Socket Assignment Part 2

Simple? Card Game

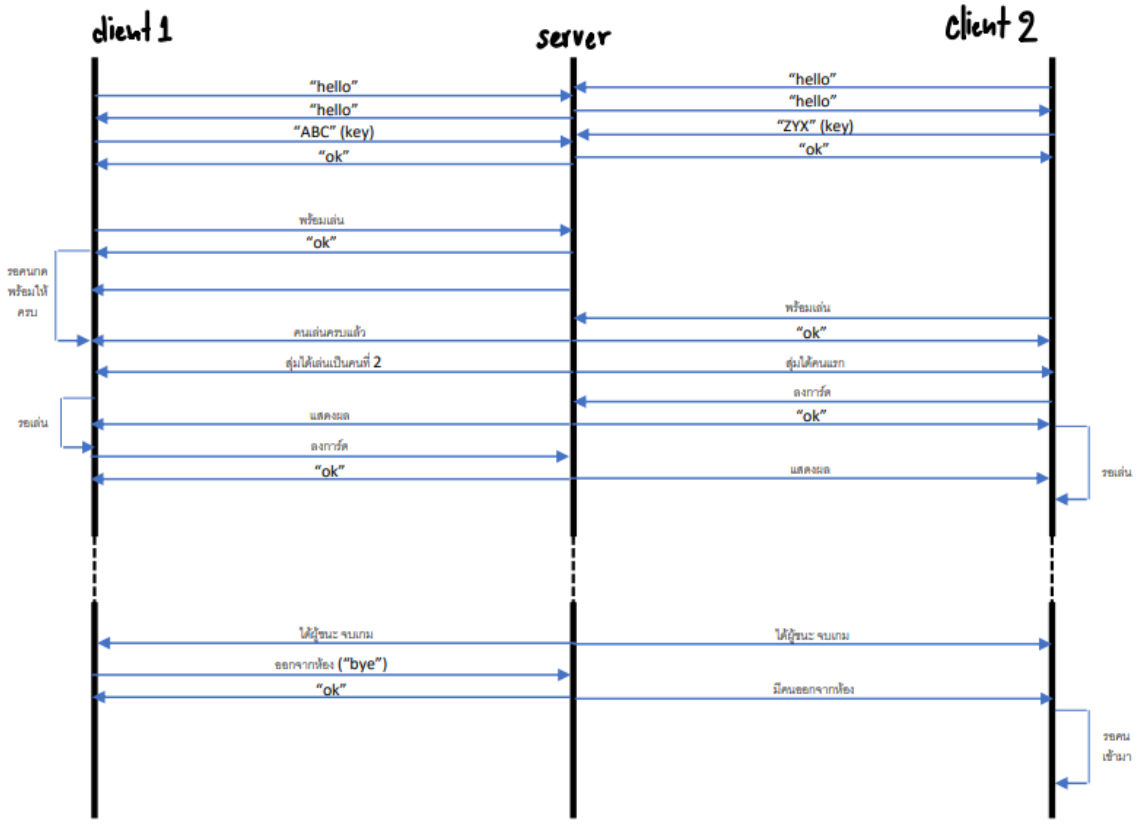
วิธีเล่น

เมื่อเข้า client ของเกมจะมีให้ใส่ชื่อ และกดว่าพร้อมเล่นหรือยัง แล้วเมื่อผู้เล่นทุกคนพร้อม เกมก็จะแจกไพ่ให้ผู้เล่นเท่าๆกัน พร้อมกับไพ่ joker คนละใบ โดยการเล่นจะผลัดการเล่นวนไพ่เรื่อยๆ โดยการลงไพ่จะลงได้ใบเดียว และต้องมีศักดิ์ที่สูงกว่าที่ลงอยู่บนโต๊ะ หรือปล่อยผ่านตานั้นไป และจะบังคับปล่อยผ่านถ้าไม่มีไพ่ศักดิ์สูงกว่าบนมือ โดยลำดับไพ่จะดูจากเลขเรียงตามลำดับต่อไปนี้ A < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < 10 < J < Q < K แต่ถ้าเลขเท่ากันให้ดูที่ตราของไพ่โดยเรียงตามลำดับคือ ดอกจิก < หลามตัด < โพธิ์แดง < โพธิ์ดำ และสำหรับไพ่ joker จะสามารถลงทับไพ่อะไรก็ได้ และสามารถถูกไพ่อะไรก็ได้ลงทับ โดยในตาแรก ผู้เล่นจะลงไพ่อะไรก็ได้ หรือถ้ามีคนข้ามตาครบทุกคน คนที่เล่นล่าสุดจะสามารถลงไพ่อะไรก็ได้ และคนที่ไพ่บนมือหมดก่อนจะเป็นผู้ชนะ

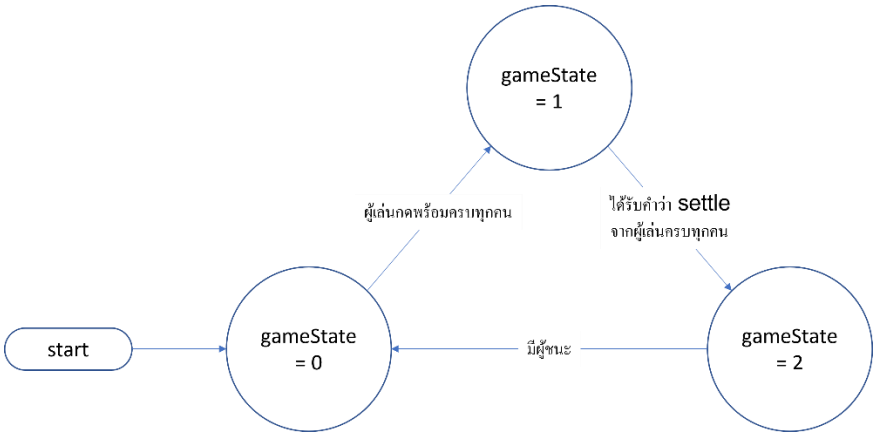
สมาชิกในกลุ่ม

1. นายสุทธิรัก มัธยวีรเกียรติ 6210110383

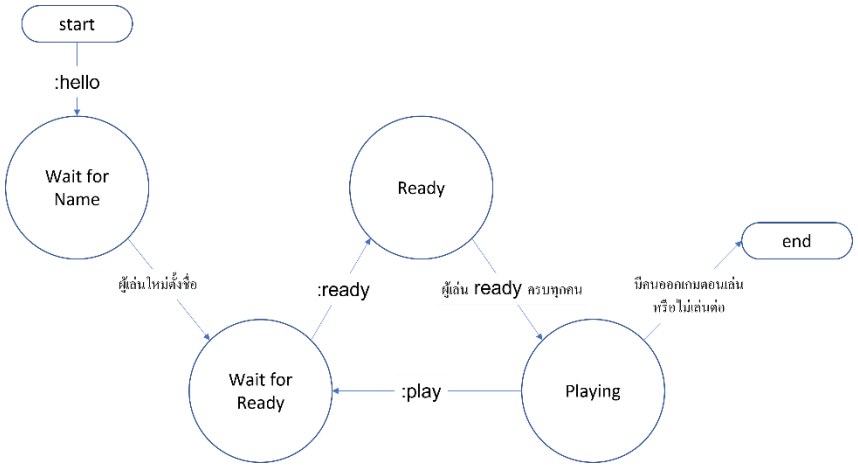
Sequence Diagram



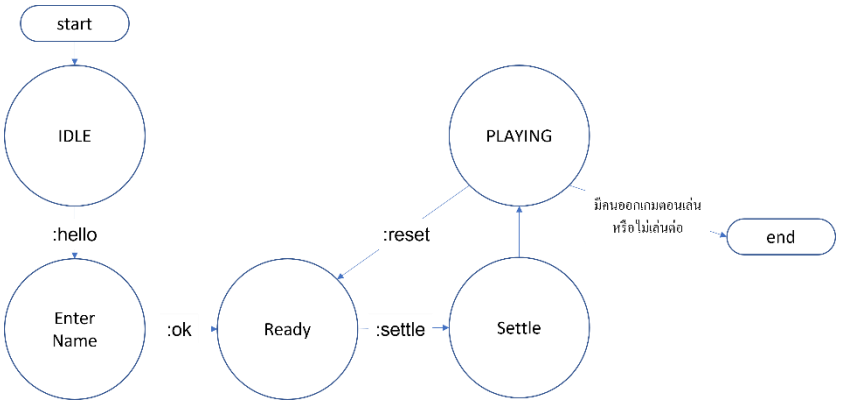
State Diagram (gameState – server.js)



State Diagram (player – server.js)



State Diagram (client.js)



server.js

```
const net = require('net')

var players = []
var gameState = 0
var playerResCount = 0
var tableTop = null
var passTurn = 0

net.createServer((socket) => {

}).listen(8080, '127.0.0.1')
console.log('Server listening on 127.0.0.1:8080')
```

สร้าง server ขึ้นมากำหนดตัวแปรที่ใช้ใน server เป็น players สำหรับเก็บรายละเอียดผู้เล่นรวมถึง socket ที่ผู้เล่นเชื่อมเข้ามา

gameState ไว้ใช้นับ state ของเกม

playerResCount ไว้ใช้นับจำนวนการตอบกลับคำสั่งของผู้เล่น

tableTop ไว้เก็บไพ่ล่าสุดที่ผู้เล่นลงไว้

passTurn ไว้นับจำนวนผู้เล่นข้ามตาติดต่อกัน

```
var msg = data.toString()
var player = findPlayer(socket.remoteAddress, socket.remotePort)

if(msg == 'play')
{
    console.log(`${player.name}(${player.addr}:${player.port}) chose to play again.`)
    player['state'] = 1
    socket.write('ok')
    console.log('!!!')
    return;
}
```

ที่ state ทั่วไปให้แปลง data ที่ได้รับเป็นข้อความเก็บใน msg แล้วหาผู้เล่นจาก ip ที่เชื่อมเข้ามาว่ามีในระบบหรือไม่แล้วเก็บใน player จากนั้นตรวจว่า msg เป็นคำว่า play หรือใช่ให้ย้อน state ของ Player(ใน server) เป็น Wait for Ready

```
if(gameState == 0)
{
    if(msg == 'hello' && player == null)
    {
        players.push({
            addr: socket.remoteAddress,
            port: socket.remotePort,
            name: `${socket.remoteAddress}:${socket.remotePort}`,
            state: 0,
            client: socket
        })
        socket.write('hello')
        socket.read()
        console.log(`${socket.remoteAddress}:${socket.remotePort} joined.`)
        return
    }
```

ที่ gameState = 0 ถ้า msg = hello และไม่เจอผู้เล่นให้เพิ่มข้อมูลผู้เล่นใน players แล้วให้ state ของผู้เล่นเป็น Wait for Name แล้วตอบกลับ

```
if(player.state == 0)   // wait for key
{
    player['name'] = msg
    player['state'] = 1
    socket.write('ok')
    console.log(`${player.addr}:${player.port} has set nickname to ${msg}.`)
    return
}
```

จากข้างบน ถ้าผู้เล่นตอบกลับมาใน state เป็น Wait For Name ให้นำข้อความที่ตอบกลับมามาเป็นชื่อผู้เล่น แล้วเปลี่ยน state เป็น Wait for Ready แล้วค่อยตอบกลับไป

```
if(player.state == 1 && msg == 'ready')     // ready
{
    player['state'] = 2
    console.log(`${player.name}(${player.addr}:${player.port}) is ready.`)
    if(getReadyPlayer() == players.length && players.length >= 2 && players.length <= 8)
    {
        console.log('All players are ready.')
        gameState = 1
        player['state'] = 3
        announce('start')
    }
    else if(players.length > 8)
    {
        while(players.length > 8)
        {
            var kickPlayer = players.splice(8, 1)
            kickPlayer.client.destroy()
        }
    }
    return
}
```

จากข้างบน ถ้าผู้เล่นตอบกลับมาใน state เป็น Wait For Ready เป็นข้อความว่า ready ให้เปลี่ยน state เป็น Ready แล้วนับผู้เล่นที่ Ready ทั้งหมดถ้าเกิดผู้เล่นทุกคนเป็น Ready และมีจำนวนตั้งแต่ 2-8 คน แล้วให้เริ่มเกม(gameState = 1)แล้วเปลี่ยน state เป็น Playing จากนั้นประกาศ start ให้ผู้เล่นทุกคน แต่ถ้ามีผู้เล่นเกินให้ตัดการเชื่อมต่อผู้เล่นที่เข้าสายที่สุดจนกว่าจะเหลือ 8 คน

```
else if(gameState == 1)
{
    if(msg == 'settle')
    {
        getResponse(function(){
            gameState = 2
            socket.emit('playGameSettle')
        })
        return;
    }
}
```

ที่ gameState = 1 ถ้า server ได้รับข้อความ settle ครบจากผู้เล่นทุกคน ให้เปลี่ยน gameState เป็น gameState = 2 แล้วทำ event ชื่อ playGameSettle

```
socket.on('playGameSettle', function(){
    shufflePlayerOrder()
    var cards = shuffleCardForPlayers()
    for(var i in players)
    {
        players[i].client.write(JSON.stringify({
            cards: cards[i],
            role: parseInt(i)+1,
            joker: 1
        }))
    }
})
```

เมื่อ event playGameSettle เริ่มให้สลับลำดับผู้เล่น แล้วสับและแจกไพ่ แล้วส่งข้อมูลอันดับการเล่น และไพ่ในมือให้แก่ผู้เล่น

```
else if(gameState == 2)
{
    var name = `${player.name}(${player.addr}:${player.port})`

    if(msg == '') return;

    if(msg == 'win')
    {
        console.log('We have the winner.')
        announce(JSON.stringify({
            cmd: 'announce', msg: `${name} won the game.\n\n\n\n\n\n\n`
        }));
        setTimeout(function(){
            announce(JSON.stringify({
                cmd: 'reset'
            }))
            gameState = 0
            playerResCount = 0
            tableTop = null
            passTurn = 0
        }, 3000)
        return;
    }
```

ที่ gameState = 2 ถ้าได้รับข้อความเปล่าก็ไม่ต้องทำอะไรต่อ แต่ถ้าได้รับข้อความ win จากผู้เล่นคนไหนให้ประกาศว่าผู้เล่นคนนั้นชนะ หลังจากนั้น 3 วินาทีให้ถามผู้เล่นแต่ละคนว่าจะเล่นต่อหรือไม่ แล้ว reset ตัว gameState กลับไปที่ 0

```
if(msg == 'gotten')
{
    getResponse(function(){
        players[0].client.write(JSON.stringify({
            cmd: 'turn',
            top: tableTop
        }))
        var tmp = players.splice(0, 1)[0]
        players.push(tmp)
    })
    return
}
```

จากข้างบน ถ้าได้ข้อความว่า gotten จากผู้เล่นครบทุกคน ตัวเกมจะให้เทิร์นการเล่นกับผู้เล่นคนแรก แล้วนำลำดับของผู้เล่นคนนั้นไปต่อหลัง players

```
if(msg == 'pass')
{
    console.log(`${name} has passed.`)
    announce(JSON.stringify({
        cmd: 'announce', msg: `${name} has passed.`
    }));
    if(++passTurn >= players.length)
    {
        console.log(`All players have passed their turn.\nClear the Table`);
        tableTop = null
        setTimeout(function(){
            players[players.length-1].client.write(JSON.stringify({
                cmd: 'turn',
                top: tableTop
            }))
            passTurn = 0
        }, 500)
    }
    else
    {
        setTimeout(function(){
            players[0].client.write(JSON.stringify({
                cmd: 'turn',
                top: tableTop
            }))
            var tmp = players.splice(0, 1)[0]
            players.push(tmp)
        }, 500)
    }
    return
}
```

จากข้างบน ถ้าได้ข้อความว่า pass จากผู้เล่น ระบบจะประกาศว่าผู้เล่นคนนั้นข้ามตา แล้วนับจำนวนผู้เล่นที่ข้ามตาต่อจากคนก่อนหน้า ถ้ามีค่ามากกว่าหรือเท่ากับจำนวนผู้เล่นให้เคลียร์โต๊ะ แล้วหลังจากนั้นครึ่งวินาทีก็ให้เทิร์นกับคนที่ข้ามตาคนสุดท้ายเล่นต่อ แต่ถ้ายังไม่เกินก็รอครึ่งวินาที แล้วค่อยให้เทิร์นกับคนถัดไป

```
passTurn = 0
console.log(`${name} has placed ${msg}.`)
announce(JSON.stringify({
    cmd: 'announce', msg: `${name} has placed ${getCardFullName(msg)}.`
}));
tableTop = msg
setTimeout(function(){
    players[0].client.write(JSON.stringify({
        cmd: 'turn',
        top: tableTop
    }))
    var tmp = players.splice(0, 1)[0]
    players.push(tmp)
}, 500)
```

แต่ถ้าไม่ได้รับตามที่ระบุข้างบน(ลงไพ่ได้ตามปกติ) ให้ reset ตัวนับจำนวนคนข้ามตา แล้วระบบจะประกาศว่าผู้เล่นคนล่าสุดลงไพ่อะไร เป็นชื่อเต็มพร้อมสัญลักษณ์ เช่น Ace of Spade (A ♠) แล้วหลังจากนั้น 5 วินาทีก็ให้เทิร์นแก่คนถัดไปได้เล่นต่อ

```
socket.on('close', function(){
    console.log(`${socket.remoteAddress}:${socket.remotePort} disconnected.`)
    var i = 0
    while(i < players.length)
    {
        if(players[i].client == socket || players[i].client.destroyed)
        {
            players.splice(i--, 1)
        }
        i++
    }
    if(gameState >= 1)
    {
        gameState = 0
        playerResCount = 0
        tableTop = null
        passTurn = 0
        announce(JSON.stringify({
            cmd: 'announce', msg: `disconnected`
        }));
        while(i < players.length)
        {
            player[i++].state = 1
        }
    }
})
```

ตรงนี้ถ้ามีผู้เล่นปิดตัว client ให้ลบข้อมูลออกจาก players แต่ถ้ากำลังเล่นอยู่(gameState >= 1) ให้ตัวเกมย้อนกลับไป gameState = 0 แล้วประกาศว่ามีคน disconnect พร้อมกับ set state ให้ผู้เล่นทุกคนที่ยังอยู่เป็น Wait for Ready

```
net.createServer((socket) => {

    if(gameState >= 1)
    {
        socket.destroy()
        return
    }
```

ตรงนี้ คือถ้ามีคนเชื่อมต่อตอนที่เล่นอยู่(gameState >= 1) ให้ตัดการเชื่อมต่อทันที

client.js

```
const net = require('net')
const readline = require('readline').createInterface({
    input: process.stdin,
    output: process.stdout
})

var state = 0
var cards = null
var joker = 0
var name = ''

client = new net.Socket()
client.connect(8080, '127.0.0.1', function(){
    client.write('hello')
})
```

ตั้งค่า module และตัวแปร โดยให้ state นับ state, cards เป็น list ของไพ่บนมือ, joker เป็นจำนวนไพ่ joker บนมือ, name เก็บชื่อผู้เล่น และตัว client ให้เชื่อมต่อกับ server แล้วส่งข้อความ hello ไปยัง server

```
client.on('data', (data) => {
    var msg = data.toString()

    if(state == 0 && msg == 'hello')    // idle
    {
        state = 1
        client.emit('inputKey')
        return;
    }

    if(state == 1 && msg == 'ok')       // input key
    {
        state = 2
        client.emit('playerReady')
        return
    }

    if(state == 2 && msg == 'start')    // all are ready
    {
        client.write('settle')
        state = 3
        return
    }
```

ถ้าผู้เล่นได้รับข้อความจาก server ให้แปลงข้อความ data แล้วเก็บใน msg

- ถ้าผู้เล่นอยู่ใน state: IDLE แล้วได้รับข้อความ hello ก็ให้ไปทำ event: inputKey แล้วเลื่อน state เป็น Enter Name
- ถ้าผู้เล่นอยู่ใน state: Enter Name แล้วได้รับข้อความ ok ก็ให้ไปทำ event: playerReady แล้วเลื่อน state เป็น Ready
- ถ้าผู้เล่นอยู่ใน state: Ready แล้วได้รับข้อความ start ก็ให้เลื่อน state เป็น Settle แล้วส่งข้อความ settle ไปให้ server

```
client.on('inputKey', function(){
    readline.question('Enter name: ', (ans) => {
        client.write(ans)
        name = ans
    })
})
```

ที่ event: inputKey จะมีการถามชื่อ แล้วส่งชื่อไปให้ server

```
client.on('playerReady', function(){
    readline.question(`${name}, are you ready? (y/N): `, (ans) => {
        if(ans.toLowerCase() == 'y' || ans.toLowerCase() == 'yes')
        {
            client.write('ready')
            return
        }
        client.emit('playerReady')
    })
})
```

ที่ event: playerReady จะมีการถามว่าผู้เล่นพร้อมหรือยัง ถ้าพร้อมแล้วให้ส่งข้อความ ready ให้ server ถ้ายังไม่พร้อมให้เริ่ม event นี้ใหม่จนกว่าจะพร้อม

```
if(state == 3)                      // settle
{
    var obj = JSON.parse(msg)
    if(obj['cmd'] == 'announce')
    {
        if(obj['msg'] == 'disconnected')
        {
            console.log('Game ends due to some player disconnected.')
            client.destroy()
        }
        else console.log(obj['msg'])
        client.write('')
        return
    }
    console.log('Your card...')
    cards = obj['cards'].sort(compareFunction)
    joker = obj['joker']
    printCardName()
    console.log(`You got Turn #${obj['role']}`)
    client.write('gotten')
    state = 4
    return
}
```

ถ้าผู้เล่นอยู่ state: SETTLE เมื่อได้รับข้อความให้แปลงข้อความเป็น list ของ ไพ่บนมือ, จำนวนไพ่ joker และอันดับการเล่น แล้วแสดงผลไพ่บนมือทั้งหมดแล้วส่งข้อความ gotten กลับไปที่ server แล้วเลื่อน state เป็น PLAYING แต่ถ้ามีประกาศให้แสดงข้อความประกาศนั้น ยกเว้นว่ามีประกาศ disconnected ก็ให้จบการทำงาน

```
if(state = 4)
{
    var obj = JSON.parse(msg)
    if(obj['cmd'] == 'turn')
    {
        client.emit('playerTurn', obj['top'], '')
    }
    else if(obj['cmd'] == 'announce')
    {
        if(obj['msg'] == 'disconnected')
        {
            console.log('Game ends due to some player disconnected.')
            client.destroy()
        }
        else console.log(obj['msg'])
        client.write('')
    }
    else if(obj['cmd'] == 'reset')
    {
        state = 1
        cards = null
        joker = 0
        client.emit('playerReplay')
    }
    return;
}
```

ถ้าผู้เล่นอยู่ state: PLAYING ถ้าได้รับข้อความ turn ให้ทำ event: playerTurn แต่ถ้าเป็นข้อความ reset ให้เลื่อน state กลับไปที่ state: Enter Name แล้ว reset ไพ่บนมือ แล้วทำ event: playerReplay แต่ถ้ามีประกาศให้แสดงข้อความประกาศนั้น ยกเว้นว่ามีประกาศ disconnected ก็ให้จบการทำงาน

```
client.on('playerTurn', function(top, err){
    console.log('----------------------------------------')
    console.log('Your Turn')
    printCardName()
    printTableTop(top)
    if(hasValidCard(top))
    {
        if(err != '') console.log(err)
        readline.question(`Pick your card (or press 9 to pass your turn)\n> `, choice => {

            if(state != 4) return;

            if(choice == '9')
            {
                client.write('pass')
                return;
```

playerTurn เป็น event ควบคุมการเล่นให้เป็นไปตามกติกา หรือวิธีเล่นข้างบนโดยจะมีการแสดงการ์ดและลำดับไพ่บนมือ

```
client.on('playerReplay', function(){
    readline.question(`${name}, do you want to play again? (Y/n): `, (ans) => {
        if(ans.toLowerCase() != 'n' && ans.toLowerCase() != 'no')
        {
            client.write('play')
            return
        }
        client.destroy()
    })
})
```

playerReplay เป็น event ถามผู้เล่นว่าจะเล่นต่อหรือไม่ ถ้าเล่นต่อให้ส่งข้อความ play ไปที่ server แต่ถ้าไม่เล่นต่อก็จบการทำงาน