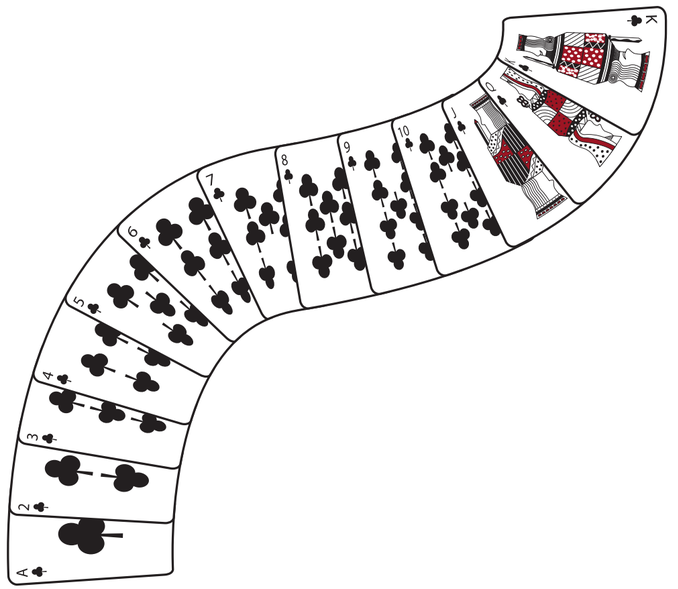
TCP/IP與網路程式設計
第四組
多人連線 21 點撲克牌遊戲

丁茂鋒 11303016A
蔡坤騰 11303021A
劉孟倣 11303025A
陳又嘉 11303059A
蘇柏誠 11303101A



介紹

- 這個程式是一個 TCP Server，讓多個 Client 透過網路連線，一起玩 21 點撲克牌遊戲，其中第一個連線的人是莊家，其餘是玩家,下面是使用到的 TCP/IP 技術重點

TCP Socket	**socket.socket() 建立伺服器**
多用戶連線	**threading.Thread()**
Client / Server 架構	Server 控制遊戲流程
同步處理	lock 避免資料競爭
網路通訊	sendall() / recv()
Timeout	settimeout() 限制玩家回應時間

程式碼

Server 全域設定與遊戲資料

```
5   HOST = "0.0.0.0"
6   PORT = 5555
7
8   players = []
9   dealer = None
10  lock = threading.Lock()
11  GAME_RUNNING = False
12
13  CARDS = ["A","2","3","4","5","6","7","8","9","10","J","Q","K"]
14  VALUES = {
15      "A":11,"2":2,"3":3,"4":4,"5":5,"6":6,"7":7,
16      "8":8,"9":9,"10":10,"J":10,"Q":10,"K":10
17  }
```

這些設定是 Server 啟動與 21 點遊戲運作的基礎，包括網路連線資訊、玩家管理，以及牌組與點數定義。

解說

變數/常數	作用	簡短解說
HOST	監聽 IP	0.0.0.0 表示接受所有來源連線
PORT	監聽 Port	伺服器開放的連線埠號
players	list	存所有玩家物件，包括莊家
dealer	Player	指向目前莊家
lock	threading.Lock	保護多執行緒操作玩家列表，避免衝突
GAME_RUNNING	bool	控制遊戲是否正在進行
CARDS	list	21 點牌組（不含花色）
VALUES	dict	每張牌對應的點數，A 預設 11，J/Q/K 10

程式碼

👉 一個 Player 物件 = 一個 TCP 連線的使用者

Server 用這個類別來管理每個玩家的狀態。

```python
class Player:
    def __init__(self, conn, name, dealer=False):
        self.conn = conn
        self.name = name
        self.is_dealer = dealer
        self.coins = 500 if dealer else 100
        self.hand = []
        self.bet = 0
        self.alive = True
        self.spectator = False
```

Server 透過 Player 類別集中管理每一個 TCP 連線玩家的遊戲狀態，方便進行多人同步與遊戲控制。

解說-玩家類別

屬性名稱	資料型態	說明
conn	socket	與 Client 的 TCP 連線，用來收送資料
name	str	玩家暱稱
is_dealer	bool	是否為莊家（第一個連線者）
coins	int	玩家金幣（莊家 500、玩家 100）
hand	list	玩家目前的手牌
bet	int	當回合下注金額
alive	bool	玩家是否仍在線（TCP 是否連線）
spectator	bool	是否為觀戰狀態（沒錢或斷線）

程式碼

```
37  def draw_card():
38      return random.choice(CARDS)
39
40  def calc(hand):
41      total = sum(VALUES[c] for c in hand)
42      aces = hand.count("A")
43      while total > 21 and aces:
44          total -= 10
45          aces -= 1
46      return total
47
48  def broadcast(msg):
49      offline = []
50      for idx, p in enumerate(players):
51          try:
52              p.conn.sendall((msg + "\n").encode())
53          except:
54              if p.alive:
55                  p.alive = False
56                  p.spectator = True
57                  offline.append((idx, p.name))
58
59      for idx, name in offline:
60          for p in players:
61              if p.alive:
62                  try:
63                      p.conn.sendall(
64                          f"玩家{idx}號 {name} 已離開房間\n".encode()
65                      )
66                  except:
67                      pass
```


解說-工具函式

函式名稱	相關程式行（重點）	簡短解說
draw_card()	random.choice(CARDS)	從牌組中隨機抽一張牌
calc(hand)	total = sum(...)	計算手牌總點數
calc(hand)	while total > 21 and aces:	A 超過 21 時由 11 改成 1
broadcast(msg)	p.conn.sendall(...)	Server 廣播訊息給所有玩家
broadcast(msg)	except:	偵測玩家斷線
broadcast(msg)	p.alive = False	標記玩家為離線

程式碼

```python
def recv_timeout(conn, prompt, timeout=10):
    try:
        conn.sendall((prompt + "\n").encode())
        conn.settimeout(timeout)
        data = conn.recv(1024)
        if not data:
            return None
        return data.decode().strip()
    except socket.timeout:
        return "TIMEOUT"
    except:
        return "DISCONNECT"
```

```python
def player_list():
    result = []
    for i, p in enumerate(players):
        role = "莊家" if p.is_dealer else f"玩家{i}"
        status = "離線" if not p.alive else ("觀戰" if p.spectator else "進行中")
        result.append(f"{role} {p.name} - 金幣:{p.coins} 狀態:{status}")
    return "\n".join(result)
```

解說-工具函式

函式名稱	相關程式行（重點）	簡短解說
recv_timeout(...)	conn.settimeout(timeout)	設定接收資料的時間限制
recv_timeout(...)	conn.recv(1024)	接收 Client 傳來的資料
recv_timeout(...)	except socket.timeout:	玩家超時未回應
player_list()	for i, p in enumerate(players):	走訪所有玩家
player_list()	role = ...	判斷莊家或玩家
player_list()	status = ...	顯示玩家狀態

程式碼

📌 功能：
準備新的一回合，
並檢查是否還有玩家可以繼續遊戲

```
94  def play_one_round():
95      global dealer
96
97      actives = [
98          p for p in players
99          if not p.is_dealer and p.alive and not p.spectator
100     ]
101
102     if not actives:
103         broadcast("玩家方已無可下注玩家，遊戲結束！")
104         return False
```

每一回合開始時，Server 會先檢查是否還有在線且能下注的玩家，若沒有，則直接結束遊戲。

解說-單回合遊戲

程式位置	程式碼重點	簡短解說
函式宣告	def play_one_round():	控制「單一回合」的遊戲流程
全域變數	global dealer	使用目前的莊家資料
玩家篩選	actives = [...]	找出還能下注的玩家
條件判斷	not p.is_dealer	排除莊家
條件判斷	p.alive	玩家必須在線
條件判斷	not p.spectator	排除觀戰玩家
結束檢查	if not actives:	沒有玩家可玩
廣播	broadcast(...)	通知所有 Client
回傳結果	return False	通知主程式遊戲結束

程式碼

📌 功能

Server 向每位玩家詢問下注金額，並處理斷線與例外狀況

```
107     for p in actives:
108         r = recv_timeout(p.conn, f"{p.name}，請下注（10秒，0=跳過）", 10)
109         if r == "DISCONNECT":
110             p.alive = False
111             p.spectator = True
112             broadcast(f"{p.name} 已離開房間")
113             continue
114         try:
115             p.bet = min(int(r), p.coins)
116         except:
117             p.bet = 0
118 
119     broadcast("=== 玩家列表 ===\n" + player_list())
```

在下注階段，Server 會限時向每位玩家收取下注金額，並即時處理斷線與錯誤輸入，確保遊戲能順利進行。

解說-下注

程式位置	程式碼重點	簡短解說
玩家迴圈	for p in actives:	依序處理每位可下注玩家
發送請求	recv_timeout(...)	限時 10 秒要求玩家下注
斷線判斷	r == "DISCONNECT"	玩家 TCP 斷線
狀態更新	p.alive = False	標記玩家離線
狀態更新	p.spectator = True	轉為觀戰
通知他人	broadcast(...)	告知其他玩家
金額轉換	int(r)	將輸入轉為數字
防超額	min(int(r), p.coins)	下注不能超過持有金幣
例外處理	except:	非法輸入自動設為 0
狀態顯示	broadcast(player_list())	顯示目前玩家狀態

程式碼

📌 功能

Server 給每位玩家與莊家發兩張起始牌，並把玩家的手牌傳給玩家

```
for p in actives + [dealer]:
    if not p.alive or p.spectator:
        continue
    p.hand = [draw_card(), draw_card()]
    if not p.is_dealer:
        p.conn.sendall(f"你的手牌: {p.hand}\n".encode())
```

Server 在每回合開始時，給每位玩家與莊家發兩張牌，玩家會看到自己的手牌，莊家手牌隱藏。

解說-發牌

程式位置	程式碼重點	簡短解說
玩家迴圈	for p in actives + [dealer]:	處理所有玩家加莊家
狀態檢查	if not p.alive or p.spectator:	斷線或觀戰玩家跳過
發牌	p.hand = [draw_card(), draw_card()]	每人發兩張牌
發給玩家	if not p.is_dealer:	只傳給玩家，不傳給莊家
發給玩家	p.conn.sendall(...)	將手牌傳送給玩家 Client

程式碼

📌 功能

Server 逐個詢問玩家是否要補牌，並檢查爆牌或五張牌限制

```
130     for p in actives:
131         if p.bet == 0 or not p.alive or p.spectator:
132             continue
133         while True:
134             r = recv_timeout(
135                 p.conn,
136                 f"{p.name} 你的牌 {p.hand} 要補牌嗎？(y/n)",
137                 10
138             )
139             if r == "DISCONNECT":
140                 p.alive = False
141                 p.spectator = True
142                 broadcast(f"{p.name} 已離開房間")
143                 break
144             if r != "y":
145                 break
```

Server 會逐一詢問玩家是否要補牌，並即時檢查爆牌或五張牌上限，保持遊戲公平與流程順暢。

解說-玩家補牌

程式位置	程式碼重點	簡短解說
玩家迴圈	for p in actives:	依序處理每位可下注玩家
跳過條件	if p.bet == 0 or not p.alive or p.spectator:	沒下注、離線或觀戰的玩家跳過
補牌迴圈	while True:	持續詢問玩家是否要補牌
限時接收	r = recv_timeout(...)	10 秒內沒回應或斷線
斷線處理	if r == "DISCONNECT":	玩家變觀戰，廣播離開訊息
停止補牌	if r != "y": break	玩家選擇不補牌

程式碼

📌 功能

Server 逐個詢問玩家是否要補牌，並檢查爆牌或五張牌限制

```
147             card = draw_card()
148             p.hand.append(card)
149             point = calc(p.hand)
150
151             if point > 21:
152                 broadcast(
153                     f"{p.name} 補到 {card} → 爆牌 {p.hand} ({point})"
154                 )
155                 break
156
157             if len(p.hand) >= 5:
158                 broadcast(
159                     f"{p.name} 五張牌 {p.hand} ({point})"
160                 )
161                 break
```

Server 會逐一詢問玩家是否要補牌，並即時檢查爆牌或五張牌上限，保持遊戲公平與流程順暢。

解說-玩家補牌

程式位置	程式碼重點	簡短解說
停止補牌	if r != "y": break	玩家選擇不補牌
抽牌	card = draw_card()	隨機抽一張牌
更新手牌	p.hand.append(card)	將新牌加入玩家手牌
計算點數	point = calc(p.hand)	判斷是否爆牌
爆牌判斷	if point > 21:	廣播玩家爆牌訊息，結束補牌
五張牌判斷	if len(p.hand) >= 5:	玩家手牌五張則自動停牌

程式碼

📌 功能

Server 自動讓莊家補牌，直到手牌點數 ≥ 17，然後廣播莊家手牌

```
while calc(dealer.hand) < 17:
    dealer.hand.append(draw_card())

broadcast(
    f"莊家手牌: {dealer.hand} ({calc(dealer.hand)})"
)
```

莊家按照規則自動補牌，直到點數至少 17，然後將結果廣播給所有玩家，保持遊戲透明。

解說-莊家補牌

程式位置	程式碼重點	簡短解說
補牌迴圈	while calc(dealer.hand) < 17:	莊家手牌點數小於 17 時繼續補牌
抽牌	dealer.hand.append(draw_card())	莊家抽牌加入手牌
廣播手牌	broadcast(f"莊家手牌: {dealer.hand} ({calc(dealer.hand)})")	通知所有玩家莊家的最終手牌與點數

程式碼

📌 功能

Server 比較每位玩家與莊家的點數，結算金幣並判斷遊戲是否結束

```
    d_point = calc(dealer.hand)
    for p in actives:
        if p.bet == 0 or not p.alive or p.spectator:
            continue

        p_point = calc(p.hand)

        if p_point > 21:
            p.coins -= p.bet
            dealer.coins += p.bet
            broadcast(
                f"{p.name} 爆牌 {p.hand} ({p_point})，輸了 {p.bet}"
            )

        elif d_point > 21:
            p.coins += p.bet
            dealer.coins -= p.bet
            broadcast(
                f"{p.name} 贏了 {p.bet}（莊家爆牌）"
            )

        elif p_point > d_point:
            p.coins += p.bet
            dealer.coins -= p.bet
            broadcast(f"{p.name} 贏了 {p.bet}")
```


解說-結算

程式位置	程式碼重點	簡短解說
莊家點數	d_point = calc(dealer.hand)	計算莊家最終點數
玩家迴圈	for p in actives:	逐一結算每位可下注玩家
跳過條件	if p.bet == 0 or not p.alive or p.spectator:	沒下注、離線或觀戰的玩家跳過
玩家點數	p_point = calc(p.hand)	計算玩家手牌點數
爆牌判斷	if p_point > 21:	玩家爆牌 → 扣掉下注金幣，莊家加錢
莊家爆牌	elif d_point > 21:	莊家爆牌 → 玩家贏錢，莊家扣錢
玩家勝利	elif p_point > d_point:	玩家點數大於莊家 → 玩家贏錢

程式碼

📌 功能

Server 比較每位玩家與莊家的點數，結算金幣並判斷遊戲是否結束

```
        elif p_point < d_point:
            p.coins -= p.bet
            dealer.coins += p.bet
            broadcast(f"{p.name} 輸了 {p.bet}")

        else:
            broadcast(
                f"{p.name} 平手 (Push) {p.hand} ({p_point})"
            )

        if p.coins <= 0:
            p.spectator = True

    broadcast("=== 回合結束 ===\n" + player_list())

    if dealer.coins <= 0:
        broadcast("玩家方獲勝！遊戲結束")
        return False

    if all(p.spectator for p in players if not p.is_dealer):
        broadcast("莊家獲勝！遊戲結束")
        return False

    return True
```


解說-結算

程式位置	程式碼重點	簡短解說
玩家失敗	elif p_point < d_point:	玩家點數小於莊家 → 玩家輸錢
平手	else:	點數相同 → 平手 (Push)
金幣檢查	if p.coins <= 0:	玩家破產 → 變成觀戰
廣播回合結束	broadcast("=== 回合結束 ===\n" + player_list())	告知所有玩家回合結束與最新狀態
遊戲結束判斷1	if dealer.coins <= 0:	莊家沒錢 → 玩家方勝利
遊戲結束判斷2	if all(p.spectator for p in players if not p.is_dealer):	所有玩家破產 → 莊家勝利
回傳結果	return True / False	控制 GAME_RUNNING 是否繼續

程式碼

📌 功能

Server 為每個連線的 Client 建立 Player，管理玩家身分與遊戲流程

```
227 def handle(conn):
228     global dealer, GAME_RUNNING
229     try:
230         conn.sendall("請輸入暱稱:\n".encode())
231         name = conn.recv(1024).decode().strip()
232 
233         with lock:
234             is_dealer = dealer is None
235             p = Player(conn, name, is_dealer)
236             players.append(p)
237             if is_dealer:
238                 dealer = p
239 
240         broadcast(f"{name} 進入房間\n" + player_list())
241 
242         if p.is_dealer:
243             conn.sendall("你是莊家，按任意鍵開始遊戲\n".encode())
244             conn.recv(1024)
245             broadcast("遊戲開始！")
246             GAME_RUNNING = True
247 
248             while GAME_RUNNING:
249                 if not play_one_round():
250                     GAME_RUNNING = False
251                     break
252     except:
253         pass
```

解說-Client 連線處理

程式位置	程式碼重點	簡短解說
函式宣告	def handle(conn):	每個 Client 連線都會呼叫這個函式
全域變數	global dealer, GAME_RUNNING	使用莊家資料與遊戲狀態
輸入暱稱	conn.sendall("請輸入暱稱:\n")	要求玩家輸入名字
接收暱稱	name = conn.recv(1024).decode().strip()	收到 Client 回傳的暱稱
建立玩家	p = Player(conn, name, is_dealer)	產生 Player 物件
加入玩家列表	players.append(p)	將 Player 存進全域玩家列表
判斷莊家	is_dealer = dealer is None	第一個進來的人成為莊家
廣播玩家加入	broadcast(f"{name} 進入房間\n" + player_list())	通知所有玩家
莊家啟動遊戲	if p.is_dealer:	莊家按任意鍵開始遊戲
遊戲主迴圈	while GAME_RUNNING:	持續執行 play_one_round()
遊戲結束	if not play_one_round(): GAME_RUNNING = False	回傳 False → 停止遊戲

程式碼

📌 功能

啟動 TCP Server，持續監聽玩家連線，並為每個 Client 開啟新的 Thread

```
def main():
    s = socket.socket()
    s.bind((HOST, PORT))
    s.listen()
    print("Server started")

    while True:
        conn, addr = s.accept()
        threading.Thread(
            target=handle,
            args=(conn,),
            daemon=True
        ).start()


if __name__ == "__main__":
    main()
```

解說

程式位置	程式碼重點	簡短解說
函式宣告	def main():	Server 主程式入口
建立 Socket	s = socket.socket()	建立 TCP Socket
綁定 IP 與 Port	s.bind((HOST, PORT))	Server 監聽指定 IP 與 Port
開始監聽	s.listen()	允許 Client 連線
提示訊息	print("Server started")	Console 顯示 Server 已啟動
接收連線	conn, addr = s.accept()	阻塞等待玩家連線
多執行緒	threading.Thread(target=handle, args=(conn,), daemon=True).start()	為每個 Client 建立 Thread，非阻塞 Server

程式碼

Client 主要負責玩家輸入、接收 Server 廣播訊息，以及將指令傳回 Server。所有遊戲邏輯都在 Server 端，Client 只是顯示介面與玩家操作。

```python
SERVER_IP = "127.0.0.1"
PORT = 5555

sock = socket.socket()
my_name = ""

def clear_screen():
    os.system('cls' if os.name == 'nt' else 'clear')

def receive():
    while True:
        try:
            data = sock.recv(1024).decode()
            if not data:
                print("[系統] 伺服器已斷線")
                break
            print(data.strip())
        except:
            print("[系統] 連線中斷")
            break
```

Client 只負責顯示和玩家輸入，所有遊戲邏輯都在 Server，透過 TCP 連線與多執行緒保持即時互動。

解說-Client

程式位置	程式碼重點	簡短解說
建立 Socket	sock = socket.socket()	建立 TCP 連線物件
Server 連線	sock.connect((SERVER_IP, PORT))	連到 Server 指定 IP & Port
清除畫面	clear_screen()	開始前清空終端顯示
玩家暱稱	my_name = input()	玩家輸入暱稱傳給 Server
接收訊息	def receive():	建立 Thread 不斷接收 Server 廣播

程式碼

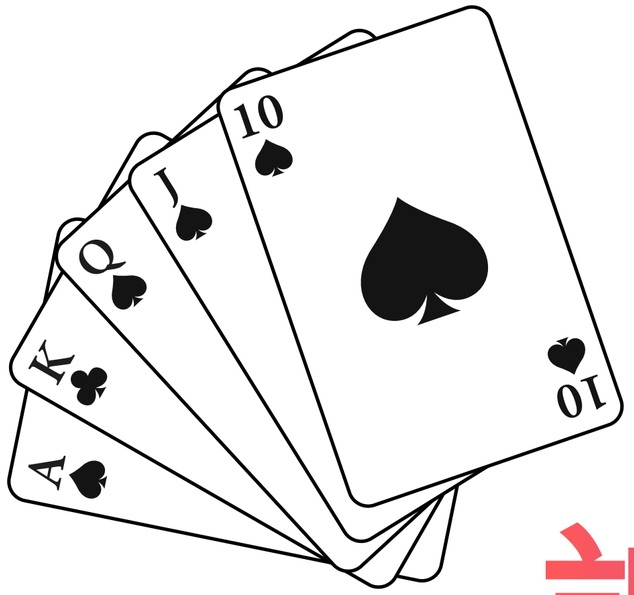
Client 主要負責玩家輸入、接收 Server 廣播訊息，以及將指令傳回 Server。所有遊戲邏輯都在 Server 端，Client 只是顯示介面與玩家操作。

```
26  def start_client():
27      sock.connect((SERVER_IP, PORT))
28      threading.Thread(target=receive, daemon=True).start()
29
30      my_name = input().strip()
31      sock.sendall((my_name + "\n").encode())
32
33      while True:
34          try:
35              msg = input()
36              sock.sendall((msg + "\n").encode())
37          except:
38              break
39
40  if __name__ == "__main__":
41      clear_screen()
42      print("=== 21 點遊戲 Client ===")
43      start_client()
```

Client 只負責顯示和玩家輸入，所有遊戲邏輯都在 Server，透過 TCP 連線與多執行緒保持即時互動。

解說-Client

程式位置	程式碼重點	簡短解說
接收處理	sock.recv(1024)	收到訊息後印在螢幕上
發送訊息	msg = input(); sock.sendall(...)	玩家輸入訊息或操作傳回 Server
多執行緒	threading.Thread(target=receive, daemon=True).start()	接收訊息與輸入同時進行，不會互相阻塞



謝謝大家