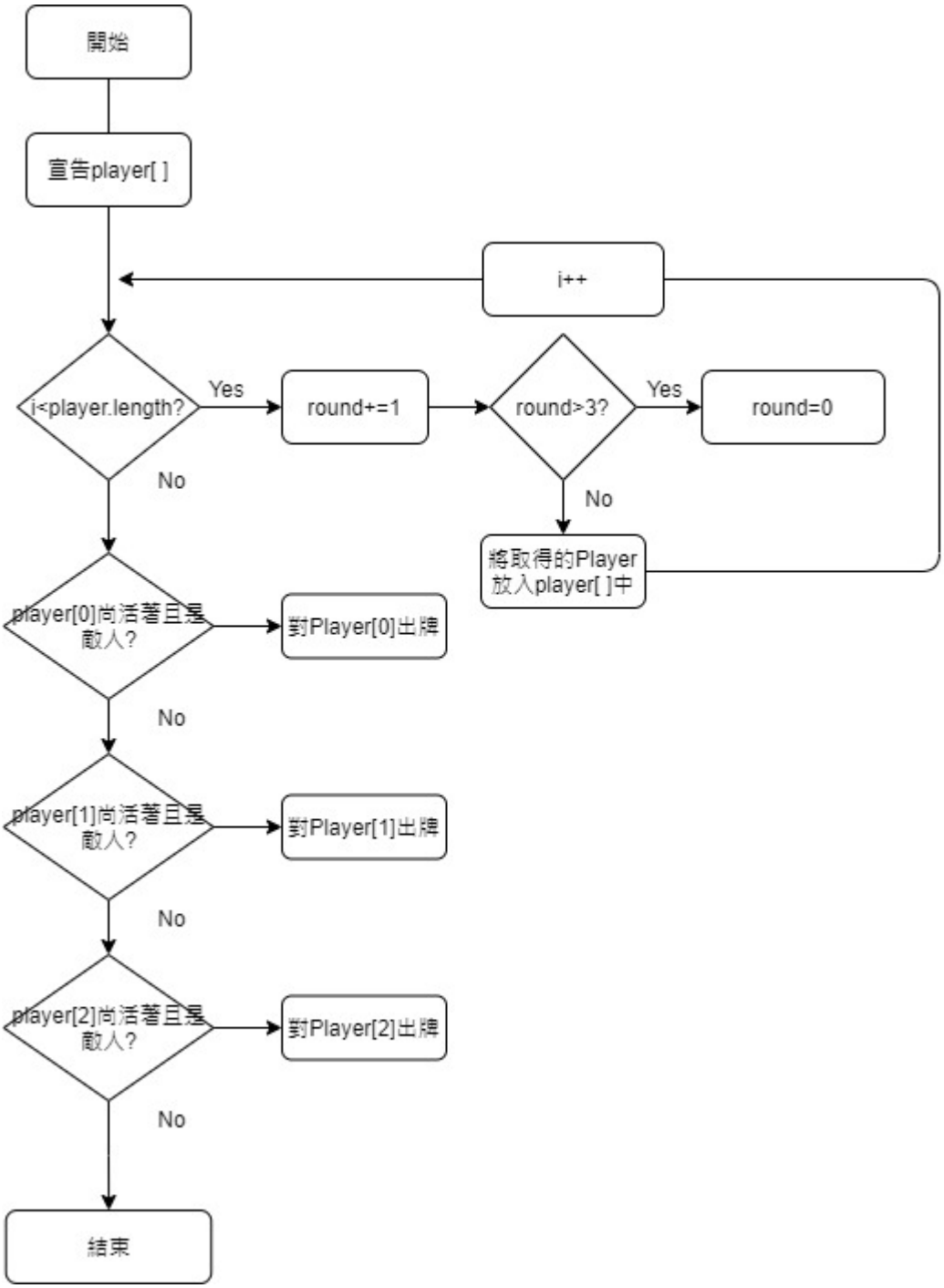
fold() 棄牌流程


開始
印出棄牌前手牌和血量
handcard.size()>blood?
Yes
foldNum=handcard.size()-blood
i=1
i++
No
i<=foldNum?
Yes
j=0
j<handcard.size()?
Yes
No
cardID=handcard.get(j)
No
foldNum=handcard.size()-blood
j++
cardName=card.getCardName(cardID)
foldNum>0?
No
Continue
Yes
cardName.equals ("特定字串內容")
k=1
No
prepare.changeHashmap (cardID,2)
k<=foldNum?
No
k++
Yes
handcard.remove(j)
break
prepare.changeHashmap (handcard.get(0),2)
handcard.remove(0)
印出棄牌後的手牌
結束

useCard() 出牌流程



Strategy.java 中的 auto_run()流程



程式碼

- **Group39_Project3_Game.java:**透過 main 主程式執行整個 project3 的內容。

```java
// This is the main class of checkpoint this time.
public class Group39_Project3_Game{
    Run | Debug
    public static void main(String[] args){
        Group39_Project3_Threads thread = new Group39_Project3_Threads(args[0], args[1], args[2],args[3]);
        Thread start = new Thread(thread);
        start.start();
    }
}
```

- **Group39_Project3_Threads.java:**讓 roundPlayer 保持在 0~3 之間循環，並且驅動 Roundplayer 使用出牌、棄牌等行動。

run() method

```java
    //Implement the run function
    public void run(){
        //initialize the game to Integer type
        int game =0;
        //intitalize the playerid start with 0
        int roundPlayer=0;
        //let the while loop keep running
        while(game==0){

            if(roundPlayer>3)
                    roundPlayer=0;
            Player player=primary.getPlayer(roundPlayer);
            System.out.println("**********");
            System.out.println("now playing: player "+player.getPlayerId());
            System.out.println("blood: "+player.getBlood());
```

39.若 game=0 時，跑迴圈

41.~42.若 roundPlayer 大於 3，則令 roundPlayer 歸 0

43.取用此局 roundPlayer 的資料並設給 Player 物件的 player

44.~46.在螢幕上印出玩家資料和血量

```java
            if(player.getBlood()!=0){
                ArrayList<Integer> hC=player.getHandCard();
                for(int element:prepare.drawCard(2)){
                    hC.add(element);
                }
                player.setHandCard(hC);
                if(roundPlayer==0/*player.getPlayerId()==0*/){
                    Player player1=primary.getPlayer(roundPlayer+1);
                    Player player2=primary.getPlayer(roundPlayer+2);
                    Player player3=primary.getPlayer(roundPlayer+3);
                    strategy=loadStrategy(inputStrategys[0],player,player1,player2,player3);
                    strategy.auto_run(0,primary);
                }
```

47.若 Player 的血量不等於 0，做{}

48.宣告 ArrayList 的 hC 來暫存開局的手牌

49.~50.透過 For-each 語法將新抽的兩張牌加進手牌中

52.更新手牌

53.若 roundPlayer 等於 0

54.~56.設置其他玩家

57.實作 StrategyInterface 設置 Strategy

58.使用 auto_run 的方法讓玩家 0 做出實際行動

```
60                  else{
61                      if(roundPlayer==1){
62                          Player player1=primary.getPlayer(roundPlayer+1);
63                          Player player2=primary.getPlayer(roundPlayer+2);
64                          Player player3=primary.getPlayer(roundPlayer-1);
65                          strategy=loadStrategy(inputStrategys[1],player,player1,player2,player3);
66                          strategy.auto_run(1,primary);
67                      }
68                      else if(roundPlayer==2){
69                          Player player1=primary.getPlayer(roundPlayer+1);
70                          Player player2=primary.getPlayer(roundPlayer-1);
71                          Player player3=primary.getPlayer(roundPlayer-2);
72                          strategy=loadStrategy(inputStrategys[2],player,player1,player2,player3);
73                          strategy.auto_run(2,primary);
74                      }
75                      else if(roundPlayer==3){
76                          Player player1=primary.getPlayer(roundPlayer-1);
77                          Player player2=primary.getPlayer(roundPlayer-2);
78                          Player player3=primary.getPlayer(roundPlayer-3);
79                          strategy=loadStrategy(inputStrategys[3],player,player1,player2,player3);
80                          strategy.auto_run(3,primary);
81                      }
82                  }
83              }
84              roundPlayer++;
85
86              doNothing(1000);
87          }
88
89      }
```

61.若 roundPlayer 等於 1

62.~64.設置其他玩家

65. 實作 StrategyInterface 設置 Strategy

66.使用 auto_run 的方法讓玩家 1 做出實際行動

68.若 roundPlayer 等於 2

69.~71.設置其他玩家

72.實作 StrategyInterface 設置 Strategy

73.使用 auto_run 的方法讓玩家 2 做出實際行動

75.若 roundPlayer 等於 3

76.~78.設置其他玩家

79.實作 StrategyInterface 設置 Strategy

80.使用 auto_run 的方法讓玩家 3 做出實際行動

84. roundPlayer+1

Group39_Project3_Strategy.java:

實作出牌和棄牌的 class。

棄牌:

```
18      public void fold(){
19          int blood=user.getBlood();
20          System.out.println("Hancard before I fold : "+handcard);
21          System.out.println("My blood : "+blood);
22          if(handcard.size()>blood){
23              int foldNum=handcard.size()-blood;
24              for(int i=1;i<=foldNum;i++){
25                  for(int j=0;j<handcard.size();j++){
26                      int cardID=handcard.get(j);
27                      String cardName=card.getCardName(cardID);
28                      if(cardName=="Miss"||cardName=="Gig Economy"||cardName=="Fire!"/*||cardName=="Bar"*/){
29                          continue;
30                      }
31                      prepare.changeHashmap(cardID,2);
32                      handcard.remove(j);
33                      break;
34                  }
35              }
36              foldNum=handcard.size()-blood;
37              if(foldNum>0){
38                  for(int i=1;i<=foldNum;i++){
39                      for(int j=0;j<handcard.size();j++){
40                          int cardID=handcard.get(j);
41                          String cardName=card.getCardName(cardID);
42                          if(cardName=="Miss"){
43                              continue;
44                          }
45                          prepare.changeHashmap(cardID,2);
46                          handcard.remove(j);
47                          break;
48                      }
49                  }
50              }
51              foldNum=handcard.size()-blood;
52              if(foldNum>0){
53                  for(int k=1;k<=foldNum;k++){
54                      prepare.changeHashmap(handcard.get(0),2);
55                      handcard.remove(0);
56                  }
57              }
58          }
59          System.out.println("Hancard after I fold : "+handcard);
60      }
```

19.設置 user 血量

20.印出 user 棄牌前的手牌 ID

21.印出 user 血量

22.若 user 手牌總數大於血量

23.宣告 foldNum 為需棄牌數量

24.~25.用巢狀迴圈來判斷是否該棄第 j 張牌，用 i 來數需棄牌總數

26.宣告 cardID 為第 j 張手牌

27.取得此張牌的卡牌名稱

28.~30.判斷是否為"Miss"或"Fire"或補血的牌，是的話繼續內層迴圈判斷下一張牌

31.否則將這張牌放入棄牌堆

32.從手牌中移出

33.跳出內層迴圈，回到外層迴圈繼續判斷是否該繼續棄牌

36.重新取得需棄牌數量

37.~50.若棄牌數量>0，則跳過 miss 棄下一張牌

51. 重新取得需棄牌數量

52.~58.依尚需棄牌數量，每次棄掉排序為 0 的牌

59.印出棄牌後的手牌 ID

出牌:

```
62      public void useCard(Player chosen,Primary primary){
63          int handcardNum=user.HandCardNum();
64          Player[] player=new Player[3];
65          int round=user.getPlayerId();
66          for(int i=0;i<player.length;i++){
67              round+=1;
68              if(round>3)
69                  round=0;
70              player[i]=primary.getPlayer(round);
71          }
72          System.out.println("Cards in my hand:");
73          for(int cardID:handcard)
74              System.out.print(card.getCardName(cardID)+"; ");
75          System.out.print("\n");
76          int enemyblood=chosen.getBlood();
77          int blood=user.getBlood();
78          for(int i=0;i<handcard.size();i++){
79              enemyblood=chosen.getBlood();
80              blood=user.getBlood();
81              int cardID=handcard.get(i);
82              String cardName=card.getCardName(cardID);
83              if(cardName.equals("Tonight I Want Some")&&enemyblood>0){
84                  gamecardrunner.applyCard(primary,user,chosen,cardID,cardName,0);
85                  prepare.changeHashmap(cardID,2);
86                  i=i-1;
87              }
88              if(cardName.equals("Please Support Counter")&&enemyblood>0){
89                  gamecardrunner.applyCard(primary,user,chosen,cardID,cardName,0);
90                  prepare.changeHashmap(cardID,2);
91                  i=i-1;
92              }
93              if(cardName.equals("Different Pay For Equal Work")&&enemyblood>0){
94                  gamecardrunner.applyCard(primary,user,chosen,cardID,cardName,0);
95                  prepare.changeHashmap(cardID,2);
96                  i=i-1;
97              }
98          }
```

63.宣告 handcardNum 取得 user 手牌總數

64.~71.宣告 Player[] player 依序存放每位玩家的 Player

72.~75.用 for-each 迴圈印出玩家手中的所有牌

76.宣告 enemyblood 取得敵人血量

77.宣告 blood 取得 user 血量

78.用 for 迴圈依續檢查手牌是否符合出牌條件

79.每次都重新取得 enemyblood

80.每次都重新取得 blood

81.宣告 cardID 為第 i 張牌

82.取得此張牌的卡牌名稱

83.~87.若此張牌的名稱為” Tonight I Want Some”，出牌後放入棄牌堆，計數器 i-1

88.~92.若此張牌的名稱為” Please Support Counter”，出牌後放入棄牌堆，計數器 i-1

93.~97.若此張牌的名稱為” Different Pay For Equal Work”，出牌後放入棄牌堆，計數器 i-1

```
99          for(int i=0;i<handcard.size();i++){
100             enemyblood=chosen.getBlood();
101             blood=user.getBlood();
102             int cardID=handcard.get(i);
103             String cardName=card.getCardName(cardID);
104             if(cardName.equals("beibigennosuke")&&chosen.HandCardNum()>0&&enemyblood>0){
105                 if((player[0].getBlood()!=0)&&(player[0].getBlood()<enemyblood)&&(player[0].getRole()!=user.getRole()-2)
106                 &&(player[0].getRole()!=user.getRole()+2)&&(!player[0].equals(chosen))&&player[0].HandCardNum()>0){
107                     gamecardrunner.applyCard(primary,user,player[0],cardID,cardName,1);
108                     prepare.changeHashmap(cardID,2);
109                     i=i-1;
110                 }
111                 else if((player[1].getBlood()!=0)&&(player[1].getBlood()<enemyblood)&&(player[1].getRole()!=user.getRole()-2)
112                 &&(player[1].getRole()!=user.getRole()+2)&&(!player[1].equals(chosen))&&player[1].HandCardNum()>0){
113                     gamecardrunner.applyCard(primary,user,player[1],cardID,cardName,1);
114                     prepare.changeHashmap(cardID,2);
115                     i=i-1;
116                 }
117                 else if((player[2].getBlood()!=0)&&(player[2].getBlood()<enemyblood)&&(player[2].getRole()!=user.getRole()-2)
118                 &&(player[2].getRole()!=user.getRole()+2)&&(!player[2].equals(chosen))&&player[2].HandCardNum()>0){
119                     gamecardrunner.applyCard(primary,user,player[2],cardID,cardName,1);
120                     prepare.changeHashmap(cardID,2);
121                     i=i-1;
122                 }
123                 else{
124                     gamecardrunner.applyCard(primary,user,chosen,cardID,cardName,1);
125                     prepare.changeHashmap(cardID,2);
126                     i=i-1;
127                 }
128             }
```

99.用 for 迴圈依續檢查手牌是否符合出牌條件

100.每次都重新取得 enemyblood

101.每次都重新取得 blood

102.宣告 cardID 為第 i 張牌

103.取得此張牌的卡牌名稱

104.若此張牌的名稱為” beibigennosuke”，且敵人血量和手牌數量皆大於 0

105.~106.繼續判斷 enemyblood 是否低於下一位敵人的血量又符合出牌條件

107.~110.是的話，對下一位敵人出牌後將牌放入棄牌堆，計數器 i-1

111.~122.重複執行上面兩個步驟

123.~127.enemyblood 大於下一位敵人的血量則對 chosen 出牌，將牌放入棄牌堆，計數器 i-1

```
129             if(cardName.equals("beibigennosuke")&&enemyblood==0){
130                 if((player[0].getBlood()!=0)&&(player[0].getRole()!=user.getRole()-2)&&(player[0].getRole()!=user.getRole()+2)
131                 &&(!player[0].equals(chosen))&&player[0].HandCardNum()>0){
132                     gamecardrunner.applyCard(primary,user,player[0],cardID,cardName,1);
133                     prepare.changeHashmap(cardID,2);
134                     i=i-1;
135                 }
136                 else if((player[1].getBlood()!=0)&&(player[1].getRole()!=user.getRole()-2)&&(player[1].getRole()!=user.getRole()+2)
137                 &&(!player[1].equals(chosen))&&player[1].HandCardNum()>0){
138                     gamecardrunner.applyCard(primary,user,player[1],cardID,cardName,1);
139                     prepare.changeHashmap(cardID,2);
140                     i=i-1;
141                 }
142                 else if((player[2].getBlood()!=0)&&(player[2].getRole()!=user.getRole()-2)&&(player[2].getRole()!=user.getRole()+2)
143                 &&(!player[2].equals(chosen))&&player[2].HandCardNum()>0){
144                     gamecardrunner.applyCard(primary,user,player[2],cardID,cardName,1);
145                     prepare.changeHashmap(cardID,2);
146                     i=i-1;
147                 }
148             }
149         }
```

129.若此張牌的名稱為” beibigennosuke”，且 chosen 已死亡

130.~131.繼續判斷可否對下一位敵人出牌

132.~.135.可以的話，對下一位敵人出牌後將牌放入棄牌堆，計數器 i-1

136.~148.重複執行上面兩個步驟

```
150         for(int i=0;i<handcard.size();i++){
151             enemyblood=chosen.getBlood();
152             blood=user.getBlood();
153             int cardID=handcard.get(i);
154             String cardName=card.getCardName(cardID);
155             if(cardName.equals("awpjopabemarrrr")&&chosen.HandCardNum()>0&&enemyblood>0){
156                 if((player[0].getBlood()!=0)&&(player[0].getBlood()<enemyblood)&&(player[0].getRole()!=user.getRole()-2)
157                 &&(player[0].getRole()!=user.getRole()+2)&&(!player[0].equals(chosen))&&player[0].HandCardNum()>0){
158                     gamecardrunner.applyCard(primary,user,player[0],cardID,cardName,1);
159                     prepare.changeHashmap(cardID,2);
160                     i=i-1;
161                 }
162                 else if((player[1].getBlood()!=0)&&(player[1].getBlood()<enemyblood)&&(player[1].getRole()!=user.getRole()-2)
163                 &&(player[1].getRole()!=user.getRole()+2)&&(!player[1].equals(chosen))&&player[1].HandCardNum()>0){
164                     gamecardrunner.applyCard(primary,user,player[1],cardID,cardName,1);
165                     prepare.changeHashmap(cardID,2);
166                     i=i-1;
167                 }
168                 else if((player[2].getBlood()!=0)&&(player[2].getBlood()<enemyblood)&&(player[2].getRole()!=user.getRole()-2)
169                 &&(player[2].getRole()!=user.getRole()+2)&&(!player[2].equals(chosen))&&player[2].HandCardNum()>0){
170                     gamecardrunner.applyCard(primary,user,player[2],cardID,cardName,1);
171                     prepare.changeHashmap(cardID,2);
172                     i=i-1;
173                 }
174                 else{
175                     gamecardrunner.applyCard(primary,user,chosen,cardID,cardName,1);
176                     prepare.changeHashmap(cardID,2);
177                     i=i-1;
178                 }
179             }
```

150.用 for 迴圈依續檢查手牌是否符合出牌條件

151.每次都重新取得 enemyblood

152.每次都重新取得 blood

153.宣告 cardID 為第 i 張牌

154.取得此張牌的卡牌名稱

155.若此張牌的名稱為” awpjopabemarrrr”，且敵人血量和手牌數量皆大於 0

156.~157.繼續判斷 enemyblood 是否低於下一位敵人的血量又符合出牌條件

158.~161.是的話，對下一位敵人出牌後將牌放入棄牌堆，計數器 i-1

162.~173.重複執行上面兩個步驟

174.~178.enemyblood 大於下一位敵人的血量則對 chosen 出牌，將牌放入棄牌堆，計數器 i-1

```
180             if(cardName.equals("awpjopabemarrrr")&&enemyblood==0){
181                 if((player[0].getBlood()!=0)&&(player[0].getRole()!=user.getRole()-2)&&(player[0].getRole()!=user.getRole()+2)
182                 &&(!player[0].equals(chosen))&&player[0].HandCardNum()>0){
183                     gamecardrunner.applyCard(primary,user,player[0],cardID,cardName,1);
184                     prepare.changeHashmap(cardID,2);
185                     i=i-1;
186                 }
187                 else if((player[1].getBlood()!=0)&&(player[1].getRole()!=user.getRole()-2)&&(player[1].getRole()!=user.getRole()+2)
188                 &&(!player[1].equals(chosen))&&player[1].HandCardNum()>0){
189                     gamecardrunner.applyCard(primary,user,player[1],cardID,cardName,1);
190                     prepare.changeHashmap(cardID,2);
191                     i=i-1;
192                 }
193                 else if((player[2].getBlood()!=0)&&(player[2].getRole()!=user.getRole()-2)&&(player[2].getRole()!=user.getRole()+2)
194                 &&(!player[2].equals(chosen))&&player[2].HandCardNum()>0){
195                     gamecardrunner.applyCard(primary,user,player[2],cardID,cardName,1);
196                     prepare.changeHashmap(cardID,2);
197                     i=i-1;
198                 }
199             }
200         }
```

180.若此張牌的名稱為” awpjopabemarrrr”，且 chosen 已死亡

181.~182.繼續判斷可否對下一位敵人出牌

183.~186.可以的話，對下一位敵人出牌後將牌放入棄牌堆，計數器 i-1

187.~199.重複執行上面兩個步驟

```
201         for(int i=0;i<handcard.size();i++){
202             enemyblood=chosen.getBlood();
203             blood=user.getBlood();
204             int cardID=handcard.get(i);
205             String cardName=card.getCardName(cardID);
206             if(cardName.equals("Fire!")&& enemyblood>0)
207             {
208                 if((player[0].getBlood()!=0)&&(player[0].getBlood()<enemyblood)&&(player[0].getRole()!=user.getRole()-2)
209                 &&(player[0].getRole()!=user.getRole()+2)&&(!player[0].equals(chosen)))
210                 {
211                     gamecardrunner.applyCard(primary,user,player[0],cardID,cardName,1);
212                     prepare.changeHashmap(cardID,2);
213                     i=i-1;
214                 }
215                 else if((player[1].getBlood()!=0)&&(player[1].getBlood()<enemyblood)&&(player[1].getRole()!=user.getRole()-2)
216                 &&(player[1].getRole()!=user.getRole()+2)&&(!player[1].equals(chosen))){
217                     gamecardrunner.applyCard(primary,user,player[1],cardID,cardName,1);
218                     prepare.changeHashmap(cardID,2);
219                     i=i-1;
220                 }
221                 else if((player[2].getBlood()!=0)&&(player[2].getBlood()<enemyblood)&&(player[2].getRole()!=user.getRole()-2)
222                 &&(player[2].getRole()!=user.getRole()+2)&&(!player[2].equals(chosen))){
223                     gamecardrunner.applyCard(primary,user,player[2],cardID,cardName,1);
224                     prepare.changeHashmap(cardID,2);
225                     i=i-1;
226                 }
227                 else{
228                     gamecardrunner.applyCard(primary,user,chosen,cardID,cardName,1);
229                     prepare.changeHashmap(cardID,2);
230                     i=i-1;
231                 }
232             }
```

201.用 for 迴圈依續檢查手牌是否符合出牌條件

202.每次都重新取得 enemyblood

203.每次都重新取得 blood

204.宣告 cardID 為第 i 張牌

205.取得此張牌的卡牌名稱

206.若此張牌的名稱為”Fire!”，且敵人血量和手牌數量皆大於 0

208.~209.繼續判斷 enemyblood 是否低於下一位敵人的血量又符合出牌條件

210.~214.是的話，對下一位敵人出牌後將牌放入棄牌堆，計數器 i-1

215.~226.重複執行上面兩個步驟

227.~231.enemyblood 大於下一位敵人的血量則對 chosen 出牌，將牌放入棄牌堆，計數器 i-1

```
233             if(cardName.equals("Fire!") && enemyblood==0){
234                 if((player[0].getBlood()!=0)&&(player[0].getRole()!=user.getRole()-2)
235                 &&(player[0].getRole()!=user.getRole()+2)&&(!player[0].equals(chosen))){
236                     gamecardrunner.applyCard(primary,user,player[0],cardID,cardName,1);
237                     prepare.changeHashmap(cardID,2);
238                     i=i-1;
239                 }
240                 else if((player[1].getBlood()!=0)&&(player[1].getRole()!=user.getRole()-2)
241                 &&(player[1].getRole()!=user.getRole()+2)&&(!player[1].equals(chosen))){
242                     gamecardrunner.applyCard(primary,user,player[1],cardID,cardName,1);
243                     prepare.changeHashmap(cardID,2);
244                     i=i-1;
245                 }
246                 else if((player[2].getBlood()!=0)&&(player[2].getRole()!=user.getRole()-2)
247                 &&(player[2].getRole()!=user.getRole()+2)&&(!player[2].equals(chosen))){
248                     gamecardrunner.applyCard(primary,user,player[2],cardID,cardName,1);
249                     prepare.changeHashmap(cardID,2);
250                     i=i-1;
251                 }
252             }
253         }
```

233.若此張牌的名稱為” Fire!”，且 chosen 已死亡

234.~235.繼續判斷可否對下一位敵人出牌

236.~239.可以的話，對下一位敵人出牌後將牌放入棄牌堆，計數器 i-1

240.~252.重複執行上面兩個步驟

```
254         for(int i=0;i<handcard.size();i++){
255             enemyblood=chosen.getBlood();
256             blood=user.getBlood();
257             int cardID=handcard.get(i);
258             String cardName=card.getCardName(cardID);
259             if(blood<4 && cardName.equals("Gig Economy"))
260             {
261                 gamecardrunner.applyCard(primary,user,chosen,cardID,cardName,0);
262                 prepare.changeHashmap(cardID,2);
263                 i=i-1;
264             }
265             if(blood<4&&cardName.equals("Bar"))
266             {
267                 gamecardrunner.applyCard(primary,user,chosen,cardID,cardName,0);
268                 prepare.changeHashmap(cardID,2);
269                 i=i-1;
270             }
271         }
272     }
```

254.用 for 迴圈依續檢查手牌是否符合出牌條件

255.每次都重新取得 enemyblood

256.每次都重新取得 blood

257.宣告 cardID 為第 i 張牌

258.取得此張牌的卡牌名稱

259.~264.若此張牌的名稱為”Gig Economy”，且玩家血量小於 4，出牌後放入棄牌堆，計數器 i-1

265.~271. 若此張牌的名稱為”Bar”，且玩家血量小於 4，出牌後放入棄牌堆，計數器 i-1

auto_run method:

```
273     public void auto_run(int roundPlayer, Primary primary){
274 
275         Player[] player=new Player[3];
276         int round=roundPlayer;
277         for(int i=0;i<player.length;i++){
278             round+=1;
279             if(round>3)
280                 round=0;
281             player[i]=primary.getPlayer(round);
282         }
283         if((player[0].getBlood()!=0)&&(player[0].getRole()!=user.getRole()-2)&&(player[0].getRole()!=user.getRole()+2))
284             useCard(player[0],primary);
285         else if((player[1].getBlood()!=0)&&(player[1].getRole()!=user.getRole()-2)&&(player[1].getRole()!=user.getRole()+2))
286             useCard(player[1],primary);
287         else if((player[2].getBlood()!=0)&&(player[2].getRole()!=user.getRole()-2)&&(player[2].getRole()!=user.getRole()+2))
288             useCard(player[2],primary);
289         fold();
290     }
291 }
```

275.~282.宣告 Player[] player 依序存放每位玩家的 Player

283.~288.若 player 和 user 的陣營不同就對其出牌

289.呼叫 fold()來棄牌

參、 系統架構

我們上一次的出牌策略是一邊看牌一邊出，但我們發現這樣的勝率不太可觀。因為當你同時有可以攻擊的”Fire!”和搶奪其他人的牌” awpjopabemarrrr”，”Fire!”卻排在” awpjopabemarrrr”之前先出掉了，就表示有可能被對方的”Miss”阻擋而攻擊無效，但如果先使用” awpjopabemarrrr”就有機會能將對手的”Miss”搶為己用。於是我們就更改了迴圈的設計，從一個大的 for 迴圈改為許多小迴圈，就能在讀完牌後做出牌的排序。

出牌的順序我們遵循一個原則就是越多牌越好，所以先從能抽從牌庫中抽牌的” Tonight I Want Some”、 ” Please Support Counter”和” Different Pay For Equal Work”開始出牌。接者依序使用” beibigennosuke”和” awpjopabemarrrr”搶奪對手的手牌，來增加自己的攻擊成功的機率，也避免先使用後者導致對手沒手牌而無法出牌的情況。就算” Different Pay For Equal Work”讓對手收到更多”Miss”或補血的功能牌，我們也能用後面的出牌順序來彌補這種不利我方的情況發生。

經過前面兩個流程我們的手牌數量會到達顛峰，這時將”Fire!”使出，如果能讓對手一擊斃命是最佳的狀況，否則輪對手出牌補血後賽況可能又變回原點。攻擊完成後是我們的補血時間，減少需要將多餘手牌棄掉的情況，留在場上越久勝利的機會越大。

另外，我們這次也加上更多判斷式來幫助程式找到下一個可攻擊對象，這樣當 chosen 已死亡但手中還有能使用的牌時，能持續進攻直到手中的牌都使用完畢。改善了每輪固定同一攻擊對象時，可能面臨到的棄牌問題，也讓攻擊牌發揮最大的利用率。

我們的棄牌策略則是採取「未雨綢繆」，不丟棄”Miss”、補血和”Fire!”這三種卡牌，其他皆可丟的模式。我們用有牌就出的方法將可以出的牌都出掉，這時手牌可能會剩輔助手牌或尚無法使用的牌，那我們的系統就會避開上面三種手牌，根據要棄牌的總數丟棄排序較前的卡牌，達到最高的卡牌使用率。

如果現在滿血手中都剩”Miss”和補血這種不能馬上使用的牌，卻依然要繼續棄牌的話，我們會優先保留”Miss”，因為他們的功能相近但補血牌只有輪到自己出牌的回合才能出，我們將補血牌棄掉也能保證對手拿不到唯僅 8 張的補血牌。

肆、 使用說明

於指令欄輸入 javac *.java 後按下 enter 鍵將所有 .java 檔一併編譯成 .class 檔，接者輸入 java Group_Project3_Game Group_Project3_Strategy ComputerStrategy Group_Project3_Strategy ComputerStrategy，設定 0 及 2 號玩家為同一陣營，使用我們設計的棄牌與出牌邏輯，1 及 4 號玩家則使用電腦邏輯進行比賽。按下 enter 鍵後比賽過程與結果將列印於螢幕上。

伍、 實際執行結果

輸入 java Group_Project3_Game Group_Project3_Strategy ComputerStrategy Group_Project3_Strategy ComputerStrategy 得到的執行結果

```
a1093355@2021_OOPI_SERVER:~/Projrct3$ java Group39_Project3_Game Group39_Project3_Strategy ComputerStrategy
 Group39_Project3_Strategy ComputerStrategy
**********
now playing: player 0
blood: 4
Cards in my hand:
Fire!; Tonight I Want Some; Miss; Fire!; Tonight I Want Some;
Use fire! To Player 1
Use Tonight I Want Some!
Use fire! To Player 1
Use Tonight I Want Some!
Use fire! To Player 1
Use fire! To Player 1
Hancard before I fold : [38, 77, 7]
My blood : 4
Hancard after I fold : [38, 77, 7]
**********
now playing: player 1
blood: 0
**********
```

```
now playing: player 2
blood: 4
Cards in my hand:
Gig Economy; Fire!; Fire!; Fire!; Tonight I Want Some;
Use fire! To Player 3
Use fire! To Player 3
Use fire! To Player 3
Use Tonight I Want Some!
Use fire! To Player 3
Hancard before I fold : [57, 75]
My blood : 4
Hancard after I fold : [57, 75]
**********
0 2WIN
**********
```

陸、 其他(延伸討論)

我們在測試程式時,發現棄牌方法和比賽結果的列印順序會是隨機的，有各式各樣的排列組合(圖一)。查找了資料以後發現這和執行緒的執行順序有關，如果讀取對象為相同資源(如變數)，則執行緒順序不會導致不同結果；但如果是「寫入」相同資源的話，不控管執行緒順序的話就可能導致兩個或多個執行緒並行寫入同個變數，造成非預期的執行結果。這時可以採用同步處理，於宣告方法中加入 synchronized 關鍵字使其變成同步方法，就能鎖定每次只有一個 thread 在裡頭執行。而 synchronized 除了可以使用於方法外，也可以用在某個特定區塊使其變成同步區塊，則區塊內的程式會一次執行完畢，才開放其他執行緒執行，確保輸出的結果會與預期相同。

於編譯時跑出兩個 Note(圖二)顯示程式裡使用了不安全的指令可以在開頭加上-Xlint 重新編譯一遍,於是我們照做之後跑出問題在 Threads.java 中的第 165 行，因為 T 是個型別變數，但是和 Constructer 可接收的型態不同，也就是兩變數不相容，因此跑出警告(圖三)。 這也牽涉到 java 為靜態語言的特性，所有的型態在編譯時期(compile time)就檢查過了，而出現型別轉換問題卻沒有編譯錯誤只是給出警告的原因在於,編譯器無法知道誰會呼叫這方法，只要使用者輸入錯誤指令，就有機會跑出例外。所以解決方法就是將 T 強制轉型成 Constructer 想要的型態，或是建立對應不同例外情況的處理方式，確保程式的安全與穩固性。

另外我們一開始以為 strategy 建構子傳進來的 playerUserone 是可以使用在 auto run 中，編譯時卻會跑出 cannot find symbol 的 error(圖四)。因為它是 local variable，在尚未初始化的情況下沒辦法直接調用，所以我們創建了新的 Plyer[]來存取傳入的 player，再藉以判斷是否要對此 player 出牌。

```
**********
Hancard before I fold : [59, 60, 34, 31]
My blood : 4
Hancard after I fold : [59, 60, 34, 31]
0 2WIN
**********
```

（圖一）

```
Note: Threads.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
a1093367@2021_OOPI_SERVER:~/homework/Project3$
```

（圖二）

```
a1093355@2021_OOPI_SERVER:~/Projrct3$ javac -Xlint:unchecked *.java
Threads.java:165: warning: [unchecked] unchecked call to getConstructor(Class<?>...) as a member of the raw
type Class
          Constructor constructor =  clazz.getConstructor(params);
                                                          ^
  where T is a type-variable:
    T extends Object declared in class Class
1 warning
```

（圖三）

```
le()-2)&&(playerUserone.getRole()!=player.getRole()+2))
                                    ^
  symbol:   variable playerUserone
  location: class Strategy
./Strategy.java:199: error: cannot find symbol
        if((playerUserone.getBlood()!=0)&&(playerUserone.
le()-2)&&(playerUserone.getRole()!=player.getRole()+2))

  symbol:   variable player
  location: class Strategy
./Strategy.java:199: error: cannot find symbol
```

（圖四）