綠色新商機 -- 第三方環保杯租借系統

Group 1

資管一 顏晟伍 / 陳昱綸 / 李國濬 / 陳建穎

目錄

1. 主題發想及需求、可行性分析

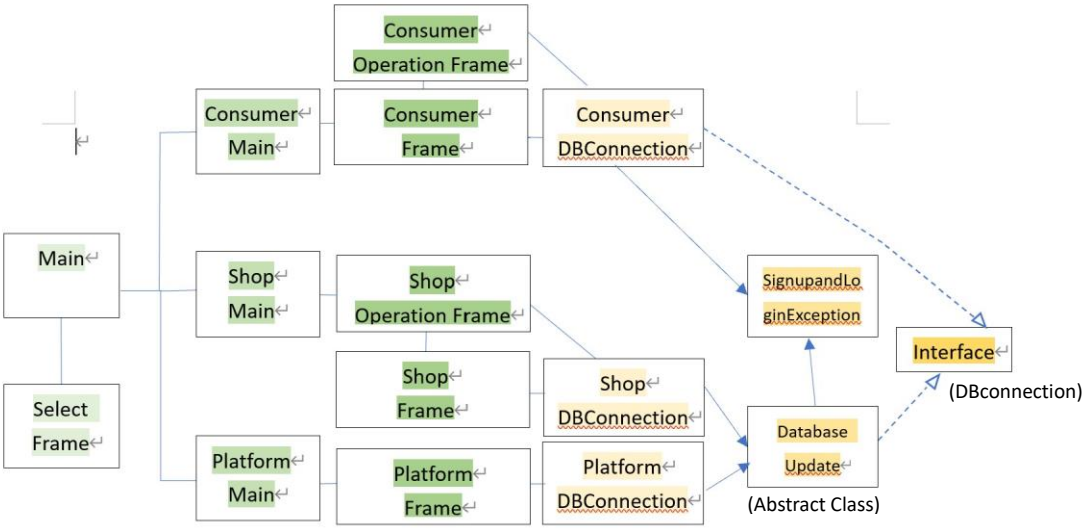
隨著環保意識的逐漸興起，減少一次性塑膠杯、自備環保杯已成為未來的主流趨勢。除此之外，由於一昧指望未來可期的發展趨勢在現在這個時刻並不現實，因此我們也從消費者、政府政策、店家端、以及平台成本端計算，四個不同身分的面向進行我們程式的可行性與市場需求。像是政府推行了一系列政策，例如固定比例需使用循環杯、使用環保杯可減 5 元等，促使消費者更傾向使用能保冷保熱的環保杯。然而，並不是所有人出門都願意攜帶環保杯，因此，我們計劃製作一個環保杯租借系統，讓消費者不用改變原有習慣(不習慣帶環保杯)，同時也讓店家不用更改經營模式(從買一次性杯改成租同樣使用次數的環保杯)。但是鑒於市面上已經有許多類似系統，我們轉而成為第三方，將環保杯租借給各店家，這樣不僅能降低店家大量購買環保杯的風險，也能減少店家提供環保的成本。

2. 與 ESG 和 SDGS 關聯

在 ESG 方面，環境：減少塑膠垃圾、減少對土地和海洋污染、使用環保杯取代一次性塑膠杯，以降低碳排放；政府：符合世界未來趨勢，也呼應政府當今政策；社會：提高民眾環保意識、不犧牲特定一方利益，不一定由店家來吸收狗買環保杯的成本。而在 SDGS 方面，符合了第 11 項永續城鄉、第 12 項責任消費與生產、第 13 項氣候變遷對策，用環保杯來達到永續發展，也藉此對抗氣候變遷。

3. 程式分析

3.1 程式架構



本系統由**三個獨立**的系統構成，分別為平台端 Platform、店家端 Shop 以及消費者端 Consumer 系統。三個系統完全獨立運行，所以**其各自的 Main 皆可作為程式進入點**。只是為展示方便我們將其共同放在一個 Project 裡面，並有個共通的主程式 *Main* 作為程式進入點。

我們的程式在架構上將**前端和後端完全分開設計**。綠色的部分為前端相關的類別，掌管各個系統的 *JFrame* 和其進入點(主程式)；黃色部分為後端相關的類別和介面，掌管各個系統用於連接資料庫的物件。此設計讓各個 Frame 類別在做連接資料庫的動作時**只需透過與其系統相對應的 *DBConnection* 物件**來做操作即可，**不須在 Frame 類別內寫任何 SQL 語句**。

各自系統的 *DBConnection* 皆 implements *DBconnection* 這個 Interface，並繼承「*SignupAndLoginExceptions*」這個 class。不過考慮到平台端和店家端系統有方法重複，顧新增了「Database Update」的**抽象類別**並將相同的方法寫在裡面，讓平台端和店家端的 *DBConnection* 去做繼承。

然後由於 Java 有**單向繼承**的特性，所以將「*SignupAndLoginExceptions*」的繼承放在了「Database Update」，並且也將 Interface「*DBconnection*」 的 implements 宣告在此。然而**該介面的抽象方法並沒有在「*Database Update*」中實作**，而是交給平台端和店家端的 *DBConnection* 去實作。

在系統操作上，若由共通的 *Main* 來啟動程式，則會先進到 *Select Frame* 來選擇要使用的系統。接著，若選**消費者和店家**會先進到登入頁面，可以進行註冊(或激活)以及登入。登入後消費者能查看目前持有杯子，及更換帳號名、密碼；而店家可以借出或接收歸還杯子，以及更改密碼；若選**第三方**會直接進入操作介面，可以查看杯子、消費者、店家等的交易紀錄，此外也可以出借給店家、從店家接收杯子，也可增加新種類的杯子，最後商家的註冊也是在第三方進行操作(店家註冊完需要回到店家頁面進行激活)。以上的動作皆會更新資料庫，或從資料庫抓取資料。至於完整的系統操作介紹則放在操作說明手冊，在此便不逐一詳述。

3.2 概念運用及 Design Pattern

運用了 Interface 和物件的概念來簡化各個頁面登入資料庫的步驟，不需要每個 Frame 裡都再寫一次連線資料庫；而 Abstract Class 除了可以設定架構，也可以將重複的 method 寫好，漸少重複；GUI 方面多運用 *JTextField.getText()*來獲取資訊，並使用 *ActionListener* 來為 *JButton* 增加功能，也用 *JTextArea.setText()*或 *append* 來列出我們要查詢的交易紀錄等等。

我們選擇的 Design Pattern 為 MVC 模式，模型（Model）在各自的 *DBConnection* 中，根據不同功能，在 method 裡設置相關的 SQL 指令，來更新資料庫或從資料庫獲取資料；視圖（View）利用 *JTextArea* 來讓使用者看到他所需要的訊息；控制器（Controller）根據不同 Button 設計不同功能，來處理使用者的需求。

另外，借助科技的幫忙，我們將全部的 Java 檔加上了 Javadoc 格式的註解，並使用 Eclipse 內建的 Javadoc 文件產生器，生成了由 HTML 撰寫的 Javadoc 文件。在文件中可以詳細看到所有程式間的關係和使用方式，內容十分詳盡。

附錄連結：組員自行製作的 javadoc

3.3 程式特點

1. 將 SQL 相關指令都整合在一個 Class 當中並將其**作為物件使用**，當中又利用到 Interface 和 Abstract Class 的概念來**規範方法**和**減少程式碼重複**。

2. 將所有要檢查的地方都做成**自定義 Exception** 並**統一放在一個 Class** 中。這樣作的優點有以下兩點：

   a. 設計方法時明列出可能出現的 Exceptions 且發生就**拋出其相對的 Exception** 物

件，這樣其他人在使用該方法時就能**明確知道可能會有哪些問題**，在維護時也較**方便修改**（和回傳 Boolean 值相比）。另外，還能**保證使用該方法的人必須處理這些問題**。

b. 在我們的程式中，有不同的 SQL 指令 Class 都拋出相同的 Exception 。因此將這些 Exception 都定義在相同的 Class 當中便能讓這些不同的 Class **只要 Import 一個 Class 就能使用這些自定義 Exception** 。這樣一來便不用在各個 Class 中定義相同的 Exception ，**減少程式重複**。

3. 因應多人共同開發一個程式，為**避免成員間可能的命名衝突**，以及明確**分劃出各自的負責內容**，我們使用了「Package」的概念，依照各個 Classes 不同的功能和類型，放在與其相對應的 Package 中。成員在開發時就在其負責的 Package 中撰寫程式即可，無須考慮可能的命名衝突。

4. 分工表

112306001 顏晟伍：	組長、報告、簡報與影片、Platform 程式碼
112306069 陳昱綸：	主要程式撰寫、資料庫建立、程式碼除錯與修改
112306081 李國濬：	簡報、Consumer 程式碼
112306081 陳建穎：	念頭發想、文件文編、Shop 程式碼

細節分工表:	
念頭發想	陳建穎
系統架構+程式整合除錯	陳昱綸
DatabaseConnection 程式碼	陳昱綸、顏晟伍
Platform 程式碼	顏晟伍
Shop 程式碼	陳建穎
Consumer 程式碼	李國濬
簡報	顏晟伍、李國濬
影片+報告	顏晟伍
UML 製圖(3-1 程式架構圖)	李國濬
文書報告(本文件)	陳建穎、陳昱綸(程式分析)
操作說明手冊	陳昱綸