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

***UML Class Diagram Generator***

**HW#7**

**107598063劉冠志**

**107598058朱峻平**

**107598069張嗣岱**

[1 Requirement Document 4](#_Toc10549214)

[1.1 Change History 4](#_Toc10549215)

[1.2 problem statement 6](#_Toc10549216)

[1.3 System Context Diagram 6](#_Toc10549217)

[1.4 Summary of System Features 7](#_Toc10549218)

[1.5 Use Case Diagram 7](#_Toc10549219)

[1.6 Use Cases 8](#_Toc10549220)

[1.6.1 新增空白UML Class Diagram元件 8](#_Toc10549221)

[1.6.2 讀取User輸入之欄位資料並寫入表格 9](#_Toc10549222)

[1.6.3 建立User Level,提供不同Level不同服務 11](#_Toc10549223)

[1.6.4 輸出User的UML Class Diagram 12](#_Toc10549224)

[1.6.5 修改UML Class Diagram手動 13](#_Toc10549225)

[1.6.6 備份輸出的檔案 15](#_Toc10549226)

[1.7 Non‐functional Requirements and Constraints 17](#_Toc10549227)

[1.8 Glossary 18](#_Toc10549228)

[1.9 The development language 18](#_Toc10549229)

[2 Domain class model 19](#_Toc10549230)

[2.1 Domain Class Diagram Showing Only Concepts 19](#_Toc10549231)

[2.1.1 Classes Identified 19](#_Toc10549232)

[2.1.2 Bad Classes 20](#_Toc10549233)

[2.1.3 Good Classes 21](#_Toc10549234)

[2.2 Add Associations 22](#_Toc10549235)

[2.3 Add Attributes 23](#_Toc10549236)

[3.Logical Architecture 24](#_Toc10549237)

[4. Use-Case Realizations with GRASP Patterns 25](#_Toc10549238)

[4.1 System Sequence Diagram 25](#_Toc10549239)

[4.2 Operation Contract 28](#_Toc10549240)

[4.2.1 selectUserLevel 28](#_Toc10549241)

[4.2.2 inputField 29](#_Toc10549242)

[4.2.3 showSketch 29](#_Toc10549243)

[4.2.4 backUpDiagram 30](#_Toc10549244)

[4.2.5 executeAddRelation 30](#_Toc10549245)

[4.2.6 outputDiagram 30](#_Toc10549246)

[4.2.7 deleteClassUnitByName 31](#_Toc10549247)

[4.2.8 clearDiagram 31](#_Toc10549248)

[4.2.9 addClassUnit 31](#_Toc10549249)

[4.3 Operation Sequence Diagram 32](#_Toc10549250)

[4.3.1 selectUserLevel 32](#_Toc10549251)

[4.3.2 inputField 33](#_Toc10549252)

[4.3.3 showSketch 34](#_Toc10549253)

[4.3.4 backUpDiagram 34](#_Toc10549254)

[4.3.5 executeAddRelation 35](#_Toc10549255)

[4.3.6 outputDiagram 35](#_Toc10549256)

[4.3.7 deleteClassUnitByName 36](#_Toc10549257)

[4.3.8 clearDiagram 36](#_Toc10549258)

[4.3.9 addClassUnit 37](#_Toc10549259)

[4.4Design Class Model 37](#_Toc10549260)

[5 Implementation Class Model 38](#_Toc10549261)

[5.1 Draw an implementation class diagram for your system (including associations, attributes and methods) (new) 38](#_Toc10549262)

[5.2 Show the difference between implementation class model and design class model (new) 39](#_Toc10549263)

[5.3 Calculate Line of Code (new) 40](#_Toc10549264)

[6 Programming 41](#_Toc10549265)

[6.1 Snapshots of system execution 41](#_Toc10549266)

[6.2 Source Code Listing 42](#_Toc10549267)

[6.2.1UmlGUI 42](#_Toc10549268)

[6.2.2View 43](#_Toc10549269)

[6.2.3 Tool 52](#_Toc10549270)

[6.2.4 ClassFormat 54](#_Toc10549271)

[6.2.5 ConcreteFormat 56](#_Toc10549272)

[6.2.6 InterfaceFormat 57](#_Toc10549273)

[6.2.7 Relation 57](#_Toc10549274)

[6.2.8 Listener 59](#_Toc10549275)

[6.2.9 ListenerHandler 62](#_Toc10549276)

[6.2.10 ArrangeCalculator 63](#_Toc10549277)

[6.2.11 ClassRelationGenerator 66](#_Toc10549278)

[6.2.12 Drawable 66](#_Toc10549279)

[6.2.13 UMLClassDiagram 66](#_Toc10549280)

[6.2.14 UMLClassDiagramDrawer 68](#_Toc10549281)

[6.2.15 ClassDetailInfo 69](#_Toc10549282)

[6.2.16 ClassMemberAbstract 71](#_Toc10549283)

[6.2.17 ClassRelationship 71](#_Toc10549284)

[6.2.18 MemberFunction 72](#_Toc10549285)

[6.2.19 MemberVariable 73](#_Toc10549286)

[6.2.20 ClassDetailInfoDTO 75](#_Toc10549287)

[6.2.21 ClassUnitGenerator 76](#_Toc10549288)

[6.2.22 ClassFormatOutputDTO 77](#_Toc10549289)

[6.2.23 DiagramReader 78](#_Toc10549290)

[6.2.24 DiagramWriter 79](#_Toc10549291)

[7 Unit Testing 81](#_Toc10549292)

[7.1 Snapshots of testing result 81](#_Toc10549293)

[7.2 Unit Testing Code Listing 82](#_Toc10549294)

[7.2.1 ClassDetailInfoDTOTest 82](#_Toc10549295)

[7.2.2 ClassDetailInfoTest 83](#_Toc10549296)

[7.2.3 ClassRelationshipTest 84](#_Toc10549297)

[7.2.4 MemberFunctionTest 85](#_Toc10549298)

[7.2.5 MemberVariableTest 86](#_Toc10549299)

[7.2.6 UMLClassDiagramTest 87](#_Toc10549300)

[7.2.7 ArrangeCalculatorTest 89](#_Toc10549301)

[7.2.8 ClassRelationGeneratorTest 92](#_Toc10549302)

[7.2.9 ClassUnitGeneratorTest 93](#_Toc10549303)

[7.2.10 ClassFormatTest 94](#_Toc10549304)

[7.2.11 RelationTest 95](#_Toc10549305)

[7.2.12 ClassFormatOutputDTOTest 97](#_Toc10549306)

[7.2.13 RelationOutputDTOTest 98](#_Toc10549307)

[7.2.14 DiagramWriterTest 99](#_Toc10549308)

[Measurement 100](#_Toc10549309)

# 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 |
| 0.40 | 全體組員 | Use-Case Realizations with GRASP Patterns | 2019/4/15 |
| 0.50 | 全體組員 | Implementation Class Model | 2019/4/29 |
| 0.60 | 全體組員 | Modified Requirement Document   1. UC-01新增空白UML Class Diagram元件 2. UC-02讀取User輸入之欄位資料並寫入表格 3. UC-05修改UML Class Diagram透過手動操作   Domain model   1. New Domain classes showing only concepts 2. Add associations 3. Add attributes 4. Combine with the original domain model derived in iteration I. | 2019/5/8 |
| 0.70 | 全體組員 | Modified Requirement Document  1. Use Case Diagram  2. Use Case Name  3. Domain Class Model  4. Modified All Use Case  5. System Sequence Diagram  6. Add Code | 2019/6/2 |

## 1.2 problem statement

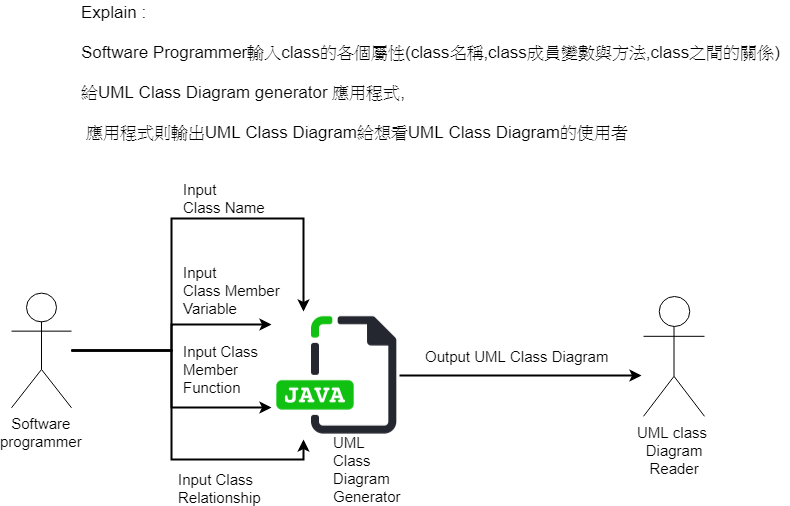
本軟體是為了需要UML Class Diagram協助開發的程式人員而設計，適用於Windows作業系統上。

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。

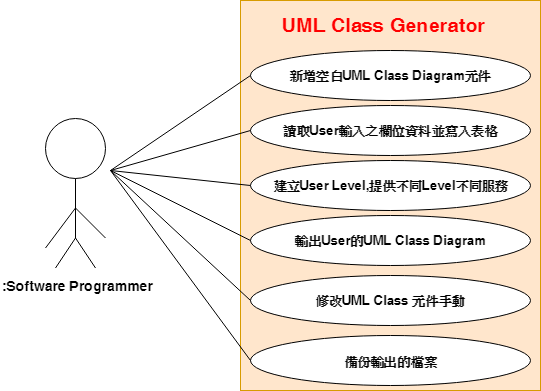
## 1.3 System Context Diagram

******

## 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

(這次修改的用紅色標記)

|  |  |
| --- | --- |
| Use Case ID | Use Case Name |
| UC‐01 | 新增空白UML Class Diagram  元件 |
| 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 |
| UC-05 | 修改UML Class Diagram手動 |
| UC-06 | 備份輸出的檔案 |

### 1.6.1 新增空白UML Class Diagram元件

(修改)

|  |  |
| --- | --- |
| **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** | 系統正常被開啟。 |
| **Postcondition** | 自動產生構成未填值之UML Class 元件。 |
| **Main Success Scenario** | 1. Software Programmer無填入各欄位值。  2. 系統顯示空白ClassInfo在Scree上。  3. Software Programmer確認並進行輸出。  4. 系統顯示UML Class Diagram sketch(僅含空白Class Unit)。 |
| **Extensions** | 2a. 系統顯示ClassInfo符合預期  2b. 系統顯示ClassInfo不符合預期   1. Software Programmer檢查各欄位是否存在殘留值 2. Software Programmer確實清除各欄位值 3. 系統再次顯示ClassInfo資訊   4a. UML Class Diagram sketch顯示  不符合預期:  1.Software Programmer操作系統  1.a. 若Software Programmer  選擇重新放棄此sketch:  1.回到功能列表  2.不填入欄位值  3.重新輸出sketch  1.b 若Software Programmer  1.選擇繼續使用此sketch:  2.User對其進行修改或其他操作 |
| **Special Requirements** | 1. 確保產生之表格不可跑版 2. 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輸入之欄位資料並寫入表格

(修改)

|  |  |
| --- | --- |
| **Use Case ID** | UC‐02 |
| **Scope** | UML Class Diagram Generator |
| **Level** | User-goal |
| **Primary Actor** | Interest |
| **Stakeholders & Interests** | Software Programmer  想要系統接收欄位資料並正確寫入表格。 |
| **Preconditions** | 系統正常被開啟。 |
| **Postcondition** | 正確讀取欄位資料並寫入表格。 |
| **Main Success Scenario** | 1. Software Programmer   填入各欄位值。   1. 系統接收各欄位值。 2. 系統顯示ClassInfo在Screen上。 3. Software Programmer確認並進行輸出。 4. 系統顯示UML Class Diagram   sketch。 |
| **Extensions** | 2-a. 系統讀ClassName發現不為空值  1. 系統正常存入ClassInfo  2-b. 系統讀ClassName發現為空值  1. 系統照常存入ClassInfo不發生程式Crash  2.自動將空白Class Name填入  ”New ClassName”  5. UML Class Diagram sketch顯示  不符合預期:  1.Software Programmer回應系統  1.a. 若Software Programmer  選擇重新放棄此sketch:  1.回到功能列表  2.重新填入欄位值  3.重新輸出sketch  1.b 若Software Programmer  選擇繼續使用此sketch:  1.User對其進行修改或其他操作 |
| **Special Requirements** | 1. 確保值正確讀入沒有產生Error 2. 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

(修改)

|  |  |
| --- | --- |
| **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系統。  2.程式被開啟，並顯示該  Software Programmer的User Level  3 3.程式顯示該User Level的可用服務。  4. Software Programmer點選服務。  5. 程式依照所選之方式產生  UML Class Diagram並顯示。 |
| **Extensions** | 1a. UML Class Diagram Generator系統無法正常開啟:  1. 應用程式通知Software Programmer  重新開啟此系統。  3a. User Level與其可用服務不符合 :  1. 應用程式重新刷新頁面。  4a. 輸出UML class Diagram不符合格式 :  1. Software Programmer 選擇重試。  2. 應用程式重新輸出UML Class Diagra。 |
| **Special Requirements** | 1. 確保其他產生方式正常執行  2. 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

(修改)

|  |  |
| --- | --- |
| **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系統。  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** | 1a. UML Class Diagram Generator系統無法正常開啟:  1. 系統通知Software Programmer重新開啟此應用程式。  2a. 沒有可輸出之UML Class Diagram :  1. 系統詢問Software Programmer是否要新增。  1.a 若選擇新增 :   * 系統新增空白的UML Class Diagram。   1.b 若選擇不新增 :   * 回到功能列表。   4a. 選擇以PNG格式輸出UML Class  Diagram。  1. 成功輸出UML Class Diagram。  4b. 選擇以JPG格式輸出UML Class  Diagram。  1. 成功輸出UML Class Diagram。  5a. 輸出UML Class Diagram不符合格式:  1. Software Programmer 重新選擇輸出方式。  2. 系統重新輸出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 修改UML Class Diagram手動

(修改)

|  |  |
| --- | --- |
| **Use Case ID** | UC‐05 |
| **Scope** | UML Class Diagram Generator |
| **Level** | User-goal |
| **Primary Actor** | Software Programmer |
| **Stakeholders & Interests** | Software Programmer  可以按照自己的需求修改UML Class Diagram。 |
| **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 製作完成UML Class Diagram並準備匯出。 |
| **Main Success Scenario** | 1. 系統顯示UML Class Diagram草圖。  2. Software Programmer 移動UML Class 元件。  3. Software Programmer連接UML Class元件。  4. Software Programmer刪除UML Class 元件。 |
| **Extensions** | 1a. 系統無法顯示UML Class Diagram的草圖  1.應用程式跳出提醒詢問是否要中斷操作  2.Software Programmer回應系統  1a. Software Programmer選擇取消  1.回到功能列表  1b. Software Programmer選擇重試  1.系統重新輸出草圖  1a.若系統仍然無法顯示草圖  1.重複執行1b.1直到成功  2a.系統顯示移動後的結果  2b.系統未顯示移動後的結果  1a.Software Programmer未選擇功能欄的  Move Unit選項  1b.Software Programmer 點選的位置未被  任何元件包含  1.系統顯示移動後結果  2.Software Programmer重新點選元  件並移動  3a.系統顯示連結後的結果  3b.系統未顯示連結後的結果  1a.Software Programmer未選擇功能欄  的Add Relation選項  1b.Software Programmer 按下的鼠標位置  未被任何元件包含  1.系統顯示移動後結果  2.Software Programmer重新連結元  件  1c. Software Programmer 放開的鼠標位置  未被任何元件包含  1.系統顯示移動後結果  2.Software Programmer重新連結元  件  4a.系統顯示刪除後的結果  4b.系統未顯示刪除後的結果  1a.Software Programmer未選擇功能欄的  Remove Unit選項  1b.Software Programmer 點選的位置未被  任何元件包含  1.系統顯示刪除後結果  2.Software Programmer重新點選要  刪除的元件 |
| **Special Requirements** | 確保元件只能在可顯示範圍內移動。 |
| **Technology And Data Variations List** | None. |
| **Frequency of Occurrence** | 時常發生 (若需修改UML Class Diagram) |
| **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

(修改)

|  |  |
| --- | --- |
| **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 已被備份 |
| **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。 |
| **Extensions** | 1a.輸出UML Class Diagram不符合格式:  1. Software Programmer 重新選擇輸出方式。  2. 應用程式重新輸出UML Class Diagram。  2a. 沒有跳出通知:  1. 程式預設備份路徑在桌面備份資料夾裡。  5a. 沒有備份檔案成功:  1. 應用程式跳出尚未備份檔案通知。  2. 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

(新增或修改以紅色標記)

## 2.1 Domain Class Diagram Showing Only Concepts

### 2.1.1 Classes Identified

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UML Class Diagram Generator** | **Refresh** | **Generate Diagram** | **Dialog** | **Function List** | **Restart** |
| **UML Class Diagram** | **Class**  **Attribute** | **Sketch** | **Interrupt Operation** |  |  |
| **Class Relation Generator** | **Class**  **Relation**  **Line** | **Automation**  **Connection**  **Class**  **Relation**  **Line** | **Manual**  **Connection**  **Class**  **Relation**  **Line** |  |  |
| **Class Unit Generator** | **Class**  **Unit** |  |  |  |  |
| **Recorder** | **Record** |  |  |  |  |
| **Field Manager** | **Read Field**  **Value** | **Field Value** | **Input Field Value** |  |  |
| **User Manager** | **User Level** | **Crash** | **Level Id** |  |  |
| **Output Manager** | **Backup Path** | **Backup**  **File** | **Error**  **Message** | **Output Diagram** |  |
| **Software Programmer** | Class Info | Screen | Canvas |  |  |
| Operator |  |  |  |  |  |
| User Level Manager |  |  |  |  |  |

***以上是從Use case節錄的名詞及行為***

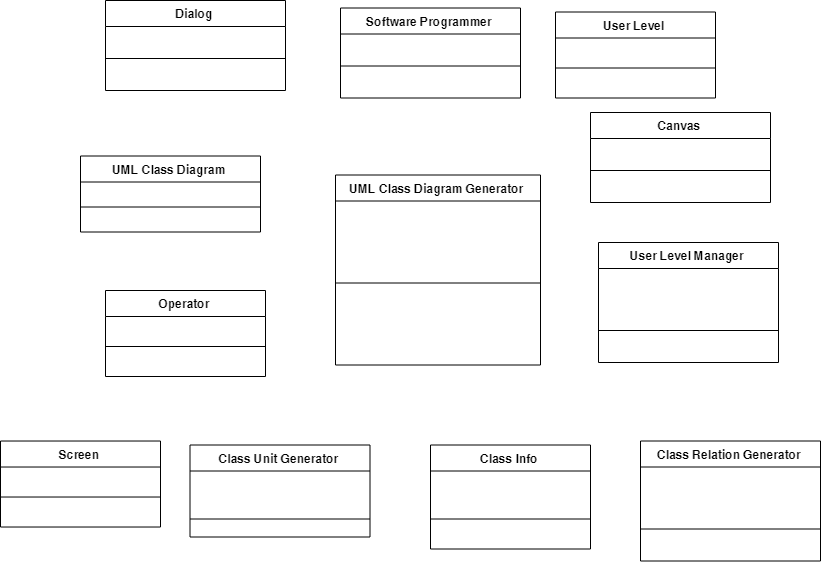
### 2.1.2 Bad Classes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vague** | **Attributes** | **Operations** | **Implementation Construction** | **Roles** |
| Crash | Error Message | Read Field Value | Class Unit | Recorder |
|  | Class  Attribute | Output | Class Relation Line | Field Manager |
|  | Backup Path | Restart | Function List | User Manager |
|  | Field Value | Refresh |  | Output Manager |
|  | Level Id | Generate  Diagram |  |  |
|  |  | Interrupt  Operation |  |  |
|  |  | Backup  File |  |  |
|  |  | Record |  |  |
|  |  | Automation  Connect  Class  Relation  Line |  |  |
|  |  | Manual  Connect  Class  Relation  Line |  |  |

* **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

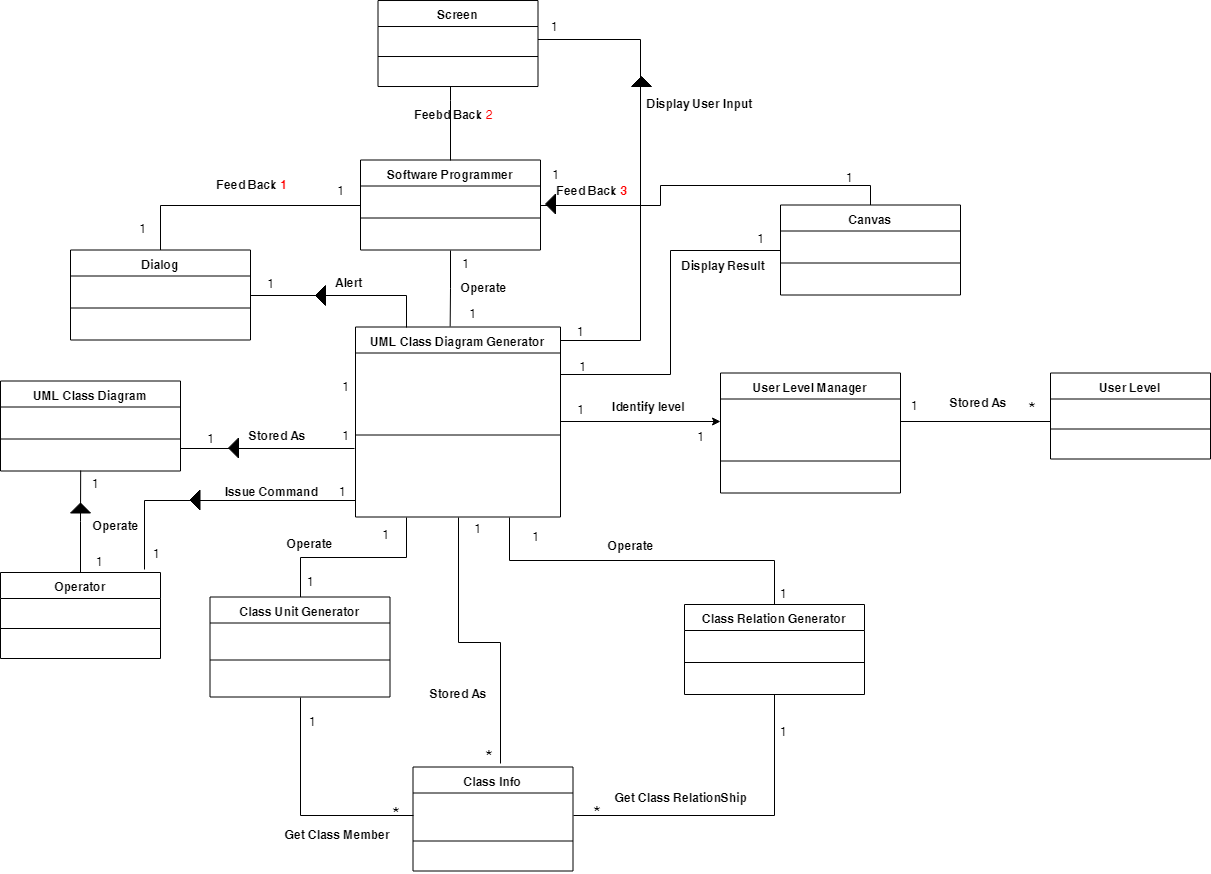
### 2.1.3 Good Classes

|  |  |  |  |
| --- | --- | --- | --- |
| UML Class Diagram Generator | User Level | Screen | Dialog |
| UML Class Diagram | Class Info | Canvas |  |
| Class Unit Generator |  |  |  |
| Class Relation Generator |  |  |  |
| Software Programmer |  |  |  |
| Operator |  |  |  |
| User Level Manager |  |  |  |



## 2.2 Add Associations

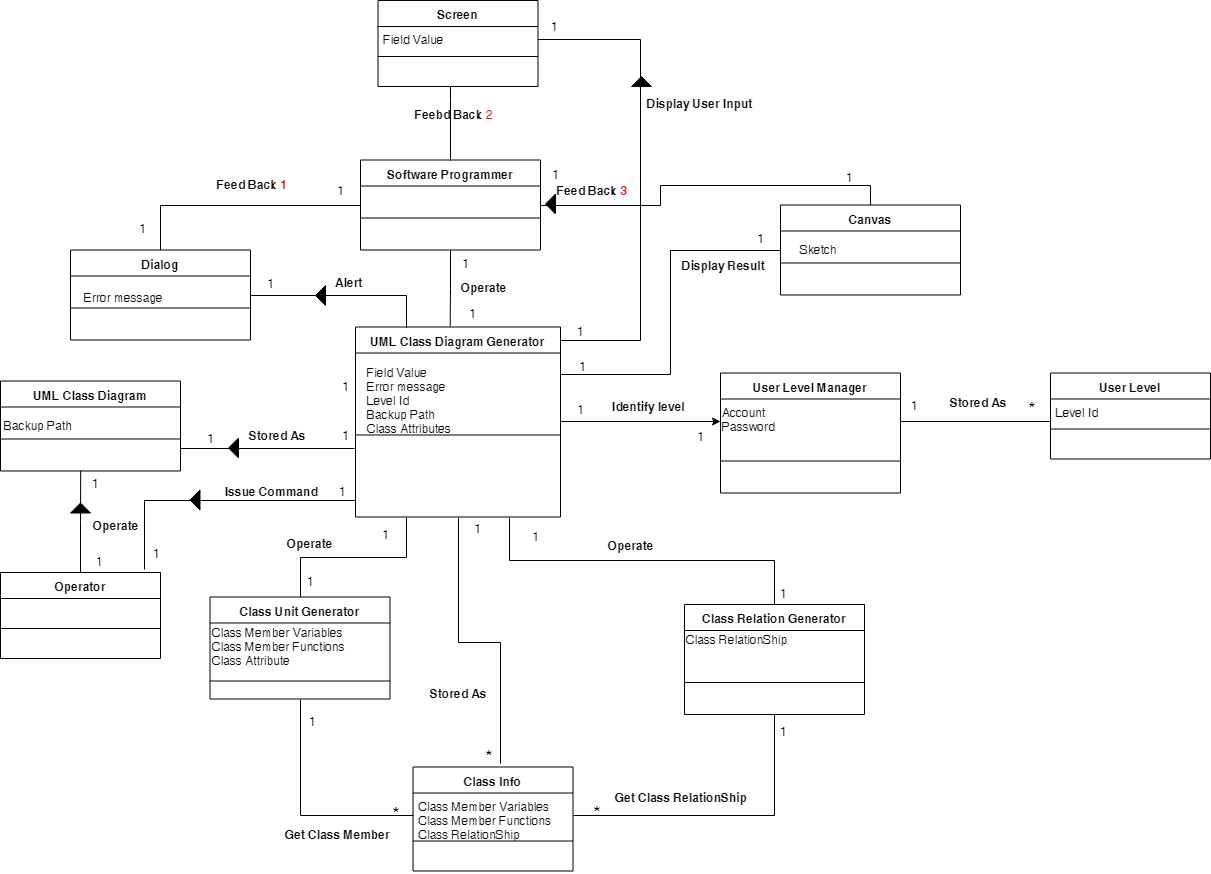
(修改)



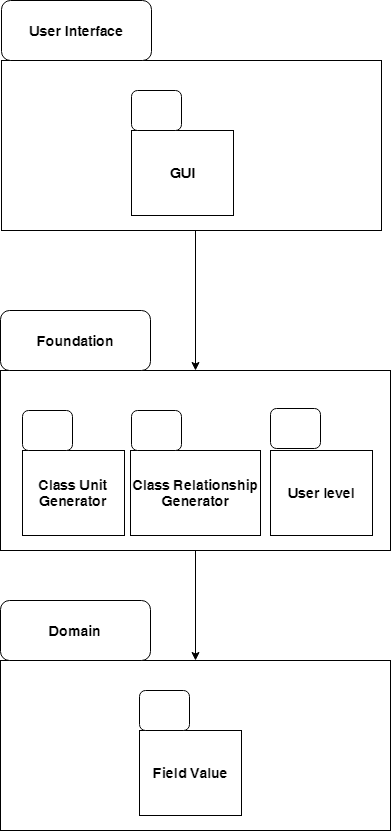
|  |  |
| --- | --- |
| Issue Command | UML Class Diagram Generator 對 Operator下操作指令 |
| Operate | 1.Software Programmer使用UML Class Diagram Generator。  2.Operator對UML Class Diagram進行CRUD操作 |
| Feed Back 1 | Software Programmer得到UML Class Diagram Generator產生之互動提醒。 |
| Feed Back 2 | Software Programmer得到UML Class Diagram Generator產生之Class Info回饋。 |
| Feed Back 3 | Software Programmer得到UML Class Diagram Generator產生之sketch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。 |
| Display User Input | UML Class Diagram Generato 顯示 Software Programmer 之輸入值。 |
| Stored As  (三個部分) | 1.Class Info儲存UML Class Diagram Generator讀入之各欄位值。  2.UML Class Diagram 儲存欲輸出之UML Class Diagram。  3.User Level 儲存 User Level Manager辨別之後的結果。 |
| 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

(修改)



# 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

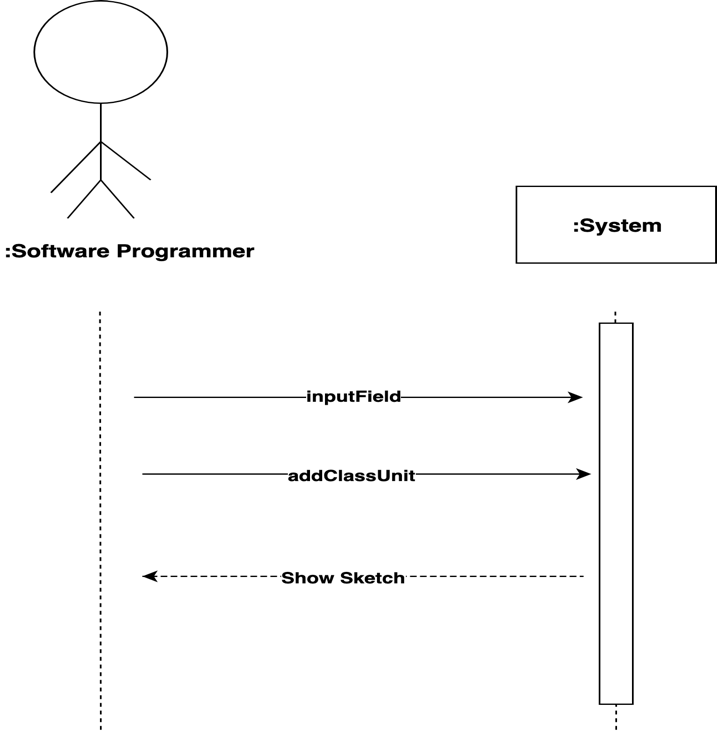
**Field Value:** 使用者輸入值相關類別

# 4. Use-Case Realizations with GRASP Patterns

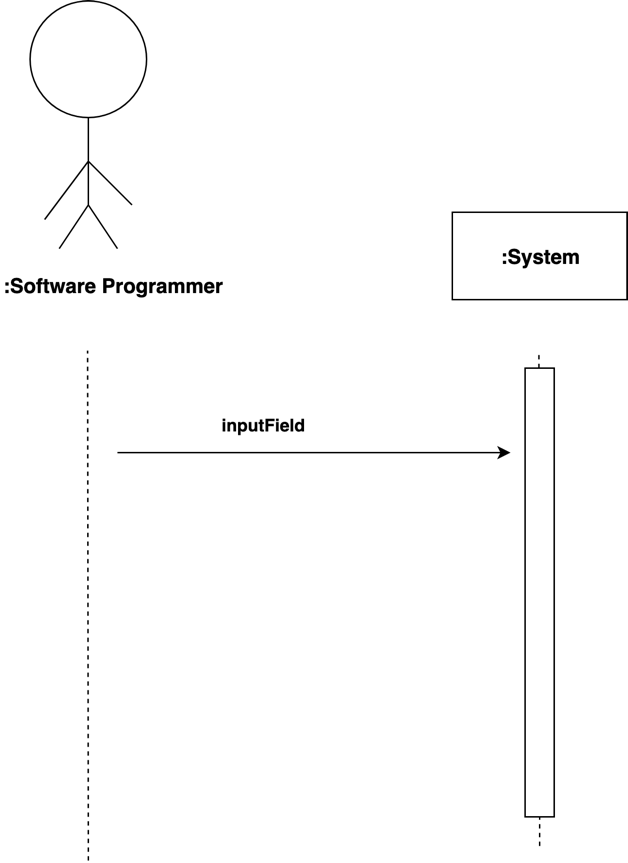
(全數修改)

## 4.1 System Sequence Diagram

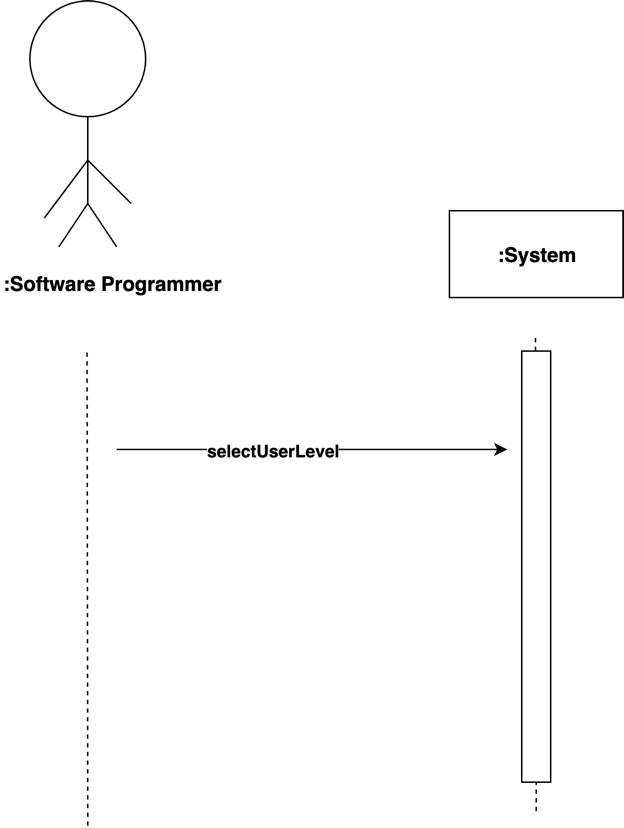
**新增空白UML Class Diagram元件**



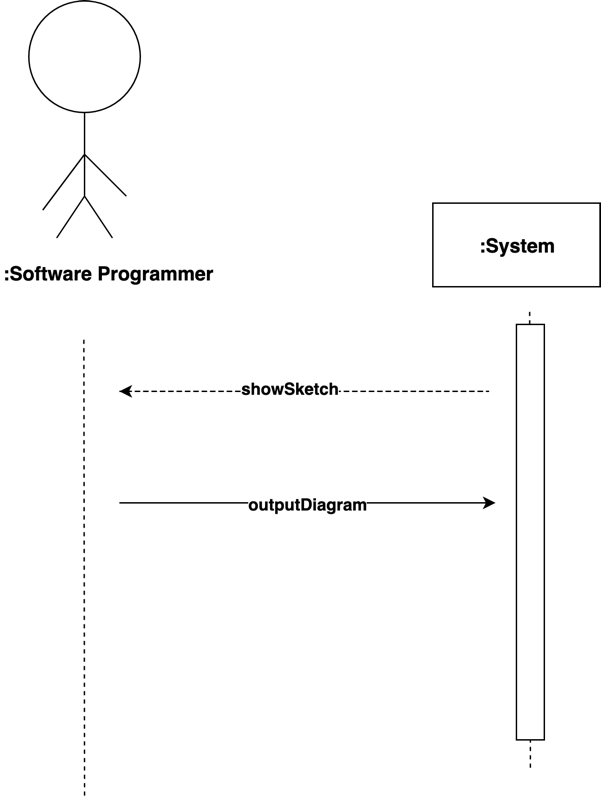
**讀取User輸入之欄位資料並寫入表格**



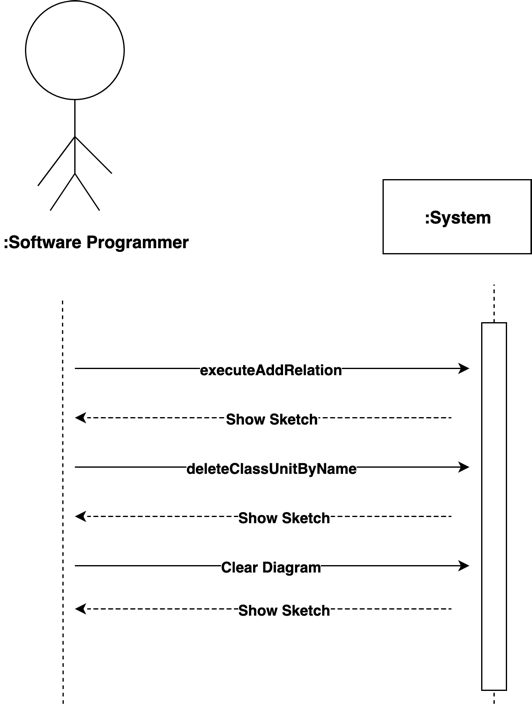
**建立User Level,提供不同Level不同服務**



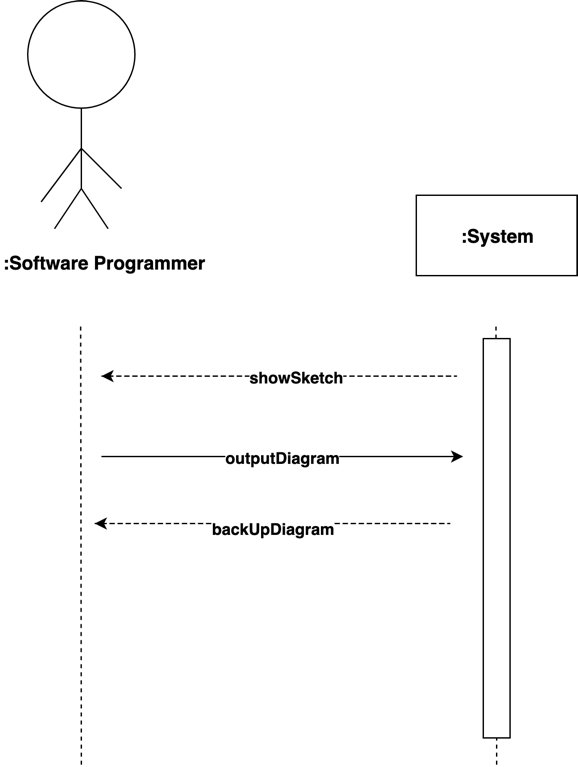
**輸出User的UML Class Diagram透過多種方式**

****

**修改UML Class Diagram透過手動操作**



**備份輸出的檔案**

****

## 4.2 Operation Contract

|  |  |
| --- | --- |
| **Contract ID** | **Operation Name** |
| **Contract01** | selectUserLevel |
| **Contract02** | inputField |
| **Contract03** | showSketch |
| **Contract04** | backUpDiagram |
| **Contract05** | executeAddRelation |
| **Contract06** | outputDiagram |
| **Contract07** | deleteClassUnitByName |
| **Contract08** | clearDiagram |
| **Contract09** | addClassUnit |

### 4.2.1 selectUserLevel

|  |  |
| --- | --- |
| **Operation** | selectUserLevel(level:string) |
| **Cross References** | Use Case : UC-03 建立User Level,提供不同Level不同服務 |
| **Preconditions** | 應用程式成功開啟。 |
| **Postconditions** | 顯示該User Level可操作之功能。 |

### 4.2.2 inputField

|  |  |
| --- | --- |
| **Operation** | inputField |
| **Cross References** | Use Case :UC-02 **讀取User輸入之欄位資料**  System Event :  **“Issue Read Command”** |
| **Preconditions** | 應用程式在畫面上提供可輸入的欄位 |
| **Postconditions** | 正確讀取欄位資料並寫入表格。 |

### 4.2.3 showSketch

|  |  |
| --- | --- |
| **Operation** | showSketch |
| **Cross References** | Use Case :UC-01新增空白UML Class Diagram元件  System Event :  **“Show UML Class Diagram Sketch”**  Use Case :UC-02 讀取User輸入之欄位資料  System Event :  **“Show UML Class Diagram Sketch”**  Use Case : UC-05修改UML Class Diagram透過手動操作  System Event :  **“Show UML Class Diagram Sketch”** |
| **Preconditions** | 能成功讀取input欄位 |
| **Postconditions** | 預覽輸入欄位轉換成UML的結果 |

### 4.2.4 backUpDiagram

|  |  |
| --- | --- |
| **Operation** | backUpDiagram |
| **Cross References** | Use Case : UC-06**備份輸出的檔案** |
| **Preconditions** | Software Programmer按下Save Button |
| **Postconditions** | 應用程式成功將UML Class Diagram備份至指定路徑 |

### 4.2.5 executeAddRelation

|  |  |
| --- | --- |
| **Operation** | executeAddRelation |
| **Cross References** | Use Case : UC-05修改UML Class Diagram透過手動操作  System Event :  “Manually Modified” |
| **Preconditions** | 草圖有被建立成功 |
| **Postconditions** | 可以顯示UML Class 之間的關係 |

### 4.2.6 outputDiagram

|  |  |
| --- | --- |
| **Operation** | outputDiagram |
| **Cross References** | Use Case : UC-04輸出User的UML Class Diagram透過多種方式 |
| **Preconditions** | 應用程式的輸出按鈕可以正確執行 |
| **Postconditions** | 應用程式成功輸出Software Programmer所選擇的格式 |

### 4.2.7 deleteClassUnitByName

|  |  |
| --- | --- |
| **Operation** | deleteClassUnitByName |
| **Cross References** | Use Case : UC-05修改UML Class Diagram透過手動操作  System Event :  “Manually Modified” |
| **Preconditions** | 程式可以成功讀取輸入欄位 |
| **Postconditions** | UML 圖可以刪除指定class |

### 4.2.8 clearDiagram

|  |  |
| --- | --- |
| **Operation** | clearDiagram |
| **Cross References** | Use Case : UC-01新增空白UML Class Diagram元件 |
| **Preconditions** | 應用程式的清除按鈕可以正確執行 |
| **Postconditions** | 應用程式成功將畫面上顯示的UML Class Diagram清除 |

### 4.2.9 addClassUnit

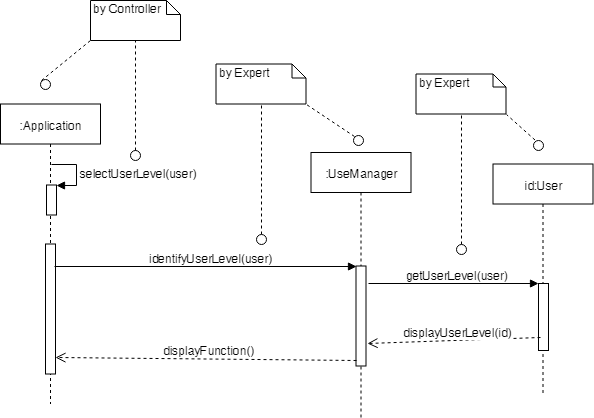
|  |  |
| --- | --- |
| **Operation** | addClassUnit |
| **Cross References** | Use Case : UC-01新增空白UML Class Diagram  元件 |
| **Preconditions** | 應用程式的新增按鈕可以正常執行 |
| **Postconditions** | 應用程式成功將Class Unit新增至畫面上 |

## 4.3 Operation Sequence Diagram

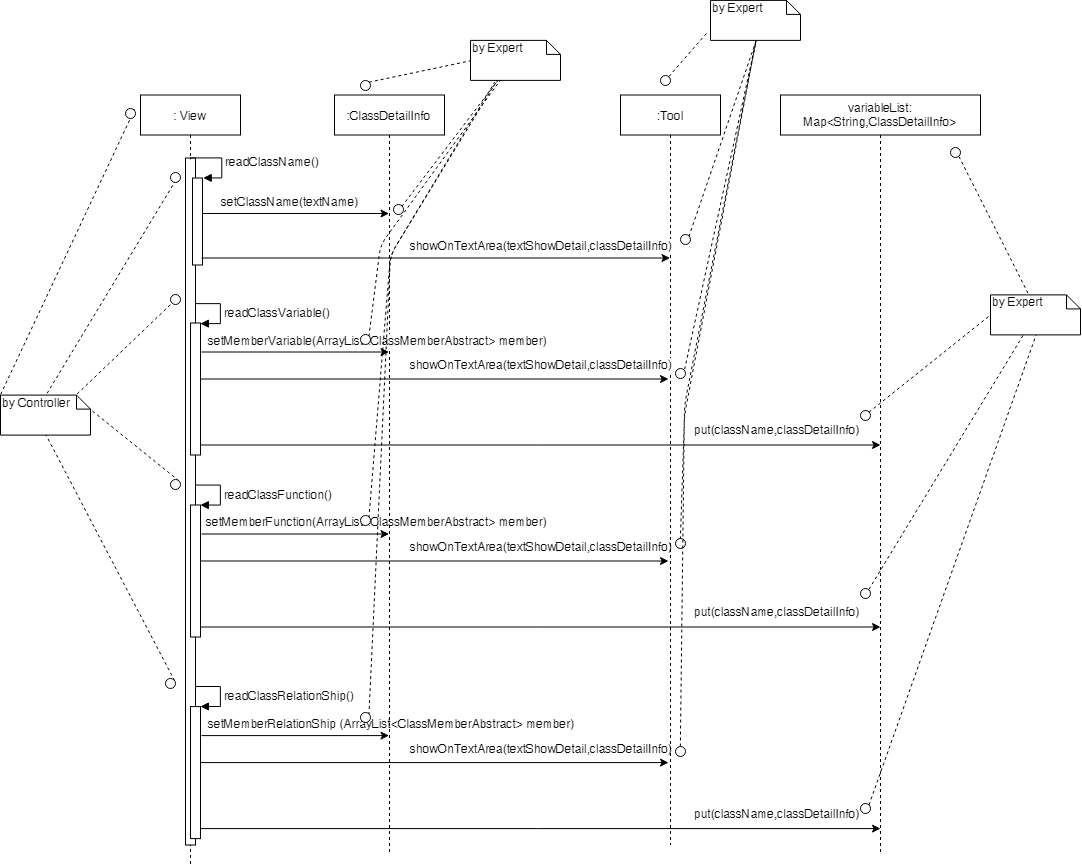
* 以下Sequence Diagram對應之System Event

已標示在Operation Contract中的Cross References

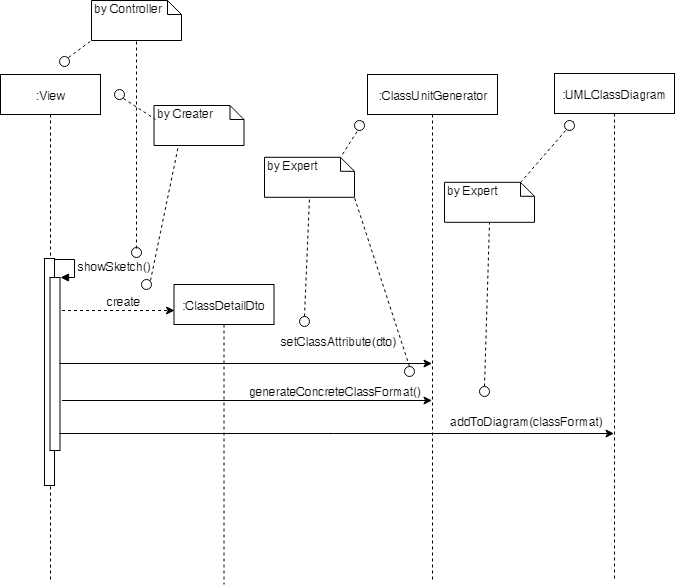
### 4.3.1 selectUserLevel



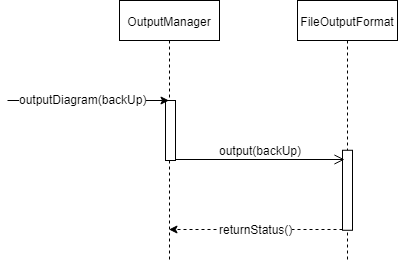
### 4.3.2 inputField

****

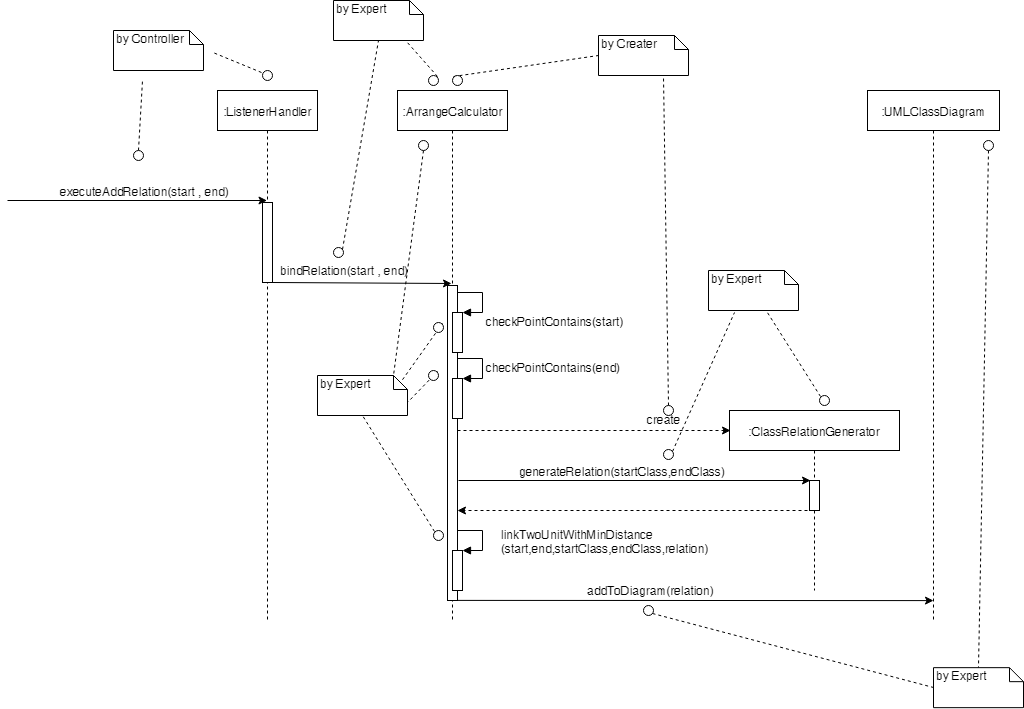
### 4.3.3 showSketch



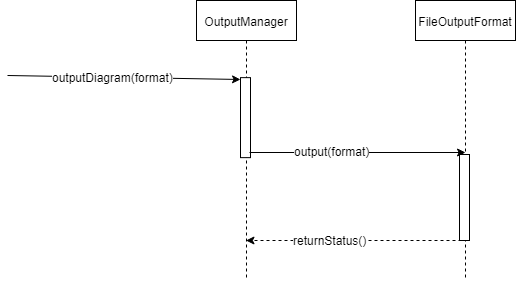
### 4.3.4 backUpDiagram

****

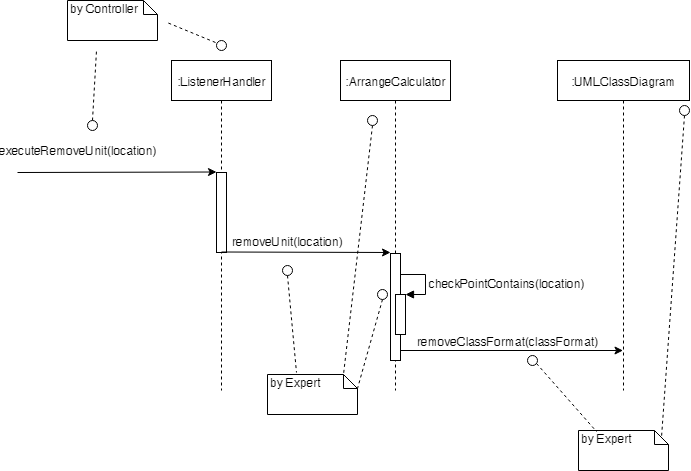
### 4.3.5 executeAddRelation



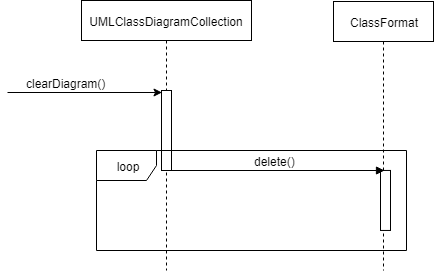
### 4.3.6 outputDiagram



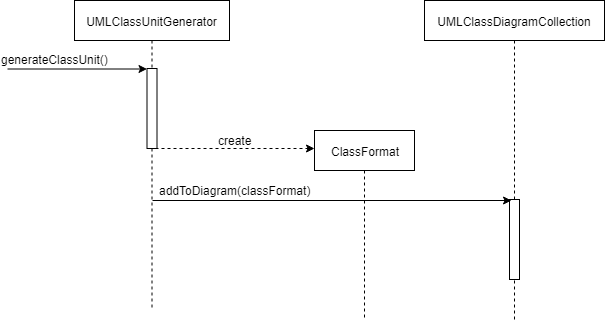
### 4.3.7 deleteClassUnitByName



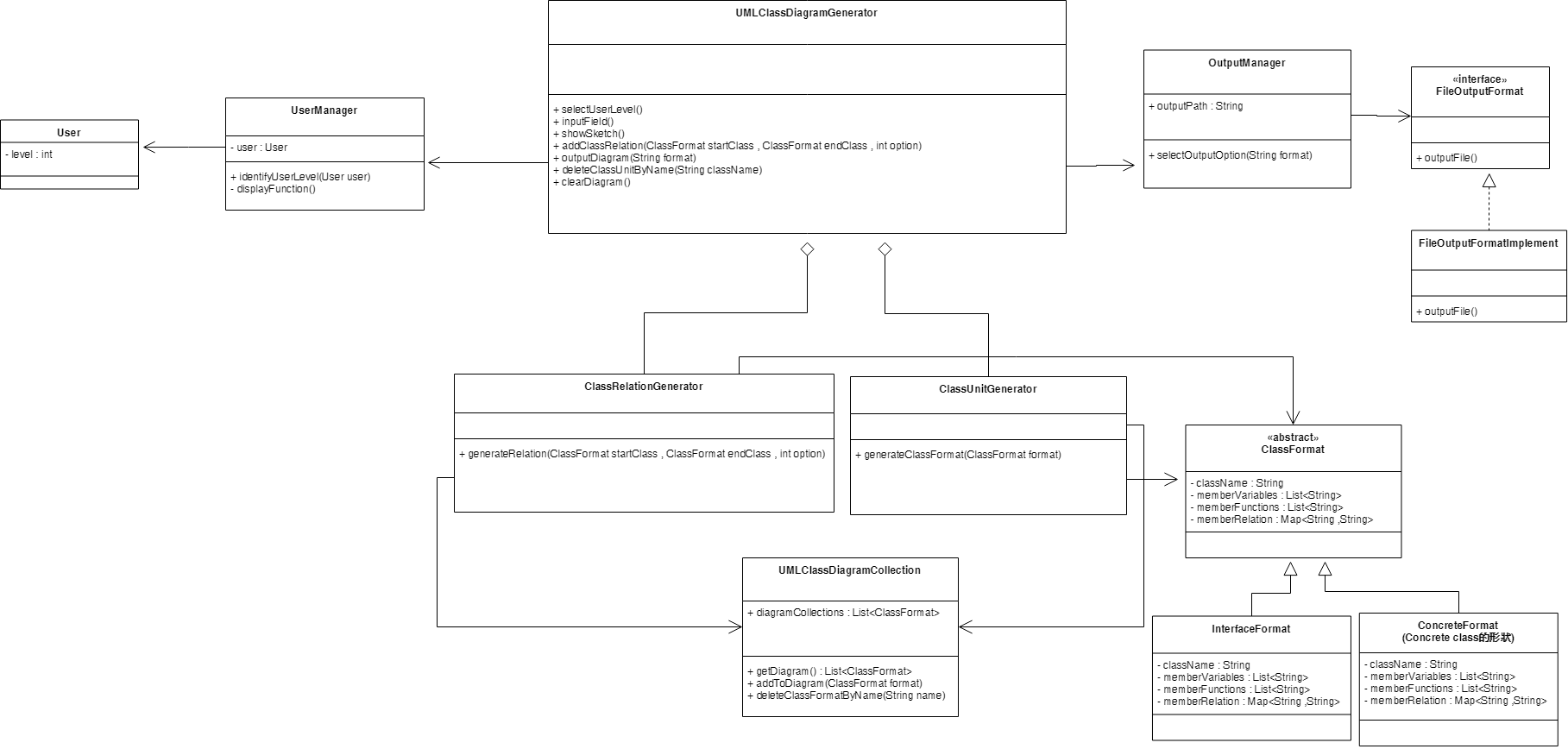
### 4.3.8 clearDiagram

****

### 4.3.9 addClassUnit

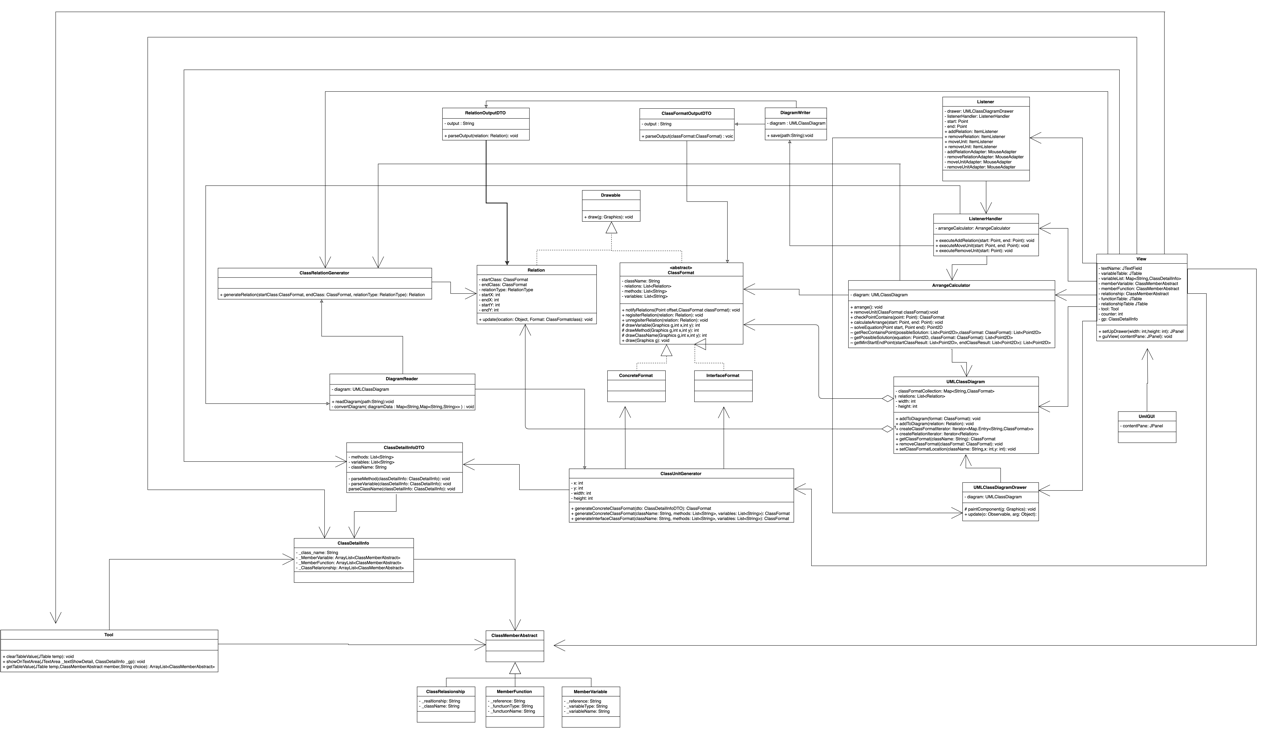
****

## 4.4Design Class Model



# 5 Implementation Class Model

# 5.1 Draw an implementation class diagram for your system (including associations, attributes and methods) (new)

****

# 5.2 Show the difference between implementation class model and design class model (new)

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Method** | **Design** | **Implement** |
| View | setUpDrawer() | NO | YES |
| guiView() | NO | YES |
| UMLClassDiagram | getDiagram() | YES | NO |
| addToDiagram() | YES | Yes |
| deleteClassFormByName() | YES | NO |
| createClassFormatIterator() | NO | YES |
| createRelationIterator() | NO | YES |
| getClassFormat() | NO | YES |
| removeClassFormat() | NO | YES |
| ClassUnitGenerator | generateConcreteClassFormat() | YES | YES |
| generateInterfaceClassFormat() | NO | YES |
| UMLClassDiagramDrawer | paintComponent() | NO | YES |
| Update() | NO | YES |
| ArrangeCalculator | Arrange() | NO | YES |
| removeUnit() | NO | YES |
| checkPointContains() | NO | YES |
| calculateArrange() | NO | YES |
| solveEquation() | NO | YES |
| getRecContainsPoint() | NO | YES |
| getPossibleSolution() | NO | YES |
| getMinStartEndPoint() | NO | YES |
| ListenerHandler | executeAddRelation() | NO | YES |
| executeMoveUnit() | NO | YES |
| executeRemoveUnit() | NO | YES |
| ClassRelationGenerator | generateRelation() | YES | YES |
| Drawable | draw() | NO | YES |
| ClassFormat | notifyRelations() | NO | YES |
| regisiterRelation() | NO | YES |
| unregisiterRelation() | NO | YES |
| drawVariable() | NO | YES |
| drawMethod() | NO | YES |
| drawClassName() | NO | YES |
| draw() | NO | YES |
| Relation | Update() | NO | YES |
| Tool | clearTableValue() | NO | YES |
| showOnTextArea() | NO | YES |
| getTableValue() | NO | YES |
| UerManager | identifyUserLevel() | YES | NO |
| displayFunction() | YES | NO |
| OutputManager | selectOption() | YES | NO |
| FileOutputFormatlmplement | outputFole() | YES | NO |
| ClassFormatOutputDTO | parseOutput() | NO | YES |
| getOutput() | NO | YES |
| DiagramReader | readDiagram() | NO | YES |
| convertDiagram() | NO | YES |
| getDiagram() | NO | YES |
| DiagramWriter | Save() | NO | YES |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Number of added | Number of removed | Number of modified |
| Class | **11** | **4** | **6** |
| Method | **38** | **4** | **6** |

# 

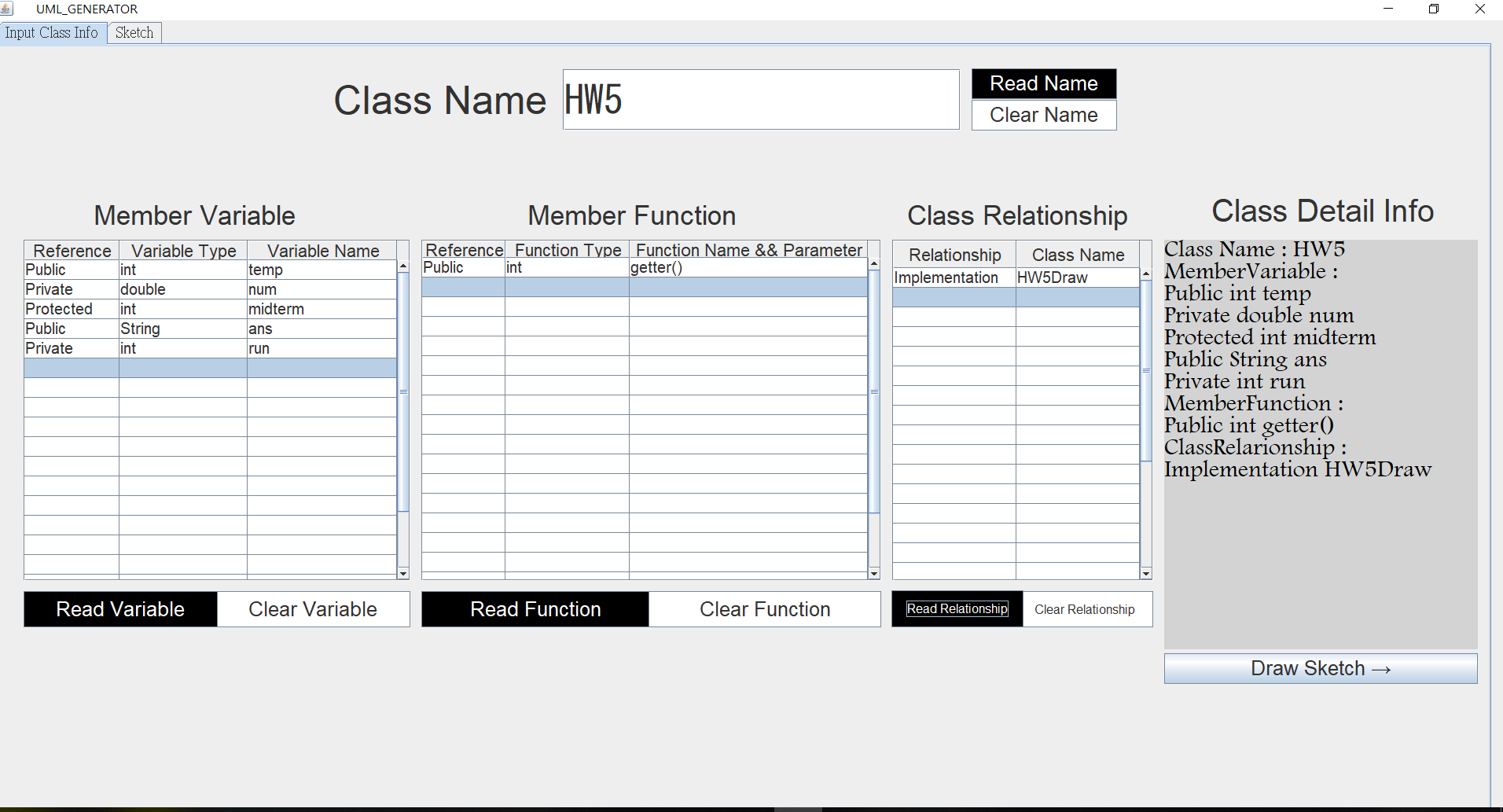
# 5.3 Calculate Line of Code (new)

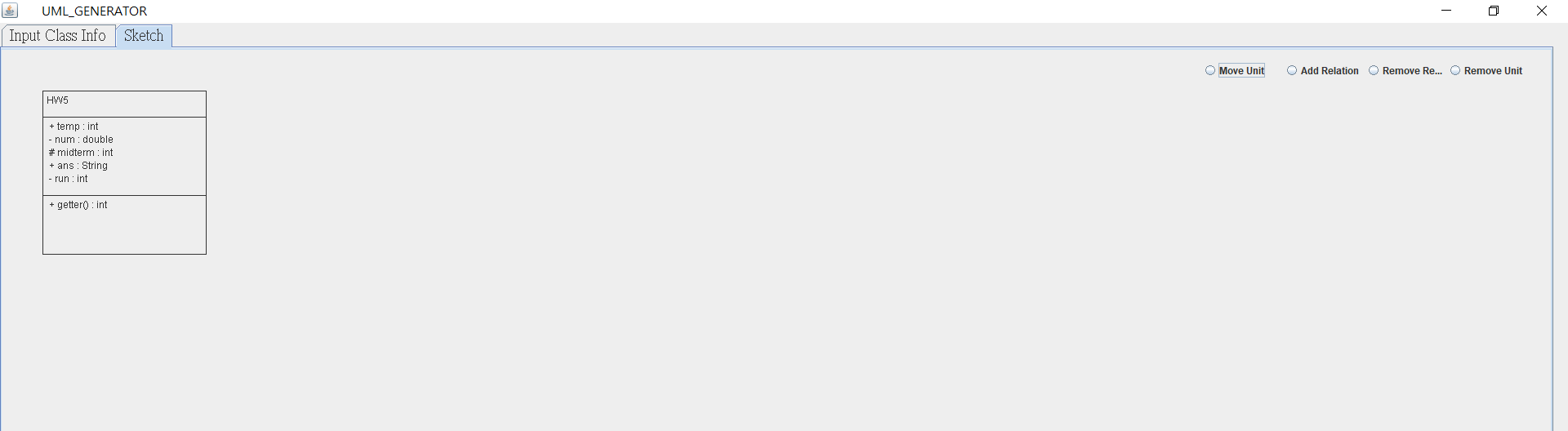
|  |  |  |  |
| --- | --- | --- | --- |
| NO | Class Name | Number of Methods | Line of Code in Class (without comment) |
| 1 | ClassDetailInfoDTO | 9 | 63 |
| 2 | ClassDetailInfo | 14 | 52 |
| 3 | classMemberAbstract | 7 | 19 |
| 4 | ClassRelationship | 7 | 32 |
| 5 | MemberFunction | 9 | 42 |
| 6 | MemberVariable | 9 | 42 |
| 7 | Drawable | 1 | 5 |
| 8 | UMLClassDiagram | 14 | 68 |
| 9 | UMLClassDiagramDrawer | 3 | 37 |
| 10 | ArrangeCalculator | 10 | 139 |
| 11 | ClassRelationGenerator | 2 | 12 |
| 12 | ClassUnitGenerator | 4 | 33 |
| 13 | Listener | 1 | 94 |
| 14 | ListenerHandler | 4 | 25 |
| 15 | ClassFormat | 16 | 99 |
| 16 | ConcreteFormat | 1 | 3 |
| 17 | InterfaceFormat | 3 | 20 |
| 18 | Relation | 12 | 68 |
| 19 | View | 3 | 451 |
| 20 | Tool | 4 | 78 |
| 21 | ClassFormatOutputDTO | 2 | 43 |
| 22 | DiagramReader | 3 | 91 |
| 23 | DiagramWriter | 1 | 48 |
| SUM | | 133 | 1564 |

***Line of Code of classes***

# 6 Programming

# 6.1 Snapshots of system execution





# 6.2 Source Code Listing

### 6.2.1UmlGUI

package uml\_generator;

import java.awt.EventQueue;

import java.awt.Font;

import javax.swing.DefaultCellEditor;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTabbedPane;

import javax.swing.JTable;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.ListSelectionModel;

import javax.swing.border.EmptyBorder;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.JTableHeader;

import javax.swing.table.TableColumn;

import ClassDetailInfo.ClassDetailInfo;

import ClassDetailInfo.ClassMemberAbstract;

import java.awt.Dimension;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.awt.Color;

import java.awt.Component;

public class UmlGUI extends JFrame {

private JPanel contentPane;

private ClassDetailInfo gp = new ClassDetailInfo();

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

UmlGUI frame = new UmlGUI();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public UmlGUI() {

setExtendedState(JFrame.MAXIMIZED\_BOTH);

setType(Type.POPUP);

setTitle(" \t\tUML\_GENERATOR");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

Dimension screensize = Toolkit.getDefaultToolkit().getScreenSize();

int height = (int)screensize.getHeight() ;

int width = (int)screensize.getWidth() ;

setBounds(0, 0, width, height);

contentPane = new JPanel();

contentPane.setPreferredSize(screensize);

contentPane.setAlignmentY(Component.TOP\_ALIGNMENT);

contentPane.setAlignmentX(Component.LEFT\_ALIGNMENT);

contentPane.setToolTipText("");

contentPane.setBorder(new EmptyBorder(10, 10, 10, 10));

setContentPane(contentPane);

contentPane.setLayout(null);

View view = new View();

view.guiView(contentPane);

}

}

### 6.2.2View

package uml\_generator;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.Map;

import java.util.TreeMap;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.JTableHeader;

import javax.swing.table.TableColumn;

import ClassDetailInfo.\*;

import adapter.ClassDetailInfoDTO;

import diagrams.UMLClassDiagram;

import diagrams.UMLClassDiagramDrawer;

import generator.ArrangeCalculator;

import generator.ClassUnitGenerator;

import listeners.Listener;

import listeners.ListenerHandler;

import shapes.ClassFormat;

import viewTool.Tool;

public class View {

private JTextField textName;

private JTable variableTable;

private Map<String,ClassDetailInfo> variableList = new TreeMap<>();

private ClassMemberAbstract memberVariable = new MemberVariable();

private ClassMemberAbstract memberFunction = new MemberFunction();

private ClassMemberAbstract relationship = new ClassRelationship();

private JTable functionTable;

private JTable relationshipTable;

private Tool tool = new Tool();

private int counter = 1;

private ClassDetailInfo classDetailInfo = new ClassDetailInfo();

UMLClassDiagram diagram;

ClassUnitGenerator unitGenerator;

UMLClassDiagramDrawer drawer;

ArrangeCalculator arrangeCalculator;

ListenerHandler listenerHandler;

Listener listener;

public JPanel setUpDrawer(int width,int height){

int x=width-50;

diagram=new UMLClassDiagram(width,height);

unitGenerator=new ClassUnitGenerator();

drawer=new UMLClassDiagramDrawer(diagram);

arrangeCalculator =new ArrangeCalculator(diagram);

arrangeCalculator.arrange();

listenerHandler=new ListenerHandler(diagram);

listener=new Listener(drawer,listenerHandler);

ButtonGroup group=new ButtonGroup();

JRadioButton moveUnit=new JRadioButton("Move Unit");

JRadioButton addRelation=new JRadioButton("Add Relation");

JRadioButton removeRelation=new JRadioButton("Remove Relation");

JRadioButton removeUnit=new JRadioButton("Remove Unit");

JButton outputButton=new JButton("Output");

JButton saveButton=new JButton("Save");

JButton openFileButton=new JButton("Open File");

openFileButton.addActionListener(listener.openFileListener);

outputButton.addActionListener(listener.outputListener);

saveButton.addActionListener(listener.saveListener);

moveUnit.addItemListener(listener.moveUnit);

addRelation.addItemListener(listener.addRelation);

removeRelation.addItemListener(listener.removeRelation);

removeUnit.addItemListener(listener.removeUnit);

openFileButton.setBounds(x-300,50,75,30);

outputButton.setBounds(x-200,50,75,30);

saveButton.setBounds(x-100,50,75,30);

moveUnit.setBounds(x-450,0,100,50);

addRelation.setBounds(x-350,0,100,50);

removeRelation.setBounds(x-250,0,150,50);

removeUnit.setBounds(x-100,0,100,50);

group.add(moveUnit);

group.add(addRelation);

group.add(removeRelation);

group.add(removeUnit);

drawer.add(openFileButton);

drawer.add(outputButton);

drawer.add(saveButton);

drawer.add(moveUnit);

drawer.add(addRelation);

drawer.add(removeRelation);

drawer.add(removeUnit);

drawer.setVisible(true);

return drawer;

}

public void guiView( JPanel contentPane) {

Toolkit tk = Toolkit.getDefaultToolkit();

Dimension d = tk.getScreenSize();

int width=(int)d.getWidth(),height=(int)d.getHeight();

JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);

tabbedPane.setFont(new Font("新細明體", Font.PLAIN, 20));

tabbedPane.setBounds(0, 0, 1902, 1033);

contentPane.add(tabbedPane);

JPanel Input = new JPanel();

tabbedPane.addTab("Input Class Info", null, Input, null);

Input.setLayout(null);

JPanel drawer = setUpDrawer(width,height); // 這邊要把你的panel替換過來!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

// JPanel drawer=new JPanel();

drawer.setBounds(0,0,width,height);

// drawer.add(label);

tabbedPane.addTab("Sketch", null, drawer, null);

drawer.setLayout(null);

// Label

JLabel className = new JLabel("Class Name");

className.setBounds(427, 39, 272, 59);

className.setFont(new Font("Arial", Font.PLAIN, 50));

Input.add(className);

// Label

// 輸入Class Name的textarea

textName = new JTextField();

textName.setBounds(719, 29, 506, 78);

textName.setFont(new Font("MS Gothic", Font.PLAIN, 50));

Input.add(textName);

textName.setColumns(20);

// 顯示框(textArea)

JTextArea textShowDetail = new JTextArea();

textShowDetail.setEditable(false);

textShowDetail.setBounds(1484, 246, 399, 521);

textShowDetail.setBackground(new Color(211, 211, 211));

textShowDetail.setForeground(Color.BLACK);

textShowDetail.setFont(new Font("Footlight MT Light", Font.PLAIN, 30));

Input.add(textShowDetail);

JLabel lblNewLabel = new JLabel("Member Variable");

lblNewLabel.setBounds(122, 194, 257, 41);

lblNewLabel.setFont(new Font("Arial", Font.PLAIN, 35));

Input.add(lblNewLabel);

JLabel lblClassDetailInfo = new JLabel("Class Detail Info");

lblClassDetailInfo.setBounds(1544, 186, 284, 47);

lblClassDetailInfo.setFont(new Font("Arial", Font.PLAIN, 40));

Input.add(lblClassDetailInfo);

JScrollPane scrollPaneVariable = new JScrollPane();

scrollPaneVariable.setBounds(33, 246, 492, 434);

Input.add(scrollPaneVariable);

variableTable = new JTable();

JTableHeader headVariable = variableTable.getTableHeader(); // 创建表格标题对象

headVariable.setPreferredSize(new Dimension(headVariable.getWidth(), 25));// 设置表头大小

headVariable.setFont(new Font("楷体", Font.PLAIN, 22));// 设置表格字体

variableTable.setSelectionMode(ListSelectionModel.MULTIPLE\_INTERVAL\_SELECTION);

variableTable.setToolTipText("");

variableTable.setFont(new Font("Arial", Font.PLAIN, 20));

variableTable.setRowHeight(25);

variableTable.setModel(new DefaultTableModel(

new Object[][] {

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

},

new String[] {

"Reference", "Variable Type", "Variable Name"

}

) {

Class[] columnTypes = new Class[] {

String.class, String.class, String.class

};

public Class getColumnClass(int columnIndex) {

return columnTypes[columnIndex];

}

});

variableTable.getColumnModel().getColumn(0).setPreferredWidth(124);

variableTable.getColumnModel().getColumn(1).setPreferredWidth(168);

variableTable.getColumnModel().getColumn(2).setPreferredWidth(196);

// 把ComboBox加入tabel1

TableColumn referenceColumn = variableTable.getColumn("Reference");

scrollPaneVariable.setViewportView(variableTable);

JLabel lblMemberFunction = new JLabel("Member Function");

lblMemberFunction.setBounds(674, 194, 266, 41);

lblMemberFunction.setFont(new Font("Arial", Font.PLAIN, 35));

Input.add(lblMemberFunction);

JScrollPane scrollPaneFunction = new JScrollPane();

scrollPaneFunction.setBounds(539, 246, 585, 434);

Input.add(scrollPaneFunction);

functionTable = new JTable();

JTableHeader headFunction= functionTable.getTableHeader(); // 创建表格标题对象

headFunction.setPreferredSize(new Dimension(headFunction.getWidth(), 22));// 设置表头大小

headFunction.setFont(new Font("楷体", Font.PLAIN, 22));// 设置表格字体

functionTable.setFont(new Font("Arial", Font.PLAIN, 20));

functionTable.setModel(new DefaultTableModel(

new Object[][] {

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null},

},

new String[] {

"Reference", "Function Type", "Function Name && Parameter"

}

) {

Class[] columnTypes = new Class[] {

String.class, Object.class, Object.class

};

public Class getColumnClass(int columnIndex) {

return columnTypes[columnIndex];

}

});

functionTable.getColumnModel().getColumn(0).setPreferredWidth(135);

functionTable.getColumnModel().getColumn(1).setPreferredWidth(204);

functionTable.getColumnModel().getColumn(2).setPreferredWidth(395);

functionTable.setRowHeight(25);

scrollPaneFunction.setViewportView(functionTable);

TableColumn referenceColumn\_function = functionTable.getColumn("Reference");

JLabel labelRelation = new JLabel("Class Relationship");

labelRelation.setFont(new Font("Arial", Font.PLAIN, 35));

labelRelation.setBounds(1157, 194, 281, 41);

Input.add(labelRelation);

JScrollPane scrollPaneRelationship = new JScrollPane();

scrollPaneRelationship.setBounds(1138, 246, 332, 434);

Input.add(scrollPaneRelationship);

// Relationship\_table

relationshipTable = new JTable();

JTableHeader headRelationship= relationshipTable.getTableHeader(); // 创建表格标题对象

headRelationship.setPreferredSize(new Dimension(headRelationship.getWidth(), 35));// 设置表头大小

headRelationship.setFont(new Font("楷体", Font.PLAIN, 22));// 设置表格字体

relationshipTable.setFont(new Font("Arial", Font.PLAIN, 20));

relationshipTable.setModel(new DefaultTableModel(

new Object[][] {

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

{null, null},

},

new String[] {

"Relationship", "Class Name"

}

) {

Class[] columnTypes = new Class[] {

String.class, String.class

};

public Class getColumnClass(int columnIndex) {

return columnTypes[columnIndex];

}

});

TableColumn referenceColumn\_relationship = relationshipTable.getColumn("Relationship");

relationshipTable.setRowHeight(25);

scrollPaneRelationship.setViewportView(relationshipTable);

// Read Name Button

JButton btnReadName = new JButton("Read Name");

btnReadName.setForeground(Color.WHITE);

btnReadName.setBounds(1239, 28, 185, 39);

btnReadName.setBackground(new Color(240, 240, 240));

btnReadName.setFont(new Font("Arial", Font.PLAIN, 26));

btnReadName.setBackground(Color.BLACK);

btnReadName.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

if ( textName.getText() != null && !textName.getText().equals(""))

classDetailInfo.setClassName(textName.getText());

else

classDetailInfo.setClassName("");

tool.showOnTextArea(textShowDetail, classDetailInfo);

}

});

Input.add(btnReadName);

JButton btnReadVariable = new JButton("Read Variable");

btnReadVariable.setForeground(Color.WHITE);

btnReadVariable.setBounds(33, 693, 247, 46);

btnReadVariable.setFont(new Font("Arial", Font.PLAIN, 26));

btnReadVariable.setBackground(Color.BLACK);

Input.add(btnReadVariable);

// Read Variable Button

btnReadVariable.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

ArrayList<ClassMemberAbstract> member = tool.getTableValue(variableTable,memberVariable,"Variable");

classDetailInfo.clearMemberVariable();

for ( int i=0; i < member.size();i++)

{

classDetailInfo.setMemberVariable(member.get(i));

String ans = member.get(i).getReference()+ " " + member.get(i).getType() + " " + member.get(i).getName() ;

textShowDetail.append(ans);

}

variableList.put(classDetailInfo.getClassName(), classDetailInfo);

tool.showOnTextArea(textShowDetail, classDetailInfo);

}

});

JButton btnReadRelationship = new JButton("Read Relationship");

btnReadRelationship.setForeground(Color.WHITE);

btnReadRelationship.setBackground(Color.BLACK);

//Read Relationship Button

btnReadRelationship.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

ArrayList<ClassMemberAbstract> member = tool.getTableValue(relationshipTable,relationship,"Relation");

classDetailInfo.clearClassRelarionship();

for ( int i=0; i < member.size();i++)

{

classDetailInfo.setClassRelarionship(member.get(i));

String ans = member.get(i).getReference()+ " " + member.get(i).getName() ;

textShowDetail.append(ans);

}

variableList.put(classDetailInfo.getClassName(), classDetailInfo);

tool.showOnTextArea(textShowDetail, classDetailInfo);

}

});

btnReadRelationship.setFont(new Font("Arial", Font.PLAIN, 16));

btnReadRelationship.setBounds(1137, 692, 168, 47);

Input.add(btnReadRelationship);

JButton btnShowSketch = new JButton("Show Sketch \u2192");

// Draw Sketch Button

btnShowSketch.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

textName.setText("");

tool.clearTableValue(variableTable);

tool.clearTableValue(functionTable);

tool.clearTableValue(relationshipTable);

textShowDetail.setText("");

tabbedPane.setSelectedComponent(drawer);

if (classDetailInfo.getClassName().equals("")){

classDetailInfo.setClassName("NewClass"+counter);

counter++;

}

ClassDetailInfoDTO dto=new ClassDetailInfoDTO(classDetailInfo);

unitGenerator.setClassAttributes(dto);

ClassFormat classFormat=unitGenerator.generateConcreteClassFormat();

diagram.addToDiagram(classFormat);

classDetailInfo = new ClassDetailInfo();

}

});

btnShowSketch.setFont(new Font("Arial", Font.PLAIN, 26));

btnShowSketch.setBounds(1484, 772, 399, 39);

Input.add(btnShowSketch);

JButton btnClearVariable = new JButton("Clear Variable");

// Clear Variable Button

btnClearVariable.setBackground(Color.WHITE);

btnClearVariable.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

tool.clearTableValue(variableTable);

classDetailInfo.clearMemberVariable();

tool.showOnTextArea(textShowDetail, classDetailInfo);

}

});

JButton btnReadFunction = new JButton("Read Function");

btnReadFunction.setForeground(Color.WHITE);

btnReadFunction.setBounds(539, 693, 290, 46);

btnReadFunction.setBackground(Color.BLACK);

//Read Function Button

btnReadFunction.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

ArrayList<ClassMemberAbstract> member = tool.getTableValue(functionTable,memberFunction,"Function");

classDetailInfo.clearMemberFunction();

for ( int i=0; i < member.size();i++)

{

classDetailInfo.setMemberFunction(member.get(i));

String ans = member.get(i).getReference()+ " " + member.get(i).getType() + " " + member.get(i).getName() ;

textShowDetail.append(ans);

}

variableList.put(classDetailInfo.getClassName(), classDetailInfo);

tool.showOnTextArea(textShowDetail, classDetailInfo);

}

});

btnReadFunction.setFont(new Font("Arial", Font.PLAIN, 26));

Input.add(btnReadFunction);

btnClearVariable.setFont(new Font("Arial", Font.PLAIN, 26));

btnClearVariable.setBounds(278, 693, 247, 46);

Input.add(btnClearVariable);

JButton btnClearFunction = new JButton("Clear Function");

//Clear Function Button

btnClearFunction.setBackground(Color.WHITE);

btnClearFunction.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

tool.clearTableValue(functionTable);

classDetailInfo.clearMemberFunction();

tool.showOnTextArea(textShowDetail, classDetailInfo);

}

});

btnClearFunction.setFont(new Font("Arial", Font.PLAIN, 26));

btnClearFunction.setBounds(828, 693, 296, 46);

Input.add(btnClearFunction);

JButton btnClearRelationship = new JButton("Clear Relationship");

// Clear Relationship Button

btnClearRelationship.setBackground(Color.WHITE);

btnClearRelationship.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

tool.clearTableValue(relationshipTable);

classDetailInfo.clearClassRelarionship();

tool.showOnTextArea(textShowDetail, classDetailInfo);

}

});

btnClearRelationship.setFont(new Font("Arial", Font.PLAIN, 16));

btnClearRelationship.setBounds(1296, 693, 174, 46);

Input.add(btnClearRelationship);

JButton btnClearName = new JButton("Clear Name");

//Clear Name Button

btnClearName.setFont(new Font("Arial", Font.PLAIN, 26));

btnClearName.setBackground(Color.WHITE);

btnClearName.setBounds(1239, 68, 185, 39);

Input.add(btnClearName);

// Clear Name Button

btnClearName.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

textName.setText(null);

textShowDetail.setText("");

classDetailInfo.clearClassName();

tool.showOnTextArea(textShowDetail, classDetailInfo);

}

});

JComboBox comboBox\_variable = new JComboBox();

comboBox\_variable.setFont(new Font("標楷", Font.PLAIN, 20));

comboBox\_variable.addItem("Public");

comboBox\_variable.addItem("Private");

comboBox\_variable.addItem("Protected");

referenceColumn.setCellEditor(new DefaultCellEditor(comboBox\_variable));

JComboBox comboBox\_function = new JComboBox();

comboBox\_function.setFont(new Font("標楷", Font.PLAIN, 20));

comboBox\_function.addItem("Public");

comboBox\_function.addItem("Private");

comboBox\_function.addItem("Protected");

referenceColumn\_function.setCellEditor(new DefaultCellEditor(comboBox\_function));

JComboBox comboBox\_relationship = new JComboBox();

comboBox\_relationship.setFont(new Font("標楷", Font.PLAIN, 20));

comboBox\_relationship.addItem("Extension");

comboBox\_relationship.addItem("Implementation");

comboBox\_relationship.addItem("Association");

comboBox\_relationship.addItem("Aggregation");

comboBox\_relationship.addItem("Composition");

comboBox\_relationship.addItem("Dependency");

referenceColumn\_relationship.setCellEditor(new DefaultCellEditor(comboBox\_relationship));

}

}

### 6.2.3 Tool

package viewTool;

import java.util.ArrayList;

import javax.swing.JTable;

import javax.swing.JTextArea;

import ClassDetailInfo.ClassDetailInfo;

import ClassDetailInfo.ClassMemberAbstract;

import ClassDetailInfo.ClassRelarionship;

import ClassDetailInfo.MemberFunction;

import ClassDetailInfo.MemberVariable;

public class Tool {

public Tool() {}

public void clearTableValue(JTable temp)

{

int row = temp.getModel().getRowCount();

int column = temp.getModel().getColumnCount();

for (int i=0; i < row; i++)

{

for (int j=0; j < column; j++)

{

temp.getModel().setValueAt("",i,j);

}

}

}

public void showOnTextArea(JTextArea \_textShowDetail, ClassDetailInfo \_gp)

{

\_textShowDetail.setText("");

if ( ! \_gp.getClassName().isEmpty())

\_textShowDetail.append("Class Name : " + \_gp.getClassName() +"\n");

if ( ! \_gp.getMemberVariable().isEmpty() )

\_textShowDetail.append("MemberVariable : " +"\n");

for ( ClassMemberAbstract temp : \_gp.getMemberVariable())

{

String ans = temp.getReference()+ " " + temp.getType() + " " + temp.getName() + "\n";

\_textShowDetail.append(ans);

}

if ( ! \_gp.getMemberFunction().isEmpty() )

\_textShowDetail.append("MemberFunction : " +"\n");

for ( ClassMemberAbstract temp : \_gp.getMemberFunction())

{

String ans = temp.getReference()+ " " + temp.getType() + " " + temp.getName() + "\n";

\_textShowDetail.append(ans);

}

if ( ! \_gp.getClassRelarionship().isEmpty() )

\_textShowDetail.append("ClassRelarionship : " +"\n");

for ( ClassMemberAbstract temp : \_gp.getClassRelarionship())

{

String ans = temp.getReference()+ " " + temp.getName() + "\n";

\_textShowDetail.append(ans);

}

}

public ArrayList<ClassMemberAbstract> getTableValue(JTable temp,ClassMemberAbstract member,String choice)

{

ArrayList<ClassMemberAbstract> result =new ArrayList<ClassMemberAbstract>();

int row = temp.getModel().getRowCount();

int column = temp.getModel().getColumnCount();

Object selected=null;

for (int i=0; i < row; i++)

{

if ( choice == "Variable" ) member = new MemberVariable();

else if ( choice == "Function") member = new MemberFunction();

else if ( choice == "Relation" ) member = new ClassRelarionship();

else return null;

for (int j=0; j < column; j++)

{

selected = temp.getModel().getValueAt(i,j);

if ( selected != "" && selected != null )

{

if ( j == 0 ) member.setReference(selected.toString());

else if ( j == 1 && temp.getColumnName(j) != "Class Name" ) member.setType(selected.toString());

else member.setName(selected.toString());

}

}

if ( selected != "" && selected != null) result.add(member);

else break;

}

return result;

}

}

### 6.2.4 ClassFormat

package shapes;

import diagrams.Drawable;

import java.awt.\*;

import java.awt.geom.Point2D;

import java.util.ArrayList;

import java.util.List;

public abstract class ClassFormat extends Rectangle implements Drawable{

private String className;

private List<Relation> relations;

private List<String> methods;

private List<String> variables;

public ClassFormat(String className,int x,int y,int width,int height) {

super(x,y,width,height);

setClassName(className);

relations=new ArrayList<>();

}

@Override

public boolean contains(Point2D p){

if(width<=0 || height<=0)

return false;

else{

if((p.getX()>=x && p.getX()<=x+width) && (p.getY()>=y && p.getY()<=y+height))

return true;

}

return false;

}

@Override

public void setLocation(int x,int y){

int offsetX=x-this.x,offsetY=y-this.y;

Point offset=new Point(offsetX,offsetY);

super.setLocation(x,y);

notifyRelations(offset,this);

}

public void notifyRelations(Point offset,ClassFormat classFormat){

for(Relation relation:relations)

relation.update(offset.clone(),classFormat);

}

public void regisiterRelation(Relation relation){

relations.add(relation);

}

public void unregisiterRelation(Relation relation){

relations.remove(relation);

}

public String getClassName(){

return this.className;

}

public void setClassName(String className) {

this.className = className;

}

public List<String> getMethods() {

return methods;

}

public void setMethods(List<String> methods) {

this.methods = methods;

}

public List<String> getVariables() {

return variables;

}

public void setVariables(List<String> variables) {

this.variables = variables;

}

protected int drawVariable(Graphics g,int x,int y) {

FontMetrics m =g.getFontMetrics();

int lineHeight=m.getHeight();

for(String variable:variables) {

y+=lineHeight;

g.drawString(variable, x, y);

}

return y;

}

protected int drawMethod(Graphics g,int x,int y) {

FontMetrics m =g.getFontMetrics();

int lineHeight=m.getHeight();

for(String method:methods) {

y+=lineHeight;

g.drawString(method, x, y);

}

return y;

}

protected int drawClassName(Graphics g,int x,int y) {

g.drawString(className, x, y);

return y;

}

@Override

public void draw(Graphics g) {

FontMetrics m =g.getFontMetrics();

int lineHeight=m.getHeight();

int width=this.width,height=this.height;

int x=this.x,y=this.y,middle=(x\*2+width)/2,padding=5;

g.drawRect(x,y,width,height);

// y=drawClassName(g,middle,y+lineHeight);

y=drawClassName(g,x+padding,y+lineHeight);

y+=lineHeight;

g.drawLine(x, y,x+width,y);

y=drawVariable(g,x+padding,y);

y+=lineHeight;

g.drawLine(x, y,x+width,y);

y=drawMethod(g,x+padding,y);

}

}

### 6.2.5 ConcreteFormat

package shapes;

public class ConcreteFormat extends ClassFormat {

public ConcreteFormat(String className, int x, int y, int width, int height) {

super(className, x, y, width, height);

}

}

### 6.2.6 InterfaceFormat

package shapes;

import java.awt.\*;

public class InterfaceFormat extends ClassFormat {

public InterfaceFormat(String className, int x, int y, int width, int height) {

super(className, x, y, width, height);

}

@Override

protected int drawClassName(Graphics g,int x,int y){

int lineHeight=g.getFontMetrics().getHeight();

g.drawString("<<interface>>",x,y);

y+=lineHeight;

y=super.drawClassName(g,x,y);

return y;

}

@Override

protected int drawVariable(Graphics g,int x, int y){

y+=2\*g.getFontMetrics().getHeight();

return y;

}

}

### 6.2.7 Relation

package shapes;

import diagrams.Drawable;

import diagrams.RelationType;

import java.awt.\*;

import java.util.Observable;

import java.util.Observer;

public class Relation implements Drawable{

private ClassFormat startClass,endClass;

private RelationType relationType;

private int startX,endX,startY,endY;

public Relation(ClassFormat startClass , ClassFormat endClass , RelationType relationType){

setStartClass(startClass);

setEndClass(endClass);

setRelationType(relationType);

setStartX(startClass.x);

setStartY(startClass.y);

setEndX(endClass.x);

setEndY(endClass.y);

startClass.regisiterRelation(this);

endClass.regisiterRelation(this);

}

public ClassFormat getStartClass() {

return startClass;

}

public ClassFormat getEndClass() {

return endClass;

}

public void setStartX(int startX) {

this.startX = startX;

}

public void setEndX(int endX) {

this.endX = endX;

}

public void setStartY(int startY) {

this.startY = startY;

}

public void setEndY(int endY) {

this.endY = endY;

}

public void setStartClass(ClassFormat startClass) {

this.startClass = startClass;

}

public void setEndClass(ClassFormat endClass) {

this.endClass = endClass;

}

public void setRelationType(RelationType relationType) {

this.relationType = relationType;

}

@Override

public void draw(Graphics g) {

g.drawString(relationType.toString(),(startX+endX)/2,(startY+endY)/2);

g.drawLine(startX,startY,endX,endY);

}

public void update(Object location,ClassFormat classFormat) {

Point point=(Point)location;

if(point!=null){

int offsetX=point.x,offsetY=point.y;

if(classFormat==startClass){

startX+=offsetX;

startY+=offsetY;

}

else{

endX+=offsetX;

endY+=offsetY;

}

}

}

}

### 6.2.8 Listener

package listeners;

import com.sun.javafx.iio.ImageStorage;

import diagrams.UMLClassDiagramDrawer;

import javax.imageio.ImageIO;

import javax.swing.\*;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.swing.filechooser.FileSystemView;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

public class Listener {

UMLClassDiagramDrawer drawer;

private ListenerHandler listenerHandler;

private Point start,end;

public Listener(UMLClassDiagramDrawer drawer, ListenerHandler listenerHandler){

this.listenerHandler=listenerHandler;

this.drawer=drawer;

start=new Point(0,0);

end=new Point(0,0);

}

public ActionListener openFileListener=new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser=new JFileChooser();

fileChooser.setCurrentDirectory(new File(System.getProperty("user.dir")));

int returnValue=fileChooser.showOpenDialog(null);

if(returnValue==JFileChooser.APPROVE\_OPTION)

listenerHandler.executeOpenDiagram(fileChooser.getSelectedFile().getPath());

}

};

public ActionListener saveListener=new ActionListener(){

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser=new JFileChooser();

fileChooser.setCurrentDirectory(new File(System.getProperty("user.dir")));

int option=fileChooser.showSaveDialog(null);

if(option==JFileChooser.APPROVE\_OPTION){

String path=fileChooser.getSelectedFile().getPath();

if(!path.contains(".diagram"))

path+=".diagram";

listenerHandler.executeSaveDiagram(path);

}

}

};

public ActionListener outputListener =new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

JFileChooser fileChooser=new JFileChooser();

fileChooser.setCurrentDirectory(new File(System.getProperty("user.dir")));

int option=fileChooser.showSaveDialog(null);

if(option==JFileChooser.APPROVE\_OPTION){

String path=fileChooser.getSelectedFile().getPath();

String fileName=fileChooser.getName(fileChooser.getSelectedFile());

String desktopPath= FileSystemView.getFileSystemView().getHomeDirectory().getPath();

int type=JOptionPane.INFORMATION\_MESSAGE;

String[]formats=new String[]{"png","jpg"};

String format=(String)JOptionPane.showInputDialog(null,"請選擇格式","選擇輸出格式",type,null,formats,formats[0]);

if(format==null)

return;

BufferedImage bufferedImage=new BufferedImage(drawer.getWidth(),drawer.getHeight(), BufferedImage.TYPE\_4BYTE\_ABGR);

Graphics2D g=bufferedImage.createGraphics();

drawer.print(g);

try{

ImageIO.write(bufferedImage,format,new File(path+"."+format));

ImageIO.write(bufferedImage,format,new File(desktopPath+"\\"+fileName+"."+format));

}catch (IOException exception){

exception.printStackTrace();}

}

}

};

public ItemListener addRelation=new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

if(e.getStateChange()==1)

drawer.addMouseListener(addRelationAdapter);

else if(e.getStateChange()==2)

drawer.removeMouseListener(addRelationAdapter);

}

};

public ItemListener removeRelation=new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

if(e.getStateChange()==1)

drawer.addMouseListener(removeRelationAdapter);

else if(e.getStateChange()==2)

drawer.removeMouseListener(removeRelationAdapter);

}

};

public ItemListener moveUnit=new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

if(e.getStateChange()==1)

drawer.addMouseListener(moveUnitAdapter);

else if(e.getStateChange()==2)

drawer.removeMouseListener(moveUnitAdapter);

}

};

public ItemListener removeUnit=new ItemListener() {

@Override

public void itemStateChanged(ItemEvent e) {

if(e.getStateChange()==1)

drawer.addMouseListener(removeUnitAdapter);

else if(e.getStateChange()==2)

drawer.removeMouseListener(removeUnitAdapter);

}

};

private MouseAdapter addRelationAdapter=new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e) {

super.mousePressed(e);

start=e.getPoint();

}

@Override

public void mouseReleased(MouseEvent e) {

super.mouseReleased(e);

end=e.getPoint();

listenerHandler.executeAddRelation((Point)start.clone(),(Point)end.clone());

}

};

private MouseAdapter removeRelationAdapter=new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e) {

super.mousePressed(e);

listenerHandler.executeRemoveRelation(e.getPoint());

}

@Override

public void mouseReleased(MouseEvent e) {

super.mouseReleased(e);

}

};

private MouseAdapter moveUnitAdapter=new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e) {

super.mousePressed(e);

start=e.getPoint();

}

@Override

public void mouseReleased(MouseEvent e) {

super.mouseReleased(e);

end=e.getPoint();

listenerHandler.executeMoveUnit((Point)start.clone(),(Point)end.clone());

}

};

private MouseAdapter removeUnitAdapter=new MouseAdapter() {

@Override

public void mousePressed(MouseEvent e) {

super.mousePressed(e);

listenerHandler.executeRemoveUnit(e.getPoint());

}

};

}

### 6.2.9 ListenerHandler

package listeners;

import diagrams.UMLClassDiagram;

import generator.ArrangeCalculator;

import output.DiagramReader;

import output.DiagramWriter;

import java.awt.\*;

import java.io.IOException;

public class ListenerHandler {

private UMLClassDiagram diagram;

private String autoSavePath;

private ArrangeCalculator arrangeCalculator;

public ListenerHandler(UMLClassDiagram diagram){

this.autoSavePath=System.getProperty("user.dir")+"/Untitled\_diagram.diagram";

this.diagram=diagram;

arrangeCalculator=new ArrangeCalculator(diagram);

}

public void executeAddRelation(Point start, Point end){

arrangeCalculator.bindRelation(start,end);

executeSaveDiagram(autoSavePath);

}

public void executeMoveUnit(Point start , Point end){

arrangeCalculator.moveUnit(start,end);

executeSaveDiagram(autoSavePath);

}

public void executeRemoveUnit(Point location){

arrangeCalculator.removeUnit(location);

executeSaveDiagram(autoSavePath);

}

public void executeRemoveRelation(Point location){

arrangeCalculator.removeRelation(location);

executeSaveDiagram(autoSavePath);

}

public void executeSaveDiagram(String path){

autoSavePath=path;

DiagramWriter diagramWriter =new DiagramWriter(diagram);

try {

diagramWriter.save(path);

}catch (IOException ioe){

throw new RuntimeException("File didn't close");

}

}

public void executeOpenDiagram(String path){

try {

DiagramReader diagramReader = new DiagramReader(path);

diagram.setDiagram(diagramReader.getDiagram());

}catch (Exception e){

throw new RuntimeException("File not found");

}

}

}

### 6.2.10 ArrangeCalculator

package generator;

import diagrams.RelationType;

import diagrams.UMLClassDiagram;

import shapes.ClassFormat;

import shapes.Relation;

import java.awt.\*;

import java.awt.geom.Point2D;

import java.util.\*;

import java.util.List;

public class ArrangeCalculator {

private UMLClassDiagram diagram;

public ArrangeCalculator(UMLClassDiagram diagram) {

this.diagram = diagram;

}

public void arrange(){

int width=diagram.getWidth(),startX=50,offsetX=300,offsetY=300;

int x=startX,y=50;

Iterator<Map.Entry<String,ClassFormat>> classFormatIterator=diagram.createClassFormatIterator();

while(classFormatIterator.hasNext()){

if(x+offsetX>=width) {

y += offsetY;

x=startX;

}

ClassFormat format=classFormatIterator.next().getValue();

format.setLocation(x,y);

x+=offsetX;

}

}

public void removeUnit(ClassFormat classFormat){

diagram.removeClassFormat(classFormat);

}

public ClassFormat checkPointContains(Point point){

ClassFormat containClass=null;

Iterator<Map.Entry<String,ClassFormat>> classFormatIterator=diagram.createClassFormatIterator();

while (classFormatIterator.hasNext()){

ClassFormat classFormat=classFormatIterator.next().getValue();

if(classFormat.contains(point))

containClass=classFormat;

}

return containClass;

}

public void calculateArrange(Point start,Point end){

ClassFormat startClass=checkPointContains(start),endClass=checkPointContains(end);

if(startClass !=null && endClass!=null){

ClassRelationGenerator relationGenerator=new ClassRelationGenerator();

Relation relation=relationGenerator.generateRelation(startClass,endClass, RelationType.Association);

Point2D equation=solveEquation(start,end);

List<Point2D>startClassResult=getPossibleSolution(equation,startClass);

startClassResult=getRecContainsPoint(startClassResult,startClass);

List<Point2D>endClassResult=getPossibleSolution(equation,endClass);

endClassResult=getRecContainsPoint(endClassResult,endClass);

try {

List<Point2D>result=getMinStartEndPoint(startClassResult,endClassResult);

Point2D startPoint=result.get(0),endPoint=result.get(1);

relation.setStartX((int)startPoint.getX());

relation.setStartY((int)startPoint.getY());

relation.setEndX((int)endPoint.getX());

relation.setEndY((int)endPoint.getY());

diagram.addToDiagram(relation);

}catch (Exception e){

e.printStackTrace();

}

}

}

public void moveUnit(Point start , Point end){

ClassFormat startClass=null;

Iterator<Map.Entry<String,ClassFormat>> classFormatIterator=diagram.createClassFormatIterator();

while (classFormatIterator.hasNext()){

ClassFormat classFormat=classFormatIterator.next().getValue();

if(classFormat.contains(start)) {

startClass = classFormat;

System.out.println("start:" + startClass.getClassName());

}

}

if(startClass!=null) {

int offsetX = end.x - start.x, offsetY = end.y - start.y;

int newX=startClass.x+offsetX,newY=startClass.y+offsetY;

try {

diagram.setClassFormatLocation(startClass.getClassName(), newX, newY);

}

catch (Exception e){

e.printStackTrace();

}

}

}

Point2D solveEquation(Point start, Point end){

double a=(start.getY()-end.getY())/(start.getX()-end.getX()),b=(end.getY()\*start.getX()-start.getY()\*end.getX())/(start.getX()-end.getX());

Point2D p=new Point2D.Double(a,b);

return p;

}

List<Point2D> getRecContainsPoint(List<Point2D> possibleSolution,ClassFormat classFormat){

List<Point2D> result=new ArrayList<>();

for(Point2D p:possibleSolution) {

if (classFormat.contains(p))

result.add(p);

}

return result;

}

List<Point2D> getPossibleSolution(Point2D equation, ClassFormat classFormat){

String axis="xyxy";

double a=equation.getX(),b=equation.getY();

int[]value={classFormat.x,classFormat.y,classFormat.x+classFormat.width,classFormat.y+classFormat.height};

List<Point2D> solution=new ArrayList<>();

for(int i=0;i<axis.length();i++){

double x=0,y=0;

if(axis.charAt(i)=='x'){

x=value[i];

y=a\*x+b;

}

else if(axis.charAt(i)=='y'){

y=value[i];

x=(y-b)/a;

}

Point2D result=new Point2D.Double(x,y);

solution.add(result);

}

return solution;

}

List<Point2D> getMinStartEndPoint(List<Point2D>startClassResult,List<Point2D>endClassResult)throws Exception{

if(startClassResult.isEmpty() || endClassResult.isEmpty())

throw new Exception("The Two List should not be empty");

Point2D minStart=new Point(0,0),minEnd=new Point(0,0);

List<Point2D> minStartEnd=new ArrayList<>();

double minDistance=Double.MAX\_VALUE;

for(Point2D startP:startClassResult){

for (Point2D endP:endClassResult){

double distance=Point.distance(startP.getX(),startP.getY(),endP.getX(),endP.getY());

if(distance<=minDistance){

minStart=startP;

minEnd=endP;

minDistance=distance;

}

}

}

minStartEnd.add(minStart);

minStartEnd.add(minEnd);

return minStartEnd;

}

}

### 6.2.11 ClassRelationGenerator

package generator;

import diagrams.RelationType;

import shapes.ClassFormat;

import shapes.Relation;

public class ClassRelationGenerator {

public ClassRelationGenerator() {

}

public Relation generateRelation(ClassFormat startClass , ClassFormat endClass , RelationType relationType) {

Relation relation=new Relation(startClass,endClass,relationType);

return relation;

}

}

### 6.2.12 Drawable

package diagrams;

import java.awt.Graphics;

public interface Drawable {

void draw(Graphics g);

}

### 6.2.13 UMLClassDiagram

package diagrams;

import java.util.\*;

import java.util.List;

import shapes.ClassFormat;

import shapes.Relation;

public class UMLClassDiagram extends Observable{

private Map<String,ClassFormat> classFormatCollection;

private List<Relation> relations;

private int width,height;

public UMLClassDiagram(int width,int height) {

super();

classFormatCollection=new TreeMap<>();

relations=new ArrayList<>();

setWidth(width);

setHeight(height);

}

public void setDiagram(UMLClassDiagram diagram){

classFormatCollection.clear();

relations.clear();

Iterator<Map.Entry<String,ClassFormat>>classFormatIterator=diagram.createClassFormatIterator();

while (classFormatIterator.hasNext()) {

ClassFormat classFormat=classFormatIterator.next().getValue();

classFormatCollection.put(classFormat.getClassName(),classFormat);

}

Iterator<Relation>relationIterator=diagram.createRelationIterator();

while(relationIterator.hasNext()){

Relation relation=relationIterator.next();

relations.add(relation);

}

setHeight(diagram.getHeight());

setWidth(diagram.getWidth());

notifyObservers();

}

public int getWidth() {

return width;

}

public void setWidth(int width) {

this.width = width;

}

public int getHeight() {

return height;

}

public void setHeight(int height) {

this.height = height;

}

@Override

public void notifyObservers(Object object){

setChanged();

super.notifyObservers(object);

}

@Override

public void notifyObservers(){

setChanged();

super.notifyObservers();

}

public void addToDiagram(ClassFormat format){

classFormatCollection.put(format.getClassName(), format);

notifyObservers(format);

}

public void addToDiagram(Relation relation){

relations.add(relation);

notifyObservers(relation);

}

public void setClassFormatLocation(String className, int x, int y) throws Exception{

getClassFormat(className).setLocation(x, y);

notifyObservers();

}

public Iterator<Map.Entry<String,ClassFormat>> createClassFormatIterator(){

return classFormatCollection.entrySet().iterator();

}

public Iterator<Relation> createRelationIterator(){

return relations.iterator();

}

public ClassFormat getClassFormat(String className)throws Exception{

if(!classFormatCollection.containsKey(className))

throw new Exception("The class '"+className+"' is not in the diagram");

else

return classFormatCollection.get(className);

}

public void removeClassFormat(ClassFormat classFormat){

if(classFormat!=null && classFormatCollection.containsKey(classFormat.getClassName())) {

classFormatCollection.remove(classFormat.getClassName());

notifyObservers();

}

}

public void removeRelation(Relation relation){

if(relation!=null && relations.contains(relation)){

relations.remove(relation);

notifyObservers();

}

}

}

### 6.2.14 UMLClassDiagramDrawer

package diagrams;

import java.awt.Graphics;

import java.util.Iterator;

import java.util.Map;

import java.util.Observable;

import java.util.Observer;

import javax.swing.JPanel;

import shapes.ClassFormat;

import shapes.Relation;

public class UMLClassDiagramDrawer extends JPanel implements Observer {

private UMLClassDiagram diagram;

public UMLClassDiagramDrawer(UMLClassDiagram diagram) {

this.diagram=diagram;

this.diagram.addObserver(this);

setSize(diagram.getWidth(),diagram.getHeight());

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Iterator<Map.Entry<String,ClassFormat>> classFormats=diagram.createClassFormatIterator();

while (classFormats.hasNext()){

Map.Entry<String,ClassFormat>entry=classFormats.next();

entry.getValue().draw(g);

}

Iterator<Relation> relations=diagram.createRelationIterator();

while (relations.hasNext())

relations.next().draw(g);

}

@Override

public void update(Observable o, Object arg) {

Drawable drawable=(Drawable)arg;

if(drawable!=null)

drawable.draw(getGraphics());

else

repaint();

}

}

### 6.2.15 ClassDetailInfo

package ClassDetailInfo;

import java.util.ArrayList;

public class ClassDetailInfo {

private String \_class\_name;

private ArrayList<ClassMemberAbstract> \_MemberVariable=new ArrayList<ClassMemberAbstract>();

private ArrayList<ClassMemberAbstract> \_MemberFunction =new ArrayList<ClassMemberAbstract>();

private ArrayList<ClassMemberAbstract> \_ClassRelarionship = new ArrayList<ClassMemberAbstract>();

public ClassDetailInfo() {}

public ClassDetailInfo(String class\_name, MemberVariable MemberVariable,MemberFunction MemberFunction, ClassMemberAbstract ClassRelarionship) {

setClassName(class\_name);

setMemberVariable(MemberVariable);

setMemberFunction(MemberFunction);

setClassRelarionship(ClassRelarionship);

}

public String getClassName() {

return \_class\_name;

}

public void setClassName(String \_class\_name) {

this.\_class\_name = \_class\_name;

}

public void clearClassName()

{

this.\_class\_name = "";

}

public ArrayList<ClassMemberAbstract> getMemberVariable() {

return \_MemberVariable;

}

public void setMemberVariable(ClassMemberAbstract variableInfo) {

this.\_MemberVariable.add(variableInfo);

}

public void clearMemberVariable() {

this.\_MemberVariable.clear();

}

public ArrayList<ClassMemberAbstract> getMemberFunction() {

return \_MemberFunction;

}

public void setMemberFunction(ClassMemberAbstract MemberFunction) {

this.\_MemberFunction.add(MemberFunction);

}

public void clearMemberFunction() {

this.\_MemberFunction.clear();

}

public ArrayList<ClassMemberAbstract> getClassRelarionship() {

return \_ClassRelarionship;

}

public void setClassRelarionship(ClassMemberAbstract ClassRelarionship) {

this.\_ClassRelarionship.add( ClassRelarionship);

}

public void clearClassRelarionship() {

this.\_ClassRelarionship.clear();

}

}

### 6.2.16 ClassMemberAbstract

package ClassDetailInfo;

public abstract class ClassMemberAbstract {

public void setReference(String \_reference) {}

public void setName(String \_name) {}

public void setType(String \_type) {}

public void reSet(){}

public String getReference()

{

return null;

}

public String getType()

{

return null;

}

public String getName()

{

return null;

}

}

### 6.2.17 ClassRelationship

package ClassDetailInfo;

public class ClassRelarionship extends ClassMemberAbstract {

private String \_realtionship, \_className;

public ClassRelarionship() {}

public ClassRelarionship(String realtionship, String className)

{

\_realtionship = realtionship;

\_className = className;

}

@Override

public String getName() {

return \_className;

}

@Override

public void setName(String \_className) {

this.\_className = \_className;

}

@Override

public String getReference() {

return \_realtionship;

}

@Override

public void setReference(String \_realtionship) {

this.\_realtionship = \_realtionship;

}

@Override

public void reSet()

{

\_realtionship = "";

\_className = "";

}

}

### 6.2.18 MemberFunction

package ClassDetailInfo;

public class MemberFunction extends ClassMemberAbstract{

private String \_reference, \_functuonType, \_functuonName;

public MemberFunction() {}

public MemberFunction(String reference, String functuonType, String functuonName)

{

\_reference = reference;

\_functuonType = functuonType;

\_functuonName = functuonName;

}

@Override

public String getType() {

return \_functuonType;

}

@Override

public void setType(String \_functuonType) {

this.\_functuonType = \_functuonType;

}

@Override

public String getReference() {

return \_reference;

}

@Override

public void setReference(String \_reference) {

this.\_reference = \_reference;

}

@Override

public String getName() {

return \_functuonName;

}

@Override

public void setName(String \_functuonName) {

this.\_functuonName = \_functuonName;

}

@Override

public void reSet()

{

\_reference = "";

\_functuonType = "";

\_functuonName = "";

}

}

### 6.2.19 MemberVariable

package ClassDetailInfo;

public class MemberVariable extends ClassMemberAbstract{

private String \_reference, \_variableType, \_variableName;

public MemberVariable() {}

public MemberVariable(String reference, String variableType, String variableName)

{

\_reference = reference;

\_variableType = variableType;

\_variableName = variableName;

}

@Override

public String getReference() {

return \_reference;

}

@Override

public void setReference(String reference) {

this.\_reference = reference;

}

@Override

public String getType() {

return \_variableType;

}

@Override

public void setType(String variableType) {

this.\_variableType = variableType;

}

@Override

public String getName() {

return \_variableName;

}

@Override

public void setName(String variableName) {

this.\_variableName = variableName;

}

@Override

public void reSet()

{

\_reference = "";

\_variableType = "";

\_variableName = "";

}

}

### 6.2.20 ClassDetailInfoDTO

package adapter;

import ClassDetailInfo.\*;

import java.util.ArrayList;

import java.util.List;

public class ClassDetailInfoDTO {

private List<String> methods;

private List<String> variables;

private String className;

public ClassDetailInfoDTO(ClassDetailInfo classDetailInfo){

setClassDetailInfo(classDetailInfo);

}

public List<String> getMethods() {

return methods;

}

public List<String> getVariables() {

return variables;

}

public String getClassName() {

return className;

}

public void setClassDetailInfo(ClassDetailInfo classDetailInfo) {

parseClassName(classDetailInfo);

parseMethod(classDetailInfo);

parseVariable(classDetailInfo);

}

private void parseMethod(ClassDetailInfo classDetailInfo){

List<ClassMemberAbstract>memberMethods=classDetailInfo.getMemberFunction();

List<String>methods=new ArrayList<>();

for(ClassMemberAbstract c:memberMethods)

methods.add(parseFormat(c));

this.methods=methods;

}

private void parseVariable(ClassDetailInfo classDetailInfo){

List<ClassMemberAbstract>memberVariable=classDetailInfo.getMemberVariable();

List<String>variables=new ArrayList<>();

for(ClassMemberAbstract c:memberVariable)

variables.add(parseFormat(c));

this.variables=variables;

}

private void parseClassName(ClassDetailInfo classDetailInfo){

this.className=classDetailInfo.getClassName();

public String parseFormat(ClassMemberAbstract classMemberAbstract){

StringBuilder builder=new StringBuilder();

switch (classMemberAbstract.getReference()){

case "Public":

builder.append(" + ");

break;

case "Private":

builder.append(" - ");

break;

case "Protected":

builder.append(" # ");

break;

case "Package":

builder.append(" ~ ");

default:

break;

}

builder.append(classMemberAbstract.getName()).append(" : ").append(classMemberAbstract.getType());

return builder.toString();

}

}

### 6.2.21 ClassUnitGenerator

package generator;

import java.util.List;

import adapter.ClassDetailInfoDTO;

import shapes.ClassFormat;

import shapes.ConcreteFormat;

import shapes.InterfaceFormat;

public class ClassUnitGenerator {

private int x,y,width,height;

public ClassUnitGenerator(){

x=50;

y=50;

width=200;

height=200;

}

public ClassFormat generateConcreteClassFormat(ClassDetailInfoDTO dto) {

ClassFormat classFormat=new ConcreteFormat(dto.getClassName(),x,y,width,height);

classFormat.setMethods(dto.getMethods());

classFormat.setVariables(dto.getVariables());

return classFormat;

}

public ClassFormat generateConcreteClassFormat(String className, List<String> methods,List<String>variables) {

ClassFormat classFormat=new ConcreteFormat(className,x,y,width,height);

classFormat.setMethods(methods);

classFormat.setVariables(variables);

return classFormat;

}

public ClassFormat generateInterfaceClassFormat(String className, List<String> methods,List<String>variables) {

ClassFormat classFormat=new InterfaceFormat(className,x,y,width,height);

classFormat.setMethods(methods);

classFormat.setVariables(variables);

return classFormat;

}

}

### 6.2.22 [ClassFormatOutputDTO](https://github.com/JoeLiu1321/OOAD/commit/f97d19aa952ee7b257a90de54d7ba4afb9adc16b#diff-a3d96276f9938092ccd14fdf1274fdf3)

|  |
| --- |
|  |
| Package adapter; |
|  |

|  |  |
| --- | --- |
|  | import shapes.ClassFormat; |
|  |  |
|  | import java.util.List; |
|  |  |
|  | public class ClassFormatOutputDTO { |
|  | private String output; |
|  |  |
|  | public ClassFormatOutputDTO(ClassFormat classFormat) { |
|  | parseOutput(classFormat); |
|  | } |
|  |  |
|  | public void parseOutput(ClassFormat classFormat){ |
|  | StringBuilder builder=new StringBuilder(); |
|  | builder.append("ClassName:").append(classFormat.getClassName()); |
|  | builder.append(";"); |
|  | builder.append("Method:"); |
|  | List<String> method=classFormat.getMethods(); |
|  | for(int i=0;i<method.size();i++) { |
|  | builder.append(method.get(i)); |
|  | if (i != method.size()-1) |
|  | builder.append(","); |
|  | } |
|  | builder.append(";"); |
|  | builder.append("Variable:"); |
|  | List<String> variable=classFormat.getVariables(); |
|  | for(int i=0;i<variable.size();i++) { |
|  | builder.append(variable.get(i)); |
|  | if (i != variable.size()-1) |
|  | builder.append(","); |
|  | } |
|  | builder.append(";"); |
|  | builder.append("StartPoint:").append(classFormat.x).append(",").append(classFormat.y); |
|  | builder.append(";"); |
|  | builder.append("HeightWidth:").append(classFormat.height).append(",").append(classFormat.width); |
|  | output=builder.toString(); |
|  | } |
|  |  |
|  | public String getOutput(){ |
|  | return output; |
|  | } |
|  | } |

### 6.2.23 DiagramReader

package output;

import diagrams.RelationType;

import diagrams.UMLClassDiagram;

import generator.ClassRelationGenerator;

import generator.ClassUnitGenerator;

import shapes.ClassFormat;

import shapes.Relation;

import java.awt.\*;

import java.util.\*;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.util.List;

public class DiagramReader {

private UMLClassDiagram diagram;

public DiagramReader(String path)throws Exception{

Toolkit tk = Toolkit.getDefaultToolkit();

Dimension d = tk.getScreenSize();

int width=(int)d.getWidth(),height=(int)d.getHeight();

diagram=new UMLClassDiagram(width,height);

readDiagram(path);

}

public void readDiagram(String path)throws Exception {

BufferedReader br=new BufferedReader(new FileReader(new File(path)));

String line="";

List<String> lineData=new ArrayList<>();

Map<String,Map<String,String>> allData=new HashMap<>();

while((line=br.readLine())!=null)

lineData.add(line);

for(int i=0;i<lineData.size();i++) {

String[] attributes = lineData.get(i).split(";");

Map<String, String> dataDetail = new HashMap<>();

for (String attr : attributes) {

String[] attrPair = attr.split(":");

if(attrPair.length>1)

dataDetail.put(attrPair[0], attrPair[1]);

else

dataDetail.put(attrPair[0],"");

}

allData.put(String.valueOf(i), dataDetail);

}

convertDiagram(allData);

}

private void convertDiagram(Map<String,Map<String,String>> diagramData){

ClassUnitGenerator classUnitGenerator=new ClassUnitGenerator();

ClassRelationGenerator classRelationGenerator=new ClassRelationGenerator();

for(Map.Entry<String,Map<String,String>>attributes:diagramData.entrySet()){

Map<String,String>attribute=attributes.getValue();

if (attribute.containsKey("ClassName")){

String className=attribute.get("ClassName");

String[]method=(attribute.get("Method").contains(",")?attribute.get("Method").split(","):new String[]{});

String[]variable=(attribute.get("Variable").contains(",")?attribute.get("Variable").split(","):new String[]{});

List<String>methods=Arrays.asList(method);

List<String>variables=Arrays.asList(variable);

String[]heightWidth=attribute.get("HeightWidth").split(",");

String[]startPoint=attribute.get("StartPoint").split(",");

classUnitGenerator.setClassAttributes(className,methods,variables);

ClassFormat classFormat=classUnitGenerator.generateConcreteClassFormat();

classFormat.setSize(Integer.parseInt(heightWidth[1]),Integer.parseInt(heightWidth[0]));

classFormat.setLocation(Integer.parseInt(startPoint[0]),Integer.parseInt(startPoint[1]));

diagram.addToDiagram(classFormat);

}

else if(attribute.containsKey("StartClass")){

String startClass=attribute.get("StartClass");

String endClass=attribute.get("EndClass");

try {

Relation relation=classRelationGenerator.generateRelation(diagram.getClassFormat(startClass),diagram.getClassFormat(endClass), RelationType.valueOf(attribute.get("Type")));

String[]startPoint=attribute.get("StartPoint").split(",");

String[]endPoint=attribute.get("EndPoint").split(",");

relation.setStartX(Integer.parseInt(startPoint[0]));

relation.setStartY(Integer.parseInt(startPoint[1]));

relation.setEndX(Integer.parseInt(endPoint[0]));

relation.setEndY(Integer.parseInt(endPoint[1]));

diagram.addToDiagram(relation);

}catch (Exception e){

throw new RuntimeException("The Relation start or end class not found");

}

}

}

}

public UMLClassDiagram getDiagram() {

return diagram;

}

}

### 6.2.24 DiagramWriter

package output;

import adapter.ClassFormatOutputDTO;

import adapter.RelationOutputDTO;

import diagrams.UMLClassDiagram;

import shapes.ClassFormat;

import shapes.Relation;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Iterator;

import java.util.Map;

public class DiagramWriter {

private UMLClassDiagram diagram;

public DiagramWriter(UMLClassDiagram diagram){

this.diagram=diagram;

}

public void save(String path)throws IOException{

FileWriter fw=null;

try {

fw = new FileWriter(new File(path));

Iterator<Map.Entry<String, ClassFormat>> classFormatIterator=diagram.createClassFormatIterator();

Iterator<Relation> relationIterator=diagram.createRelationIterator();

while (classFormatIterator.hasNext()){

ClassFormat classFormat=classFormatIterator.next().getValue();

ClassFormatOutputDTO classFormatOutputDTO=new ClassFormatOutputDTO(classFormat);

fw.write(classFormatOutputDTO.getOutput());

fw.write("\r\n");

}

while(relationIterator.hasNext()) {

Relation relation = relationIterator.next();

RelationOutputDTO relationOutputDTO = new RelationOutputDTO(relation);

fw.write(relationOutputDTO.getOutput());

fw.write("\r\n");

}

}catch (IOException exception){

System.out.println("Invalid path");

}

finally {

fw.close();

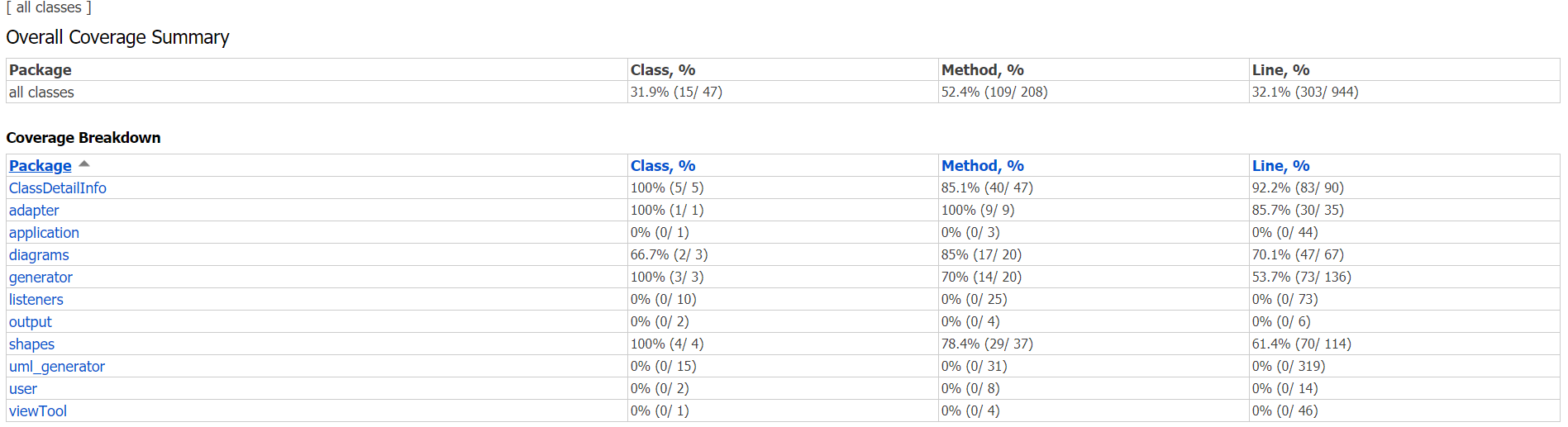
}

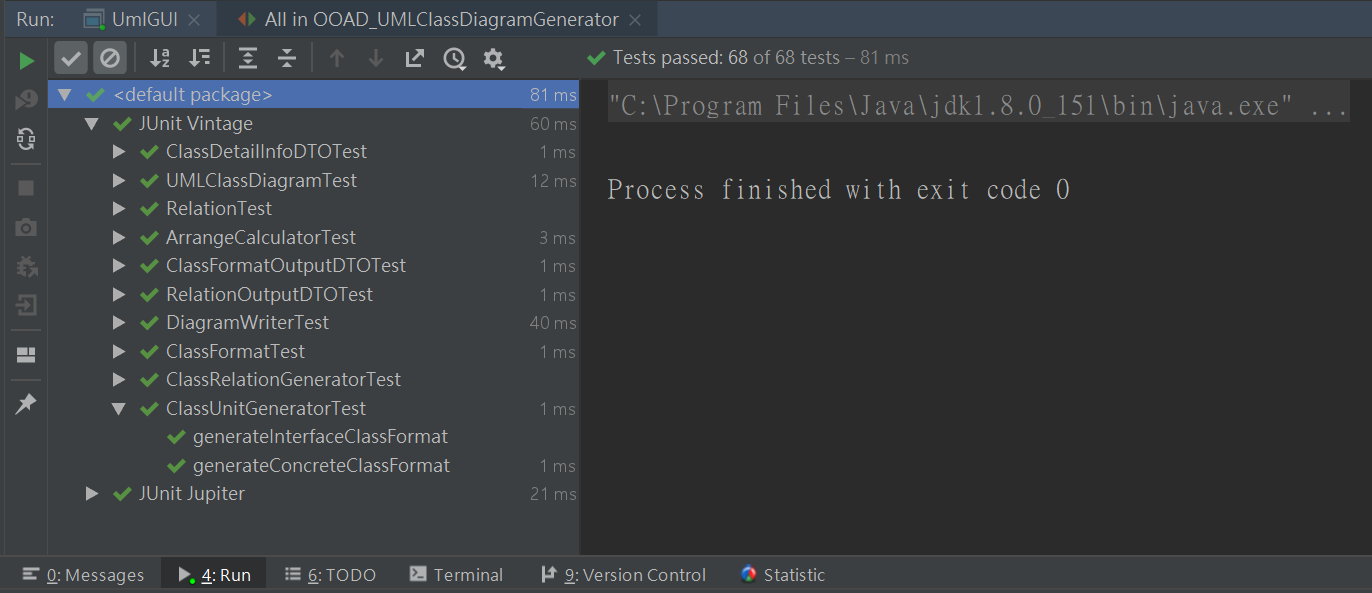
}

}

# 7 Unit Testing

# 7.1 Snapshots of testing result





# 7.2 Unit Testing Code Listing

### 7.2.1 ClassDetailInfoDTOTest

package adapter;

import ClassDetailInfo.\*;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class ClassDetailInfoDTOTest {

ClassDetailInfoDTO dto;

@Before

public void setUp(){

ClassRelationship relation=new ClassRelationship("implement","main");

MemberVariable variable=new MemberVariable("Public","int","test");

MemberFunction method =new MemberFunction("Public","int","test()");

ClassDetailInfo info=new ClassDetailInfo("main",variable,method,relation);

dto=new ClassDetailInfoDTO(info);

}

@Test

public void getMethods() {

assertEquals(" + test() : int",dto.getMethods().get(0));

}

@Test

public void getVariables() {

assertEquals(" + test : int",dto.getVariables().get(0));

}

@Test

public void getClassName() {

assertEquals("main",dto.getClassName());

}

}

### 7.2.2 ClassDetailInfoTest

package ClassDetailInfo;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

class ClassDetailInfoTest {

private ClassDetailInfo classDetailInfo;

private ClassMemberAbstract memberVariable;

private ClassMemberAbstract memberFunction;

private ClassMemberAbstract relation;

@BeforeEach

void setUp() throws Exception {

memberVariable = new MemberVariable("Public", "int","GP");

memberFunction = new MemberFunction("Public","int","draw(int x, int y)");

relation = new ClassRelationship("Extension","Draw");

classDetailInfo = new ClassDetailInfo();

classDetailInfo.setClassName("UML Generator");

classDetailInfo.setMemberVariable(memberVariable);

classDetailInfo.setMemberFunction(memberFunction);

classDetailInfo.setClassRelarionship(relation);

}

@Test

void testGetClassName() {

assertEquals("UML Generator", classDetailInfo.getClassName());

}

@Test

void testClearClassName() {

classDetailInfo.clearClassName();

assertEquals("", classDetailInfo.getClassName());

}

@Test

void testGetMemberVariable() {

assertEquals(memberVariable, classDetailInfo.getMemberVariable().get(0));

}

@Test

void testClearMemberVariable() {

classDetailInfo.clearMemberVariable();

assertTrue(classDetailInfo.getMemberVariable().isEmpty());

}

@Test

void testGetMemberFunction() {

assertEquals(memberFunction, classDetailInfo.getMemberFunction().get(0));

}

@Test

void testClearMemberFunction() {

classDetailInfo.clearMemberFunction();

assertTrue(classDetailInfo.getMemberFunction().isEmpty());

}

@Test

void testGetClassRelarionship() {

assertEquals(relation, classDetailInfo.getClassRelarionship().get(0));

}

@Test

void testClearClassRelarionship() {

classDetailInfo.clearClassRelarionship();

assertTrue(classDetailInfo.getClassRelarionship().isEmpty());

}

}

### 7.2.3 ClassRelationshipTest

/\*\*

\*

\*/

package ClassDetailInfo;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

/\*\*

\* @author GP

\*

\*/

class ClassRelationshipTest {

/\*\*

\* @throws Exception

\*/

private ClassMemberAbstract relation;

@BeforeEach

void setUp() throws Exception {

relation = new ClassRelationship();

relation.setReference("Extension");

relation.setName("GP");

}

/\*\*

\* Test method for {@link ClassRelationship#reSet()}.

\*/

@Test

void testReSet() {

relation.reSet();

assertEquals("", relation.getReference());

assertEquals("", relation.getName());

}

/\*\*

\* Test method for {@link ClassRelationship#getReference()}.

\*/

@Test

void testGetReference() {

assertEquals("Extension", relation.getReference());

}

/\*\*

\* Test method for {@link ClassRelationship#getName()}.

\*/

@Test

void testGetName() {

assertEquals("GP", relation.getName());

}

}

### 7.2.4 MemberFunctionTest

package ClassDetailInfo;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

class MemberFunctionTest {

private ClassMemberAbstract memberFunction;

@BeforeEach

void setUp() throws Exception {

memberFunction = new MemberFunction();

memberFunction.setReference("Public");

memberFunction.setType("int");

memberFunction.setName("draw(int x, int y)");

}

@Test

void testReSet() {

memberFunction.reSet();

assertEquals("", memberFunction.getReference());

assertEquals("", memberFunction.getType());

assertEquals("", memberFunction.getName());

}

@Test

void testGetReference() {

assertEquals("Public", memberFunction.getReference());

}

@Test

void testGetType() {

assertEquals("int", memberFunction.getType());

}

@Test

void testGetName() {

assertEquals("draw(int x, int y)", memberFunction.getName());

}

}

### 7.2.5 MemberVariableTest

package ClassDetailInfo;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

class MemberVariableTest {

private ClassMemberAbstract memberVariable;

@BeforeEach

void setUp() throws Exception {

memberVariable = new MemberVariable();

memberVariable.setReference("Public");

memberVariable.setType("int");

memberVariable.setName("GP");

}

@Test

void testReSet() {

memberVariable.reSet();

assertEquals("", memberVariable.getReference());

assertEquals("", memberVariable.getType());

assertEquals("", memberVariable.getName());

}

@Test

void testGetReference() {

assertEquals("Public", memberVariable.getReference());

}

@Test

void testGetType() {

assertEquals("int", memberVariable.getType());

}

@Test

void testGetName() {

assertEquals("GP", memberVariable.getName());

}

}

### 7.2.6 UMLClassDiagramTest

package diagrams;

import generator.ClassRelationGenerator;

import generator.ClassUnitGenerator;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import shapes.ClassFormat;

import shapes.Relation;

import java.awt.\*;

import java.util.Arrays;

import java.util.Iterator;

import static org.junit.jupiter.api.Assertions.\*;

public class UMLClassDiagramTest {

private UMLClassDiagram diagram;

private final String[]methods={"testMethod()","main()"},variables={"int tag","String name"};

private final String startClassName ="main",endClassName="UMLClassDiagram";

private ClassFormat startClass,endClass;

private Relation relation;

private final int width=500,height=500;

@Before

public void setUp(){

diagram=new UMLClassDiagram(width,height);

ClassUnitGenerator classUnitGenerator = new ClassUnitGenerator();

ClassRelationGenerator classRelationGenerator = new ClassRelationGenerator();

classUnitGenerator.setClassAttributes(startClassName, Arrays.asList(methods),Arrays.asList(variables));

startClass=classUnitGenerator.generateConcreteClassFormat();

classUnitGenerator.setClassAttributes(endClassName, Arrays.asList(methods),Arrays.asList(variables));

endClass=classUnitGenerator.generateConcreteClassFormat();

relation= classRelationGenerator.generateRelation(startClass,endClass,RelationType.Association);

}

@Test

public void getWidth() {

assertEquals(width,diagram.getWidth());

}

@Test

public void setWidth() {

diagram.setWidth(0);

assertEquals(0,diagram.getWidth());

}

@Test

public void getHeight() {

assertEquals(height,diagram.getHeight());

}

@Test

public void setHeight() {

diagram.setHeight(-50);

assertEquals(-50,diagram.getHeight());

}

@Test

public void addRelationToDiagram() {

diagram.addToDiagram(relation);

Iterator<Relation> iterator=diagram.createRelationIterator();

assertEquals(relation,iterator.next());

}

@Test

public void addClassFormatToDiagram()throws Exception {

diagram.addToDiagram(startClass);

assertSame(startClass, diagram.getClassFormat(startClass.getClassName()));

}

@Test

public void setClassFormatLocation() throws Exception{

Point expectLocation=new Point(50,50);

diagram.addToDiagram(endClass);

diagram.setClassFormatLocation(endClassName,expectLocation.x,expectLocation.y);

assertEquals(expectLocation,diagram.getClassFormat(endClassName).getLocation());

}

@Test

public void createClassFormatIterator() {

diagram.addToDiagram(startClass);

ClassFormat actual=diagram.createClassFormatIterator().next().getValue();

assertSame(startClass, actual);

}

@Test

public void createRelationIterator() {

diagram.addToDiagram(relation);

Relation actual=diagram.createRelationIterator().next();

assertEquals(relation,actual);

}

@Test(expected = Exception.class)

public void getClassFormat() throws Exception{

diagram.addToDiagram(endClass);

assertSame(endClass,diagram.getClassFormat(endClass.getClassName()));

assertNull(diagram.getClassFormat(startClass.getClassName()));

}

@Test(expected = Exception.class)

public void removeClassFormat() throws Exception{

diagram.addToDiagram(startClass);

diagram.removeClassFormat(startClass);

assertNull(diagram.getClassFormat(startClass.getClassName()));

}

@Test

public void removeRelation() {

diagram.addToDiagram(relation);

diagram.removeRelation(relation);

Boolean actual=diagram.createRelationIterator().hasNext();

assertEquals(false,actual);

}

}

### 7.2.7 ArrangeCalculatorTest

package generator;

import diagrams.UMLClassDiagram;

import generator.ArrangeCalculator;

import shapes.ClassFormat;

import shapes.ConcreteFormat;

import org.junit.Test;

import java.awt.Point;

import java.awt.geom.Point2D;

import java.util.ArrayList;

import java.util.List;

import static org.junit.Assert.\*;

public class ArrangeCalculatorTest {

@Test

public void arrange() {

}

@Test

public void calculateArrange() {

}

@Test

public void moveUnit() {

}

@Test

public void solveEquation(){

UMLClassDiagram classDiagram=new UMLClassDiagram(500,500);

ArrangeCalculator arrangeCalculator=new ArrangeCalculator(classDiagram);

Point start=new Point(60,60),end=new Point(480,270);

Point2D slopeAndOffset=arrangeCalculator.solveEquation(start,end);

assertEquals(0.5,slopeAndOffset.getX(),0.1);

assertEquals(30,slopeAndOffset.getY(),0.1);

}

@Test

public void solveEquation1(){

UMLClassDiagram classDiagram=new UMLClassDiagram(500,500);

ArrangeCalculator arrangeCalculator=new ArrangeCalculator(classDiagram);

Point start=new Point(60,60),end=new Point(240,480);

Point2D slopeAndOffset=arrangeCalculator.solveEquation(start,end);

assertEquals(2.33,slopeAndOffset.getX(),0.1);

assertEquals(-80,slopeAndOffset.getY(),0.1);

}

@Test

public void getPossibleSolution(){

UMLClassDiagram classDiagram=new UMLClassDiagram(500,500);

ClassFormat startClass=new ConcreteFormat("StartClass",10,10,100,100);

ClassFormat endClass=new ConcreteFormat("EndClass",190,430,100,100);

classDiagram.addToDiagram(startClass);

classDiagram.addToDiagram(endClass);

ArrangeCalculator arrangeCalculator=new ArrangeCalculator(classDiagram);

Point start=new Point(60,60),end=new Point(240,480);

Point2D slopeAndOffset=arrangeCalculator.solveEquation(start,end);

List<Point2D>startClassResult=arrangeCalculator.getPossibleSolution(slopeAndOffset,startClass);

assertEquals(new Point2D.Double(10.0,-56.666666666666664),startClassResult.get(0));

assertEquals(new Point2D.Double(38.57142857142857,10.0),startClassResult.get(1));

}

@Test

public void getRecContainsPoint(){

UMLClassDiagram classDiagram=new UMLClassDiagram(500,500);

ClassFormat startClass=new ConcreteFormat("StartClass",10,10,100,100);

ClassFormat endClass=new ConcreteFormat("EndClass",190,430,100,100);

classDiagram.addToDiagram(startClass);

classDiagram.addToDiagram(endClass);

ArrangeCalculator arrangeCalculator=new ArrangeCalculator(classDiagram);

Point start=new Point(60,60),end=new Point(240,480);

Point2D slopeAndOffset=arrangeCalculator.solveEquation(start,end);

List<Point2D>startClassResult=arrangeCalculator.getPossibleSolution(slopeAndOffset,startClass);

startClassResult=arrangeCalculator.getRecContainsPoint(startClassResult,startClass);

assertEquals(true,startClass.contains(startClassResult.get(0)));

assertEquals(true,startClass.contains(startClassResult.get(1)));

List<Point2D>endClassResult=arrangeCalculator.getPossibleSolution(slopeAndOffset,endClass);

endClassResult=arrangeCalculator.getRecContainsPoint(endClassResult,endClass);

assertEquals(true,endClass.contains(endClassResult.get(0)));

assertEquals(true,endClass.contains(endClassResult.get(1)));

}

@Test

public void getMinStartEndPoint()throws Exception{

UMLClassDiagram classDiagram=new UMLClassDiagram(500,500);

ArrangeCalculator arrangeCalculator=new ArrangeCalculator(classDiagram);

List<Point2D> start=new ArrayList<>();

List<Point2D> end=new ArrayList<>();

start.add(new Point(45,10));

start.add(new Point(95,110));

end.add(new Point(255,430));

end.add(new Point(290,500));

List<Point2D> result = arrangeCalculator.getMinStartEndPoint(start, end);

assertEquals(new Point(95,110),result.get(0));

assertEquals(new Point(255,430),result.get(1));

}

}

### 7.2.8 ClassRelationGeneratorTest

package generator;

import diagrams.RelationType;

import org.junit.Before;

import org.junit.Test;

import shapes.ClassFormat;

import shapes.Relation;

import java.util.Arrays;

import static org.junit.Assert.\*;

public class ClassRelationGeneratorTest {

private final String[]methods={"testMethod()","main()"},variables={"int tag","String name"};

private final String startClassName ="main",endClassName="UMLClassDiagram";

private ClassFormat startClass,endClass;

@Before

public void setUp(){

ClassUnitGenerator classUnitGenerator=new ClassUnitGenerator();

classUnitGenerator.setClassAttributes(startClassName, Arrays.asList(methods),Arrays.asList(variables));

startClass=classUnitGenerator.generateConcreteClassFormat();

classUnitGenerator.setClassAttributes(endClassName, Arrays.asList(methods),Arrays.asList(variables));

endClass=classUnitGenerator.generateConcreteClassFormat();

}

@Test

public void generateRelation() {

ClassRelationGenerator classRelationGenerator=new ClassRelationGenerator();

Relation actualRelation=classRelationGenerator.generateRelation(startClass,endClass, RelationType.Association);

assertSame(startClass,actualRelation.getStartClass());

assertSame(endClass,actualRelation.getEndClass());

assertEquals(RelationType.Association,actualRelation.getRelationType());

}

}

### 7.2.9 ClassUnitGeneratorTest

package generator;

import ClassDetailInfo.\*;

import adapter.ClassDetailInfoDTO;

import org.junit.Before;

import org.junit.Test;

import shapes.ClassFormat;

import java.util.Arrays;

import static org.junit.Assert.assertArrayEquals;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class ClassUnitGeneratorTest {

private final String[]methods={"testMethod()","main()"},variables={"int tag","String name"};

private final String startClassName ="main";

private ClassUnitGenerator classUnitGenerator;

private ClassDetailInfoDTO dto;

@Before

public void setUp(){

ClassRelationship relation=new ClassRelationship("implement","main");

MemberVariable variable=new MemberVariable("Public","int","test");

MemberFunction method =new MemberFunction("Public","int","test()");

ClassDetailInfo info=new ClassDetailInfo("main",variable,method,relation);

dto =new ClassDetailInfoDTO(info);

classUnitGenerator=new ClassUnitGenerator(dto);

}

@Test

public void generateConcreteClassFormat() {

classUnitGenerator.setClassAttributes(startClassName, Arrays.asList(methods),Arrays.asList(variables));

ClassFormat startClass=classUnitGenerator.generateConcreteClassFormat();

assertEquals(startClassName,startClass.getClassName());

assertArrayEquals(methods,startClass.getMethods().toArray());

assertArrayEquals(variables,startClass.getVariables().toArray());

}

@Test

public void generateInterfaceClassFormat() {

classUnitGenerator.setClassAttributes(dto);

ClassFormat startClass=classUnitGenerator.generateInterfaceClassFormat();

assertEquals(dto.getClassName(),startClass.getClassName());

assertArrayEquals(dto.getMethods().toArray(),startClass.getMethods().toArray());

assertArrayEquals(dto.getVariables().toArray(),startClass.getVariables().toArray());

}

}

### 7.2.10 ClassFormatTest

package shapes;

import org.junit.Before;

import org.junit.Test;

import java.awt.\*;

import java.awt.geom.Point2D;

import java.util.Arrays;

import static org.junit.Assert.\*;

public class ClassFormatTest {

private ClassFormat classFormat;

private String className="mainTest";

private int width=100;

private int height=100;

@Before

public void setUp(){

int x = 0;

int y = 0;

classFormat=new ConcreteFormat(className, x,y,width,height);

}

@Test

public void contains() {

assertTrue(classFormat.contains(new Point2D.Double(50,50)));

assertFalse(classFormat.contains(new Point2D.Double(width+1,height+1)));

}

@Test

public void setLocation() {

classFormat.setLocation(400,400);

assertEquals(new Point(400,400),classFormat.getLocation());

}

@Test

public void getClassName() {

assertEquals(className,classFormat.getClassName());

}

@Test

public void setClassName() {

String newClassName="RelationTest";

classFormat.setClassName(newClassName);

assertEquals(newClassName,classFormat.getClassName());

}

@Test

public void getMethods() {

String []methods={"test()","main()"};

classFormat.setMethods(Arrays.asList(methods));

assertArrayEquals(methods,classFormat.getMethods().toArray());

}

@Test

public void setMethods() {

String []methods={"test(int a)","main(String[]args)"};

classFormat.setMethods(Arrays.asList(methods));

assertArrayEquals(methods,classFormat.getMethods().toArray());

}

@Test

public void getVariables() {

String []variables={"int i","String className"};

classFormat.setVariables(Arrays.asList(variables));

assertArrayEquals(variables,classFormat.getVariables().toArray());

}

@Test

public void setVariables() {

String []variables={"UMLClassDiagram diagram","String gp"};

classFormat.setVariables(Arrays.asList(variables));

assertArrayEquals(variables,classFormat.getVariables().toArray());

}

}

### 7.2.11 RelationTest

package shapes;

import diagrams.RelationType;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.assertEquals;

import static org.junit.Assert.assertSame;

public class RelationTest {

private ClassFormat startClass,endClass;

private String startClassName ="mainTest",endClassName="main";

private RelationType relationType=RelationType.Composition;

Relation relation;

private int width=100;

private int height=100;

@Before

public void setUp(){

int x = 0;

int y = 0;

startClass =new ConcreteFormat(startClassName, x,y,width,height);

endClass =new ConcreteFormat(endClassName, x,y,width,height);

relation=new Relation(startClass,endClass,relationType);

}

@Test

public void getStartClass() {

assertSame(startClass,relation.getStartClass());

}

@Test

public void getEndClass() {

assertSame(endClass,relation.getEndClass());

}

@Test

public void setStartX() {

int startX=100;

relation.setStartX(startX);

assertEquals(startX,relation.getStartX());

}

@Test

public void setEndX() {

int endX=100;

relation.setEndX(endX);

assertEquals(endX,relation.getEndX());

}

@Test

public void setStartY() {

int startY=100;

relation.setStartY(startY);

assertEquals(startY,relation.getStartY());

}

@Test

public void setEndY() {

int endY=100;

relation.setEndY(endY);

assertEquals(endY,relation.getEndY());

}

@Test

public void setStartClass() {

ClassFormat startClassTest =new ConcreteFormat(startClassName+"\_testdata", 60,80,width,height);

relation.setStartClass(startClassTest);

assertSame(startClassTest,relation.getStartClass());

}

@Test

public void setEndClass() {

ClassFormat endClassTest =new ConcreteFormat(endClassName+"\_testdata", 60,80,width,height);

relation.setEndClass(endClassTest);

assertSame(endClassTest,relation.getEndClass());

}

@Test

public void setRelationType() {

RelationType testType=RelationType.Dependency;

relation.setRelationType(testType);

assertEquals(testType,relation.getRelationType());

}

@Test

public void getRelationType() {

assertEquals(relationType,relation.getRelationType());

}

}

### 7.2.12 ClassFormatOutputDTOTest

package adapter;

import diagrams.RelationType;

import org.junit.Before;

import org.junit.Test;

import shapes.ClassFormat;

import shapes.ConcreteFormat;

import shapes.Relation;

import static org.junit.Assert.\*;

public class RelationOutputDTOTest {

private ClassFormat startClass,endClass;

private String startClassName ="mainTest",endClassName="main";

private RelationType relationType=RelationType.Composition;

Relation relation;

private int width=100;

private int height=100;

@Before

public void setUp(){

int x = 0;

int y = 0;

startClass =new ConcreteFormat(startClassName, x,y,width,height);

endClass =new ConcreteFormat(endClassName, x,y,width,height);

relation=new Relation(startClass,endClass,relationType);

}

@Test

public void getOutput() {

RelationOutputDTO relationOutputDTO=new RelationOutputDTO(relation);

// System.out.println(relationOutputDTO.getOutput());

assertEquals("StartClass:mainTest;EndClass:main;Type:Composition;StartPoint:0,0;EndPoint:0,0",relationOutputDTO.getOutput());

}

}

### 7.2.13 RelationOutputDTOTest

package adapter;

import diagrams.RelationType;

import org.junit.Before;

import org.junit.Test;

import shapes.ClassFormat;

import shapes.ConcreteFormat;

import shapes.Relation;

import static org.junit.Assert.\*;

public class RelationOutputDTOTest {

private ClassFormat startClass,endClass;

private String startClassName ="mainTest",endClassName="main";

private RelationType relationType=RelationType.Composition;

Relation relation;

private int width=100;

private int height=100;

@Before

public void setUp(){

int x = 0;

int y = 0;

startClass =new ConcreteFormat(startClassName, x,y,width,height);

endClass =new ConcreteFormat(endClassName, x,y,width,height);

relation=new Relation(startClass,endClass,relationType);

}

@Test

public void getOutput() {

RelationOutputDTO relationOutputDTO=new RelationOutputDTO(relation);

// System.out.println(relationOutputDTO.getOutput());

assertEquals("StartClass:mainTest;EndClass:main;Type:Composition;StartPoint:0,0;EndPoint:0,0",relationOutputDTO.getOutput());

}

}

### 7.2.14 DiagramWriterTest

package output;

import diagrams.RelationType;

import diagrams.UMLClassDiagram;

import generator.ClassRelationGenerator;

import generator.ClassUnitGenerator;

import org.junit.Before;

import org.junit.Test;

import shapes.ClassFormat;

import shapes.Relation;

import java.util.Arrays;

public class DiagramWriterTest {

private UMLClassDiagram diagram;

private final String[]methods={"testMethod()","main()"},variables={"int tag","String name"};

private final String startClassName ="main",endClassName="UMLClassDiagram";

private ClassFormat startClass,endClass;

private Relation relation;

private final int width=500,height=500;

@Before

public void setUp(){

diagram=new UMLClassDiagram(width,height);

ClassUnitGenerator classUnitGenerator = new ClassUnitGenerator();

ClassRelationGenerator classRelationGenerator = new ClassRelationGenerator();

classUnitGenerator.setClassAttributes(startClassName, Arrays.asList(methods),Arrays.asList(variables));

startClass=classUnitGenerator.generateConcreteClassFormat();

classUnitGenerator.setClassAttributes(endClassName, Arrays.asList(methods),Arrays.asList(variables));

endClass=classUnitGenerator.generateConcreteClassFormat();

relation= classRelationGenerator.generateRelation(startClass,endClass, RelationType.Association);

diagram.addToDiagram(startClass);

diagram.addToDiagram(endClass);

diagram.addToDiagram(relation);

}

@Test

public void save() throws Exception{

String path=System.getProperty("user.dir")+"/tmp.txt";

DiagramWriter diagramWriter =new DiagramWriter(diagram);

diagramWriter.save(path);

DiagramReader diagramReader=new DiagramReader(path);

}

}

# Measurement

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

hw2:2019/02/28 ~ 2019/03/07 Total : 8hrs

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

hw4:2019/04/03 ~ 2019/04/15 Total : 20hrs

hw5:2019/04/26 ~ 2019/04/28 Total : 40hrs

hw6:2019/05/30 ~ 2019/05/10 Total : 40hrs

hw7:2019/05/25 ~ 2019/06/04 Total : 35hrs

**Source Code:** [**https://github.com/JoeLiu1321/OOAD**](https://github.com/JoeLiu1321/OOAD)