***Object-Oriented Analysis and Design***

***UML Class Diagram Generator***

**HW#3**

**107598063劉冠志**

**107598058朱峻平**

**107598069張嗣岱**

目錄

[1 Requirement Document 3](#_Toc4506032)

[1.1 Change History 3](#_Toc4506033)

[1.2 problem statement 3](#_Toc4506034)

[1.3 System Context Diagram 4](#_Toc4506035)

[1.4 Summary of System Features 4](#_Toc4506036)

[1.5 Use Case Diagram 5](#_Toc4506037)

[1.6 Use Cases 6](#_Toc4506038)

[1.6.1 新增未填值之UML Class 元件 6](#_Toc4506039)

[1.6.2 讀取User輸入之欄位資料 8](#_Toc4506040)

[1.6.3 建立User Level,提供不同Level不同服務 9](#_Toc4506041)

[1.6.4 輸出User的UML Class Diagram透過多種方式 10](#_Toc4506042)

[1.6.5 連接UML Class 元件透過手動操作 11](#_Toc4506043)

[1.6.6 備份輸出的檔案 13](#_Toc4506044)

[1.7 Non‐functional Requirements and Constraints 14](#_Toc4506045)

[1.8 Glossary 16](#_Toc4506046)

[1.9 The development language 16](#_Toc4506047)

[2 Domain class model 17](#_Toc4506048)

[2.1 Domain Class Diagram Showing Only Concepts 17](#_Toc4506049)

[2.1.1 Classes Identified 17](#_Toc4506050)

[2.1.2 Bad Classes 18](#_Toc4506051)

[2.1.3 Good Classes 19](#_Toc4506052)

[2.2 Add Associations 20](#_Toc4506053)

[2.3 Add Attributes 21](#_Toc4506054)

[3.Logical Architecture 22](#_Toc4506055)

[4. Use-Case Realizations with GRASP Patterns 23](#_Toc4506056)

[4.1 System Sequence Diagram 23](#_Toc4506057)

[4.2 Operation Contract 26](#_Toc4506058)

[4.3 Operation Sequence Diagram 26](#_Toc4506059)

[Design Class Model 26](#_Toc4506060)

# 1 Requirement Document

## 1.1 Change History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Description of Version | Date Completed |
| 0.10 | 全體組員 | Problem statement | 2019/2/21 |
| 0.20 | 全體組員 | Requirement Document | 2019/3/7 |
| 0.30 | 全體組員 | Domain Class Model | 2019/3/26 |

## 1.2 problem statement

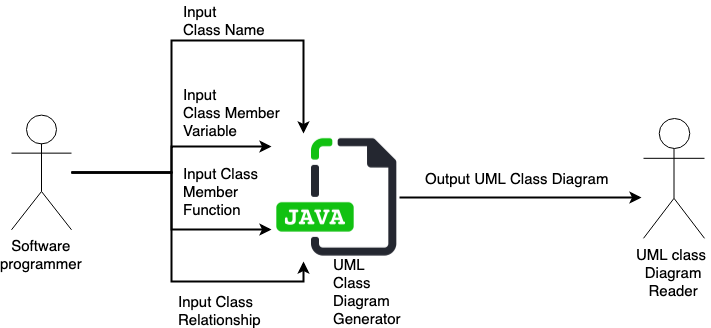
本軟體是為了需要UML Class Diagram協助開發的程式人員而設計，適用於Windows作業系統上。

在軟體開發中，設計UML Class Diagram是不可或缺的，但大多數的程式開發人員，僅使用紙筆或簡易的繪圖工具，使得畫出來的UML Class Diagram不易保存、不易閱讀。

本軟體可以讓程式開發人員透過GUI簡易的設計且輸出UML Class Diagram。省去手寫的時間及提高可閱讀性，並補齊其他簡易繪圖工具不足之功能，即使是初學者，只要熟悉本軟體的操作模式，亦可繪製出準確無誤的UML Class Diagram。

程式開發人員只需給定Class名稱、成員變數、成員函數及其繼承或實作關係，透過本軟體的運算及繪圖，即可產生對應的UML Class Diagram。

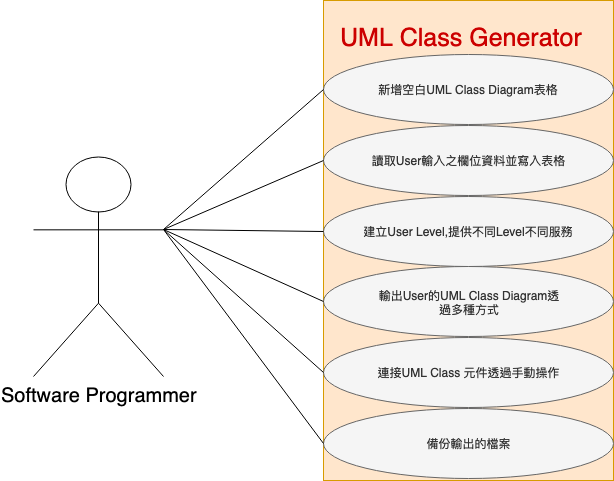
## 1.3 System Context Diagram

******

## 1.4 Summary of System Features

|  |  |
| --- | --- |
| **Feature ID** | **Description** |
| FEA-01 | 成功讀入USER之INPUT,不產生錯誤 |
| FEA-02 | 可新增修改刪除UML CLASS DIAGRAM |
| FEA-03 | 提供UML CLASS DIAGRAM備份檔案 |
| FEA-04 | 可於視窗顯示UML CLASS DIAGRAM |
| FEA-05 | 輸出UML CLASS DIAGRAM之檔案 |

## 1.5 Use Case Diagram

******

## 1.6 Use Cases

|  |  |
| --- | --- |
| Use Case ID | Use Case Name |
| UC‐01 | 新增空白UML Class Diagram  表格 |
| UC‐02 | 讀取User輸入之欄位資料  並寫入表格 |
| UC-03 | 建立User Level,提供不同Level不同服務 |
| UC-04 | 輸出User的UML Class Diagram透過多種方式 |
| UC-05 | 連接UML Class 元件透過手動操作 |
| UC-06 | 備份輸出的檔案 |

### 1.6.1 新增未填值之UML Class 元件

|  |  |
| --- | --- |
| **Use Case ID** | UC‐01 |
| **Scope** | UML Class Diagram Generator |
| **Level** | User-goal |
| **Primary Actor** | Software Programmer |
| **Stakeholders & Interests** | Software Programmer  想要UML Class Diagram Generator自動產生未填值之UML Class 元件。 |
| **Preconditions** | UML Class Diagram Generator  可正常執行。 |
| **Postcondition** | 自動產生構成未填值之UML Class 元件。 |
| **Main Success Scenario** | 1. Software Programmer開啟   UML Class Diagram Generator  應用程式。  2. 程式被開啟，並顯示功能列表。  3. Software Programmer新增空白UML Class Diagram。  4. 程式顯示UML Class Diagram  草圖與輸入欄位。  5. Software Programmer決定進行輸出。  6. 程式跳出詢問是否不輸入欄位值之提醒。  7. Software Programmer決定不輸入欄位值。  8. 程式顯示並輸出包含Software Programmer所選之UML Class元件的Diagram。 |
| **Extensions** | 1. UML Class Diagram Generator   應用程式無法正常開啟:   * 通知Software Programmer重新開啟此應用程式。  1. 功能列表顯示異常 :  * 應用程式重新刷新頁面。  1. UML Class Diagram的草圖無法顯示 :  * 跳出提醒詢問是否需要重新輸出亦或中斷操作。 * Software Programmer回應程式。   4a. 若Software Programmer  選擇取消:   * 紀錄錯誤訊息，並回到功能列表。   4b. 若Software Programmer  選擇重試:   * 重新刷新頁面輸出草圖。   8. 程式輸出UML Class Diagram圖  不符合格式:   * 使用者選擇重新產生。 * 程式再次輸出UML Class Diagram。 |
| **Special Requirements** | * 確保產生之表格不可跑版 * 確保產生之欄位不可有缺少 |
| **Technology And Data Variations List** | None. |
| **Frequency of Occurrence** | 時常發生 (若需新增表格) |
| **Miscellaneous** | 用於防止Programmer拉取的元件及欄位大小不一致。 |

### 1.6.2 讀取User輸入之欄位資料

|  |  |
| --- | --- |
| **Use Case ID** | UC‐02 |
| **Scope** | UML Class Diagram Generator |
| **Level** | User-goal |
| **Primary Actor** | Software Programmer |
| **Stakeholders & Interests** | Software Programmer  只需輸入欄位資料程式可以正確讀取  並寫入表格。 |
| **Preconditions** | Software Programmer已輸入  欄位資料。 |
| **Postcondition** | 正確讀取欄位資料並寫入表格。 |
| **Main Success Scenario** | 1. Software Programmer開啟   UML Class Diagram Generator  應用程式。   1. 程式被開啟，並顯示功能列表。 2. Software Programmer   填入各欄位值。   1. 程式讀入各欄位值。 2. 程式顯示UML Class Diagram   草圖。   1. Software Programmer決定進行輸出。 2. 程式顯示並輸出具欄位值之   UML Class Diagram。 |
| **Extensions** | 1. UML Class Diagram Generator  應用程式無法正常開啟:   * 應用程式通知Software Programmer   重新開啟此應用程式。   1. 功能列表顯示異常 :  * 應用程式重新刷新頁面。   4. 程式讀入各欄位值時產生錯誤   * 應用程式重新刷新頁面。 * 應用程式再次讀入欄位值。  1. UML Class Diagram的草圖無法顯示 :  * 應用程式跳出Dialog詢問是否需要重新輸出亦或中斷操作。 * User回應程式。   5a. 若Software Programmer  選擇取消:   * 應用程式紀錄錯誤訊息，並回到功能列表。   5b. 若Software Programmer  選擇重試:   * 應用程式重新刷新頁面輸出具欄位值之草圖。   8. 程式輸出UML Class Diagram圖  不符合格式:   * 使用者選擇重新產生。 * 應用程式再次輸出UML Class Diagram。 |
| **Special Requirements** | * 確保值正確讀入沒有產生Error * 確保值有正確填入表格欄位 |
| **Technology And Data Variations List** | None. |
| **Frequency of Occurrence** | 時常發生 (若欄位有填入值) |
| **Miscellaneous** | 用於防止Programmer自行填入欄位值會有誤。 |

### 1.6.3 建立User Level,提供不同Level不同服務

|  |  |
| --- | --- |
| **Use Case ID** | UC‐03 |
| **Scope** | UML Class Diagram Generator |
| **Level** | User-goal |
| **Primary Actor** | Software Programmer |
| **Stakeholders & Interests** | Software Programmer  想要有更方便的UML Class Diagram 產生方式。 |
| **Preconditions** | UML Class Diagram Generator  可依特定屬性將User劃分成不同Level。 |
| **Postcondition** | 依照Programmer 的User Level高低，提供User更方便的UML Class Diagram 產生方式 |
| **Main Success Scenario** | 1.Software Programmer開啟  UML Class Diagram Generator應用程式。  2.程式被開啟，並顯示該  Software Programmer的User Level  3 3.程式顯示該User Level的其他產生方式。  4. Software Programmer點選其他產生方式。  5. 程式依照其他方式產生  UML Class Diagram並顯示。 |
| **Extensions** | 1. UML Class Diagram Generator應用程式無法正常開啟:  * 應用程式通知Software Programmer   重新開啟此應用程式。  3.User Level與其他產生方式不符合 :   * 應用程式重新刷新頁面。   4. 輸出UML class Diagram不符合格式 :   * Software Programmer 選擇重試。 * 應用程式重新輸出UML Class Diagram。 |
| **Special Requirements** | * 確保其他產生方式正常執行 * 確保UML Class Diagram正常輸出 |
| **Technology And Data Variations List** | None. |
| **Frequency of Occurrence** | 時常發生 (若需其他產生方式) |
| **Miscellaneous** | 用於讓Software Programmer更簡單的產生UML Class Diagram。 |

### 1.6.4 輸出User的UML Class Diagram透過多種方式

|  |  |
| --- | --- |
| **Use Case ID** | UC‐04 |
| **Scope** | UML Class Diagram Generator |
| **Level** | User-goal |
| **Primary Actor** | Software Programmer |
| **Stakeholders & Interests** | Software Programmer  想要有更多的UML Class Diagram輸出方式。 |
| **Preconditions** | UML Class Diagram Generator  可以正確地輸出UML Class Diagram。 |
| **Postcondition** | 依照Software Programmer所選的方式正確的輸出UML Class Diagram |
| **Main Success Scenario** | 1.Software Programmer開啟  UML Class Diagram Generator應用程式。  2.Software Programmer選擇欲輸出的  UML Class Diagram。  3 3.程式顯示可輸出的方式。  4. Software Programmer選擇想要的輸出方式。  5. 應用程式依照所選方式輸出UML Class Diagram。 |
| **Extensions** | 1.UML Class Diagram Generator應用程式無法正常開啟:   * 應用程式通知Software Programmer重新開啟此應用程式。   2.沒有可輸出之UML Class Diagram :   * 應用程式詢問Software Programmer是否要新增。   2.a 若選擇新增 :   * 應用程式新增空白的UML Class Diagram。   2.b 若選擇不新增 :   * 回到功能列表。   5. 輸出UML Class Diagram不符合格式:   * Software Programmer 重新選擇輸出方式。 * 應用程式重新輸出UML Class Diagram。 |
| **Special Requirements** | * 確保能以Software Programmer所選的輸出方式正確輸出 |
| **Technology And Data Variations List** | None. |
| **Frequency of Occurrence** | 時常發生 (若需其他輸出方式) |
| **Miscellaneous** | 用於讓Software Programmer更方便的輸出及存放UML Class Diagram。 |

### 1.6.5 連接UML Class 元件透過手動操作

|  |  |
| --- | --- |
| **Use Case ID** | UC‐05 |
| **Scope** | UML Class Diagram Generator |
| **Level** | User-goal |
| **Primary Actor** | Software Programmer |
| **Stakeholders & Interests** | Software Programmer  可以按照自己的喜好連接UML Class 元件。 |
| **Preconditions** | Software Programmer 已經將資料填入但未選擇元件彼此連接關係。 |
| **Postcondition** | Software Programmer 可以手動將元件連接至其他元件。 |
| **Main Success Scenario** | 1. 程式顯示UML Class Diagram  草圖。  2. Software Programmer決定手動連接元件。  3. Software Programmer透過滑鼠拖曳連線。  4. Software Programmer決定進行輸出。  5. 程式顯示並輸出包含手動連線之UML Class Diagram。 |
| **Extensions** | 1. UML Class Diagram的草圖無法顯示 :   * 應用程式跳出提醒詢問是否需要重新輸出亦或中斷操作。 * Software Programmer回應程式。   1a. 若Software Programmer  選擇取消:   * 應用程式紀錄錯誤訊息，並回到功能列表。   1b. 若Software Programmer  選擇重試:   * 應用程式重新刷新頁面輸出草圖。   3.若連線不如Software Programmer預期 :  3a.若Software Programmer  選擇上一步 :   * 應用程式將畫面回到上一步。   3b.若Software Programmer選  則清除全部元件關係:   * 畫面顯示沒有連接關係之UML Class Diagram。 |
| **Special Requirements** | 確保兩元件之間只能有一條連接關係。 |
| **Technology And Data Variations List** | None. |
| **Frequency of Occurrence** | 時常發生 (若需手動連接) |
| **Miscellaneous** | 提供Software Programmer在自動產生時未選擇元件關係的補救方式。 |

### 1.6.6 備份輸出的檔案

|  |  |
| --- | --- |
| **Use Case ID** | UC‐06 |
| **Scope** | UML Class Diagram Generator |
| **Level** | User-goal |
| **Primary Actor** | Software Programmer |
| **Stakeholders & Interests** | Software Programmer  為了防止檔案被無意刪除，有額外的儲存備份。 |
| **Preconditions** | Software Programmer 可以將檔案輸出。 |
| **Postcondition** | UML Class Diagram 有備份的儲存地點。 |
| **Main Success Scenario** | 1. 程式依照所選方式輸出UML Class Diagram。  2. 程式跳出通知是否要指定另外的備份地點。  3.Software Programmer 輸入欲備份的路徑。  4.Software Programmer決定輸出。  5.程式至指定路徑完成備份。  6.完成備份後跳出通知。 |
| **Extensions** | 1.輸出UML Class Diagram不符合格式:   * Software Programmer 重新選擇輸出方式。 * 應用程式重新輸出UML Class Diagram。   2. 沒有跳出通知:  2a. 程式預設備份路徑在桌面備份資料夾裡。   1. 沒有備份檔案成功:  * 應用程式跳出尚未備份檔案通知。 * Software Programmer再次執行步驟一。 |
| **Special Requirements** | 需要預設一個備份檔案的路徑 |
| **Technology And Data Variations List** | None. |
| **Frequency of Occurrence** | 時常發生 (若需備份) |
| **Miscellaneous** | 用於防止檔案遺失。 |

## 1.7 Non‐functional Requirements and Constraints

|  |  |  |
| --- | --- | --- |
| **NFR ID** | **Category** | **Description** |
| NFR-01 | Performance | 使用者按下新增，應用程式必須立刻反應，不可停滯太久。 |
| NFR-02 | Reliability | *將欄位值寫入表格不可跑版或誤填值*。 |
| NFR-03 | Portability | *應用程式必須支援跨平台*。 |
| NFR-04 | Modifiability | *程式分工清楚以便修改或擴充功能*。 |
| NFR-05 | Reliability | 輸出的UML Class Diagram 不可跑版或錯誤 |
| NFR-06 | Readability | Code style要一致，variable及method命名要有意義 |
| NFR-07 | Reliability | 紀錄間距以防移動後大小跑版 |

## 1.8 Glossary

|  |  |
| --- | --- |
| Item | Definition or Description |
| UML Class Diagram | 是軟體工程的統一建模語言一種靜態結構圖，  該圖描述了系統的類別集合，類別的屬性和類別之間的關係。 |
| UML Class Diagram Generator | 正確讀入Software Programmer輸入之各欄位值可自動產生UML Class Diagram的產生器。 |
| Dialog | 用來在用戶介面中向用戶顯示[信息](https://zh.wikipedia.org/wiki/%E4%BF%A1%E6%81%AF)。 |
| 備份檔案 | 將欲輸出之檔案先行留存一份。 |
| User Level | 使用者的等級 |

## 1.9 The development language

為了跨平台所以使用”Java”撰寫。

# 2 Domain class model

## 2.1 Domain Class Diagram Showing Only Concepts

### 2.1.1 Classes Identified

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UML Class Diagram Generator** | **刷新頁面** | **產生Diagram** | **Dialog** | **功能列表** | **重新**  **開啟** |
| **UML Class Diagram** | **Class**  **屬性** | **草圖** | **中斷操作** |  |  |
| **Class 關係Generator** | **Class**  **關係線** | **自動連接Class關係線** | **手動連接Class關係線** |  |  |
| **Class 元件Generator** | **Class**  **元件** |  |  |  |  |
| **Recorder** | **紀錄** |  |  |  |  |
| **Field Manager** | **讀取欄位值** | **欄位值** | **輸入欄位值** |  |  |
| **User Manager** | **User Level** | **Crash** | **Level Id** |  |  |
| **Output Manager** | **備份**  **路徑** | **備份檔案** | **錯誤訊息** | **輸出Diagram** |  |
| **Software Programmer** |  |  |  |  |  |

***以上是從Use case節錄的名詞及行為***

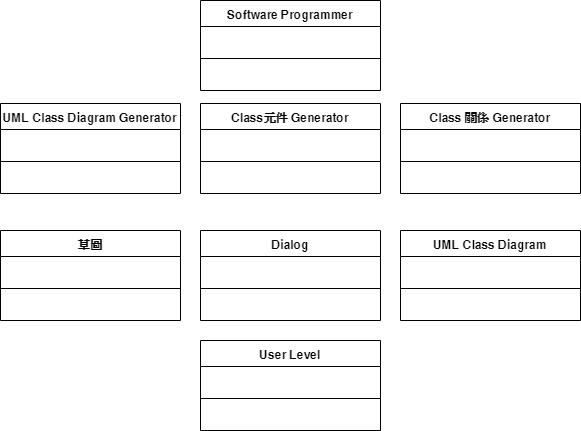
### 2.1.2 Bad Classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vague** | **Attributes** | **Operations** | **Implementation Construction** | **Roles** |
| Crash | 錯誤訊息 | 讀取欄位值 | Class 元件 | Recorder |
|  | Class  屬性 | 輸出 | Class關係線 | Field Manager |
|  | 備份路徑 | 重新開啟 | 功能列表 | User Manager |
|  | 欄位值 | 刷新頁面 |  | Output Manager |
|  | Level Id | 產生Diagram |  |  |
|  |  | 中斷操作 |  |  |
|  |  | 備份檔案 |  |  |
|  |  | 紀錄 |  |  |
|  |  | 自動連接Class關係線 |  |  |
|  |  | 手動連接Class關係線 |  |  |

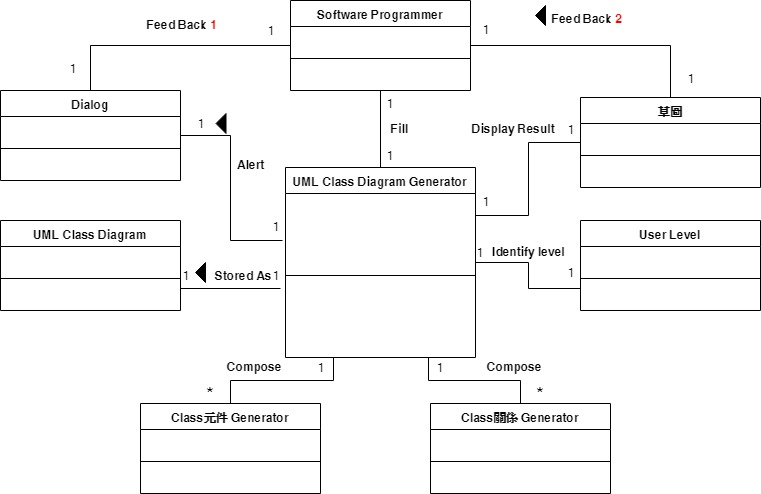
* **Vague：**定義模糊不清
* **Attributes：**列為屬性
* **Operation：**操作步驟
* **Implementation Construction：**實作產生的結構
* **Roles：**角色介面

### 2.1.3 Good Classes

|  |  |  |  |
| --- | --- | --- | --- |
| UML Class Diagram Generator | User Level | 草圖 | Dialog |
| UML Class Diagram |  |  |  |
| Class 元件 Generator |  |  |  |
| Class 關係 Generator |  |  |  |
| Software Programmer |  |  |  |

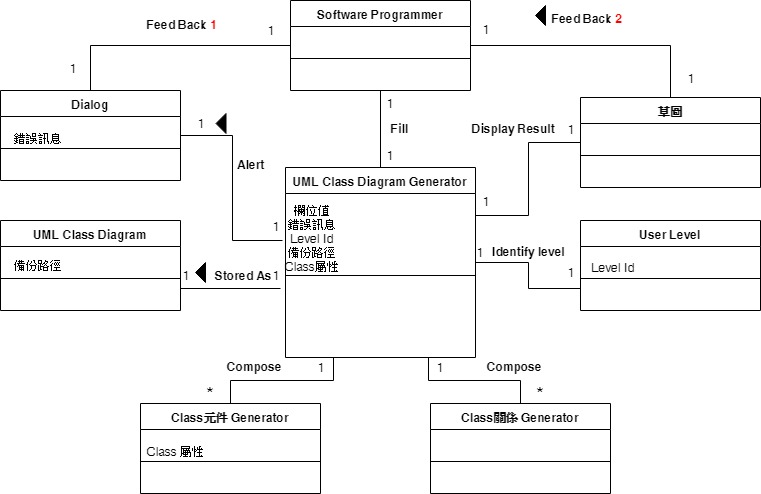


## 2.2 Add Associations

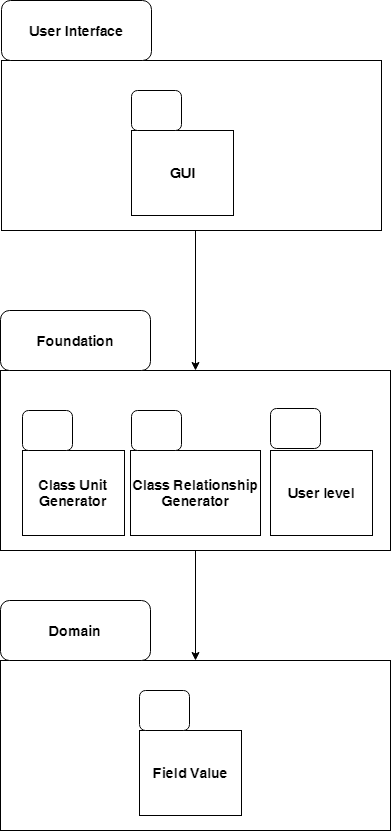


|  |  |
| --- | --- |
| Use | Software Programmer使用UML Class Diagram Generator。 |
| Feed Back 1 | Software Programmer得到UML Class Diagram Generator產生之互動提醒。 |
| Feed Back 2 | Software Programmer得到UML Class Diagram Generator產生之草圖回饋。 |
| Alert | UML Class Diagram Generator產生警示訊息Dialog。 |
| Display Result | UML Class Diagram Generator顯示欲輸出之草圖。 |
| Stored As | UML Class Diagram Generator儲存欲輸出之  UML Class Diagram。 |
| Identify level | UML Class Diagram Generator辨別不同使用者等級及權限透過User level。 |
| Compose | Class元件 Generator及Class 關係 Generator構成UML Class Diagram Generator |

## 2.3 Add Attributes



# 3.Logical Architecture



**GUI :** UML class Diagram Generator 的相關類別

**Class Unit Generator:**  Class元件生成的相關類別

**Class Relationship Generator:**  Class關係生成的相關類別

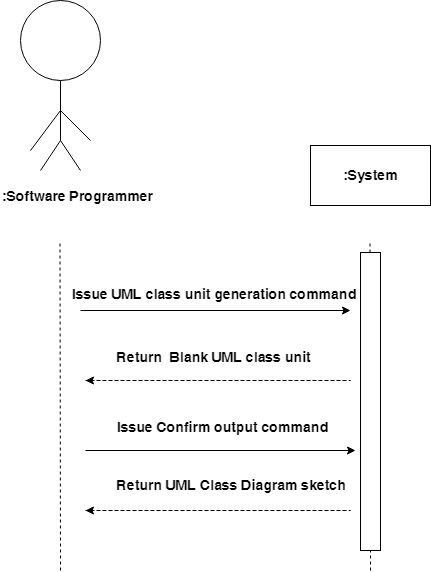
**User level:** 使用者階級權限相關類別

**Field Value:** 使用者輸入值相關類別

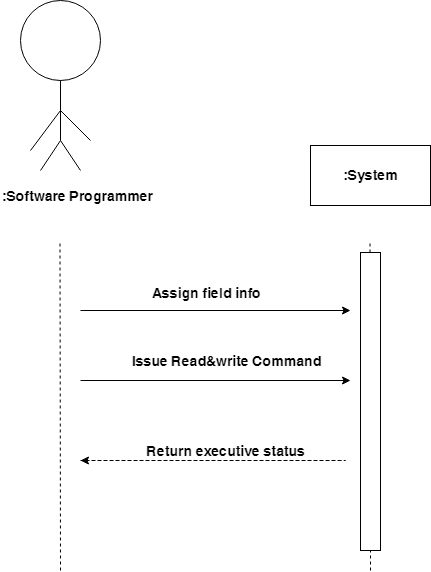
# 4. Use-Case Realizations with GRASP Patterns

## 4.1 System Sequence Diagram

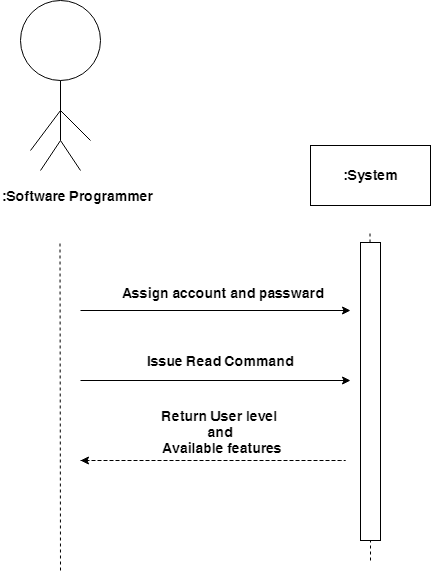
**新增空白UML Class Diagram表格**



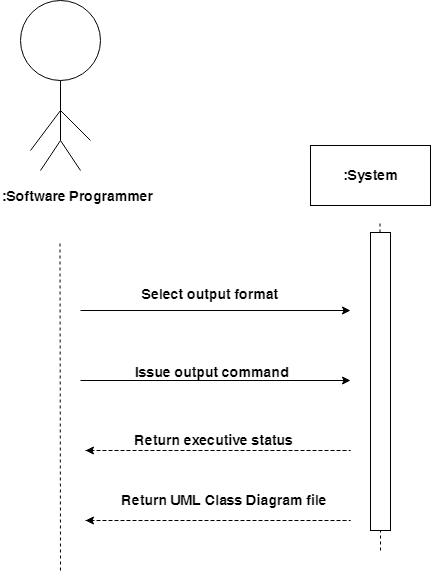
**讀取User輸入之欄位資料並寫入表格**



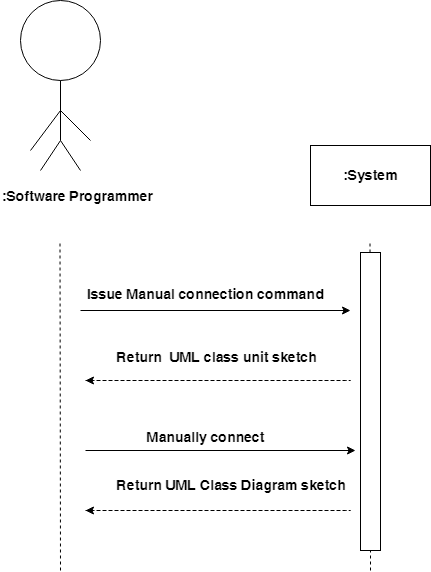
**建立User Level,提供不同Level不同服務**



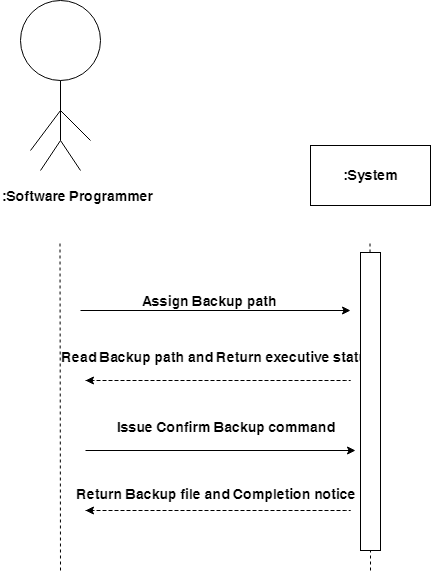
**輸出User的UML Class Diagram透過多種方式**

****

**連接UML Class 元件透過手動操作**



**備份輸出的檔案**

****

## 4.2 Operation Contract

|  |  |
| --- | --- |
| **Contract ID** | **Operation Name** |
| **Contract01** |  |
| **Contract02** |  |
| **Contract03** |  |
| **Contract04** |  |
| **Contract05** |  |
| **Contract06** |  |
| **Contract07** |  |
| **Contract08** |  |

|  |  |
| --- | --- |
| **Operation** |  |
| **Cross References** |  |
| **Preconditions** |  |
| **Postconditions** |  |

## 4.3 Operation Sequence Diagram

# Design Class Model

***Measurement***

hw1:2019/02/21 ~ 2019/02/22 Total : 3hrs

hw2:2019/02/28 ~ 2019/3/07 Total : 8hrs

hw3:2019/03/11 ~ 2019/3/14 Total : 6hrs