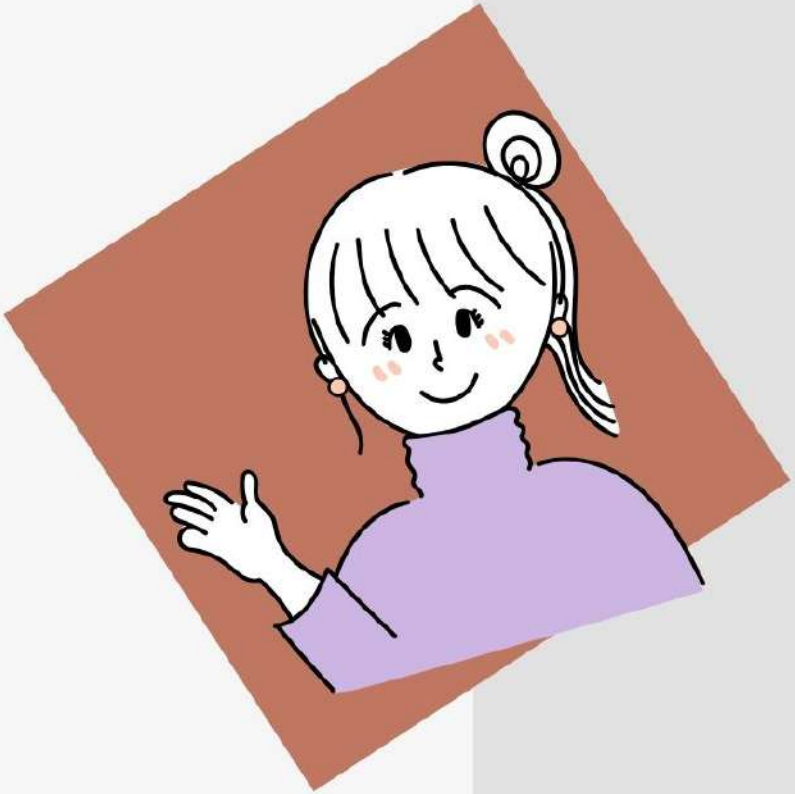
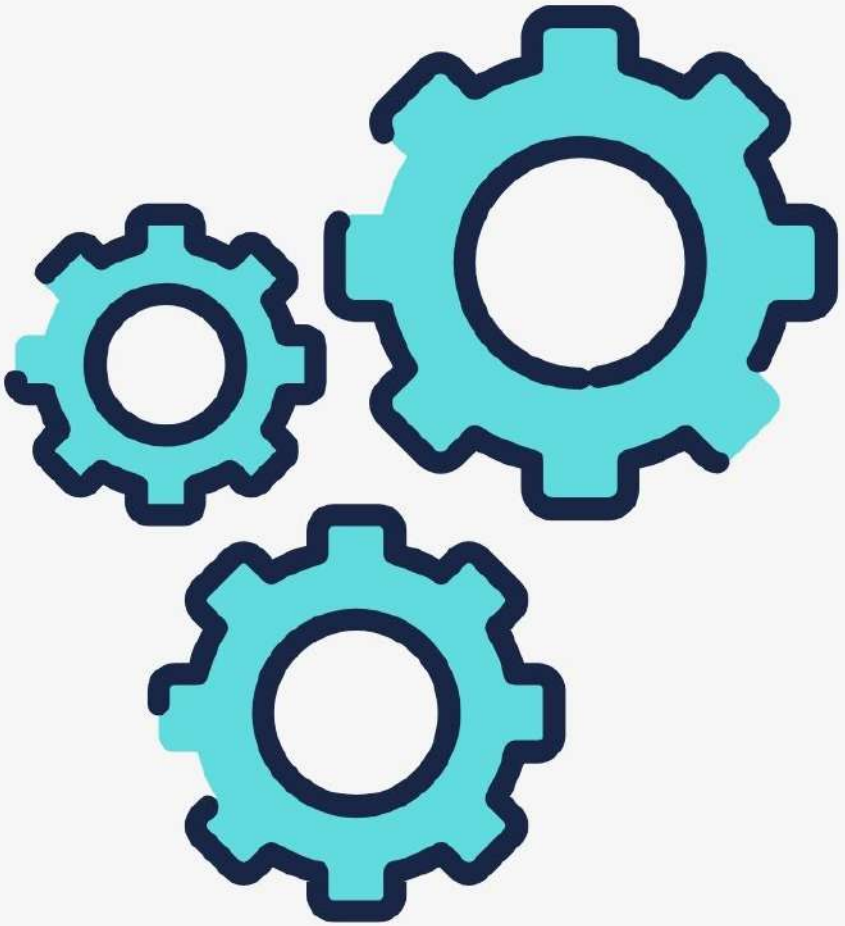
教室伺服器互動系統

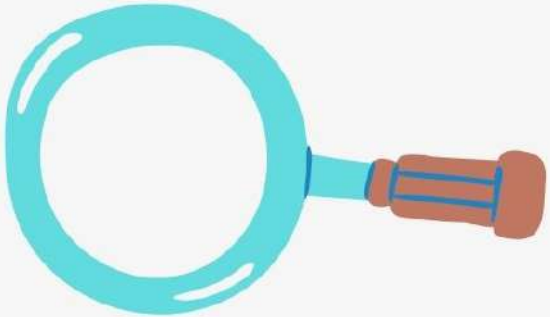
CONTENTS



簡介

主要是模擬教室學員與老師的互動
學生端發指令，例如喝水、睡覺、滑手機等。
老師端發指令，例如廣播、清理教室、留言給某位同學等。
教室伺服器收到命令後檢核是否可以，在返回對應訊息。





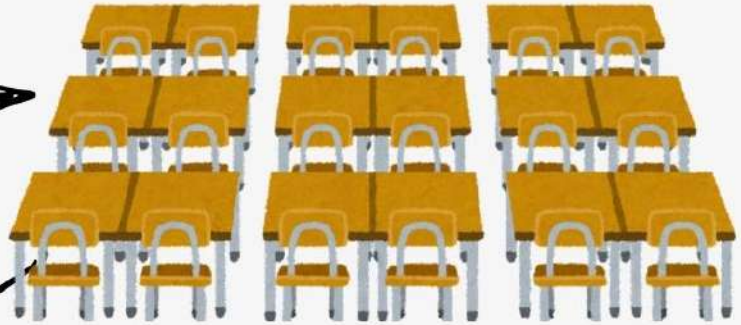
專案目的

仿造遊戲伺服器，簡化為較生活化的案例並用JAVA學到的技術實現伺服器、玩家、與後臺工具互動的概念。

系統架構

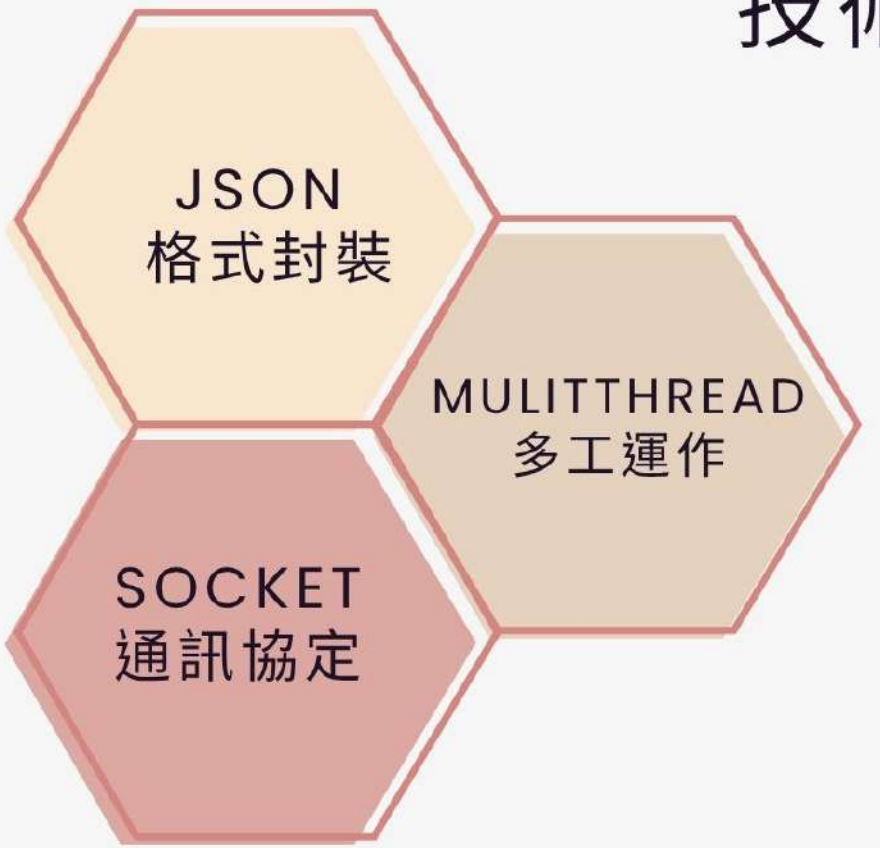
教室伺服器
(SERVER)

老師端
(TOOLS)



學生
(CLIENT)

技術重點



分工-CLIENT

透過Studentgenerate來產出學生 並隨機決定簽到跟後續行為
再從StudentClient向ClassroomServer發送學生的狀態

```java
public class Studentgenerate {

    // Student 內部類別保持不變
    static class Student {
        private String id;
        private String name;
        private int attendanceStatus; // 0 代表未到, 1 代表簽到
        private int activityChoice; // 儲存簽到學生隨機選擇的活動狀態

        public Student(String id) {
            this.id = id;
            this.name = "學員";

            Random random = new Random();
            int attendanceRoll = random.nextInt(bound:10); // 10% 未到機率

            if (attendanceRoll == 0) {
                this.attendanceStatus = 0; // 未到
                this.activityChoice = -1; // 未到的學生沒有活動選擇
            } else {
                this.attendanceStatus = 1; // 簽到
                this.activityChoice = random.nextInt(bound:4) + 1; // 1, 2, 3 之間的隨機數
            }
        }
```

```java
private void sendIdentityToServer() {
    if (out != null) {
        // out.println("STUDENT:" + clientStudentName);
        LoginInfo info = new LoginInfo(AuthType.STUDENT, id, name);
        out.println(info.toJson());
        // System.out.println(clientStudentName + " 已發送身份識別。");
    } else {
        System.err.println(clientStudentName + " 輸出流未初始化，無法發送身份。");
    }
}

private void sendAttendanceStatusToServer(int status) {
    if (out != null) {
        String statusString = (status == 1) ? "Y" : "N";
        out.println("ATTENDANCE:" + statusString);
        System.out.println(clientStudentName + " 已發送簽到狀態: " + statusString);
    } else {
        System.err.println(clientStudentName + " 輸出流未初始化，無法發送簽到狀態。");
    }
}
```

分工-SERVER

```
Socket clientSocket = serverSocket.accept();
PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
BufferedReader in =
    new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
String loginData = in.readLine();
LoginInfo info = LoginInfo.fromJson(loginData);
String id = info.getId();
String name = info.getName();
// 學生登入 student
if (info.getAuthType().equals(AuthType.STUDENT)) {
    StudentHandler handler = new StudentHandler(clientSocket, id, name);
    students.add(handler);
    String studentKey = "s_" + info.getId();
    if (!memoryCache.containsKey(studentKey))
        memoryCache.put(studentKey, new StudentInfo(handler)); // 加入 cache
    new Thread(handler).start();
    System.out.printf("%s 號 %s 已進入教室\n", info.getId(), info.getName());
} // 導師登入 gmtools
else if (info.getAuthType().equals(AuthType.TEACHER)) {
    TeacherHandler handler = new TeacherHandler(clientSocket, id, name);
    teachers.add(handler);
    out.println(String.format("%s 老師您已登入伺服器", name));
    new Thread(handler).start();
    System.out.printf("%s 老師已進入教室\n", info.getName());
}
```

```
教室伺服器已啟動..
4 號 學員 已進入教室
4 學員 在滑手機
4 學員 在滑手機
2 號 學員 已進入教室
2 學員 已簽到
簽到成功
2 學員 在喝水
2 學員 在喝水
1 號 學員 已進入教室
1 學員 已簽到
簽到成功
1 學員 在喝水
1 學員 在喝水
5 號 學員 已進入教室
5 學員 已簽到
簽到成功
5 學員 在滑手機
5 學員 在滑手機
3 號 學員 已進入教室
3 學員 已簽到
簽到成功
3 學員 在喝水
3 學員 在喝水
尼奧 老師已進入教室
```

伺服器 - ClassroomServer
功能摘要：

- 依據登入身分（教師/學生）建立對應連線處理器。
- 使用Socket傳遞JSON格式命令進行互動。
- 記憶體快取(memoryCache)儲存學生出席與行為紀錄。
- 支援擴充的功能指令處理模組（如點名、廣播、清場等。
- 教師指令 → 伺服器處理 → 回傳訊息給學生端或即時廣播。
- 使非同步處理，讓每位學生/教師以獨立執行緒處理，保持即時互動

分工-GMTOOLS

依功能選單製作對應命令，
搭配Server實作對應功能
即可完成Tools的互動增修

```
尼奧 老師您已登入伺服器

===== 功能選單 =====
1. 廣播
2. 統計人數
3. 請同學來找老師
4. 留話給學員
5. 教室清場
q. 離開
請選擇功能: 2

目前教室學生人數：5

3 學員 [未簽到] [在喝水]

5 學員 [已簽到] [在滑手機]

4 學員 [已簽到] [在喝水]

2 學員 [未簽到] [在滑手機]

1 學員 [未簽到] [在講話]
```

```java
System.out.println(x:"\n===== 功能選單 =====");
System.out.println(x:"1. 廣播");
System.out.println(x:"2. 統計人數");
System.out.println(x:"3. 請同學來找老師");
System.out.println(x:"4. 留話給學員");
System.out.println(x:"5. 教室清場");
System.out.println(x:"q. 離開");
System.out.print(s:"請選擇功能: ");
String target = "";
String content = "";
// 用輸入功能卡迴圈循環
String choice = scanner.nextLine().trim().toLowerCase();
switch (choice) {
    case "1":
        System.out.println(x:"請輸入要廣播的訊息");
        content = scanner.nextLine().trim();
        doAction(new Message(type:"broadcast", target, content));
        break;
    case "2":
        doAction(new Message(type:"count", target, content));
        break;
    case "3":
        System.out.println(x:"請輸入您要尋找的學員代號");
        target = scanner.nextLine().trim().toLowerCase();
        doAction(new Message(type:"find", target, content));
        break;
    case "4":
        System.out.println(x:"請輸入學員代號");
        target = scanner.nextLine().trim().toLowerCase();
        System.out.println(x:"請輸入您想傳遞的訊息");
        content = scanner.nextLine().trim();
        doAction(new Message(type:"memo", target, content));
        break;
    case "5":
        doAction(new Message(type:"clearroom", target, content));
        break;
    case "q":
        doAction(new Message(type:"quit", target, content));
```

未來可擴充

- 封包內容加密與驗證機制（強化安全性）
- 客戶端登入機制與帳號資料庫連結
- 權限等級控管（支援教師、學生、管理員）
- Session 管理與使用者身份驗證（Login / Logout）
- 封包指令模組化，可快速擴充功能種類
- 自動紀錄指令歷程與互動 Log（可供日後分析）
- 封包格式升級為更完整的 JSON Schema 定義

應用場景

- Client-Server 架構練習與展示
- Socket 封包格式與反序列化演練
- 權限控管系統設計與驗證流程實作
- 小型多人互動系統的原型開發平台
- 可進一步整合 Web 前端介面或 REST API

此專案時間有限以及考量展示環境，因此將架構簡化再簡化，未來很多人連線架構都可以延伸運用

THE END

指導老師：簡志軒

製作成員：

張世杰　JACK
郭昀翰　MILLER
馮立忠　NEIL