軟體品質保證與軟體測試

期末專題

題目：自動販賣機

組員：余榮泰、張憲忠

組別:T13

1. 內容 (what- function)
   1. 硬幣有三種幣別1元、 5元、 10元。
   2. 販賣機可因為使用者投入的金額，告知使用者那些商品可以購買。
   3. 販賣機有投幣、退幣找零、按鈕購物及顯示物品功能。
   4. 販賣機可賣六種商品。
   5. 分為補貨模式以及購買模式。
   6. 可以告知使用者商品的剩餘數量。
   7. 若金額或是商品數量不足會告知並不能購買。
   8. 提供使用者視窗介面。
   9. 使用Java Swing。　Swing 包含了構建圖形介面（GUI）的各種元件，如: 視窗、標籤、按鈕、文字框等。
   10. 使用Junit測試框架以及EclEmma測試涵蓋度，相互配合可以檢驗還有哪些

敘述還沒有被執行到、測試到。

1. 設計 (how- design)

自動販賣機可以提供

* 1. 選擇販賣機內的商品
  2. 投入零錢
  3. 退幣找零
  4. 獲取商品
  5. 提供販賣機內的商品列表，包括單價、庫存
  6. 並且能對於商品進行補貨
  7. 提供GUI讓使用者明白操作。

1. 程式模組說明 (how- code)

一開始先寫出一個自動販賣機的簡易型雛形程式，並開始進行下一步的修改，為了可以更加清楚程式結果，所以做出圖形使用者介面(GUI)，因為圖形介面對於使用者來說更為簡便易用。

使用Java Swing 圖形介面開發，我們的 Java 的圖形介面，使用按鈕、文字框呈現，再使用ActionEvent 類別處理按鈕被按下的事件，實作ActionListener 介面後，每點一下按鈕，就能做出指定的回應。

為了能為程式提供測試方法，我們將所有的output，也就是把所有的System.out改掉， 我們用到了JUnit的這兩個功能：@Test以及Assert.assertEquals(expected, actual)來驗證expected、actual 是否為相同。

1. 成果 (what- result), 包含畫面的說明



此圖為程式執行後的圖形使用者介面，一開始的預設為購買模式。



再分別投入錢幣1、5、10元後，使用者金額累加為16元。

1. 測試報告 (test)
   1. 測試方法

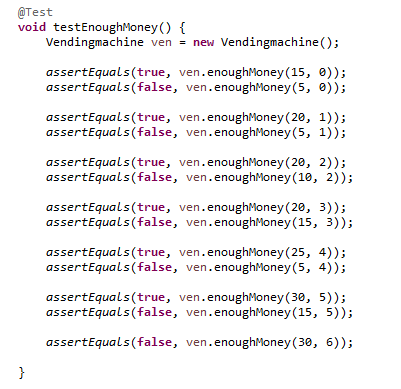
單元測試、整合測試、系統測試

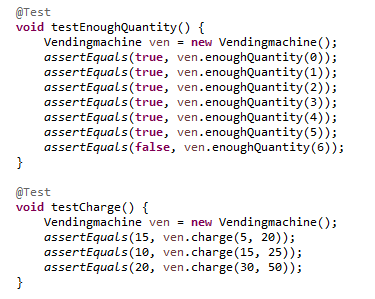
* 1. 測試設計與案例說明

2.1單位測試

**testEnoughMoney(),testEnoughQuantity()**為測試投入金額與商品數量不足時應回傳false

**testCharge()**則為當投入金額>商品金額時，應回傳找零金額

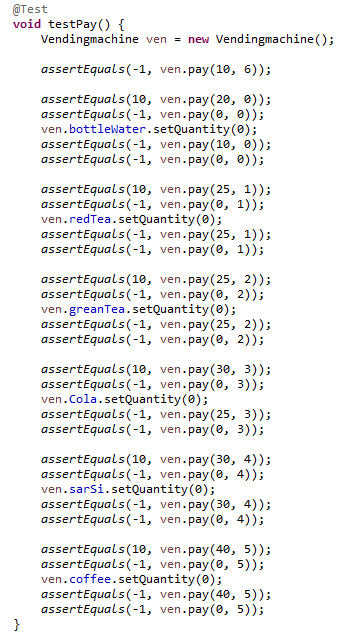




2.2整合測試

testPay()這測試案例中呼叫了veningmachine中的物件，以測試在不同的投入金額以及商品數量下是否能購買商品

testCGameFDollar()為測試GUI中按了button之後，label、狀態是否正確



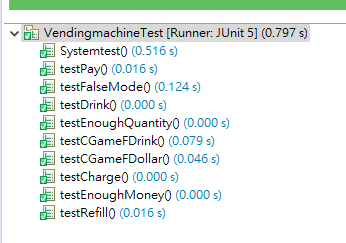


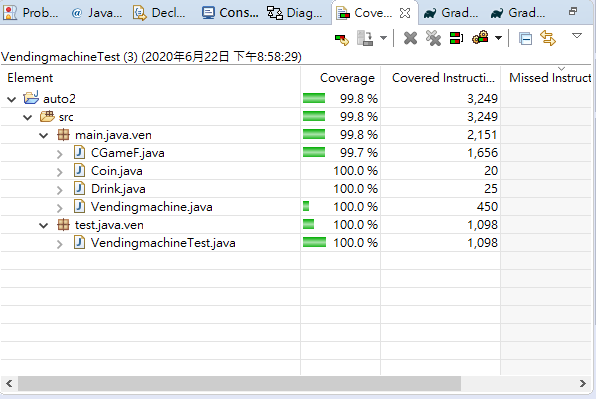
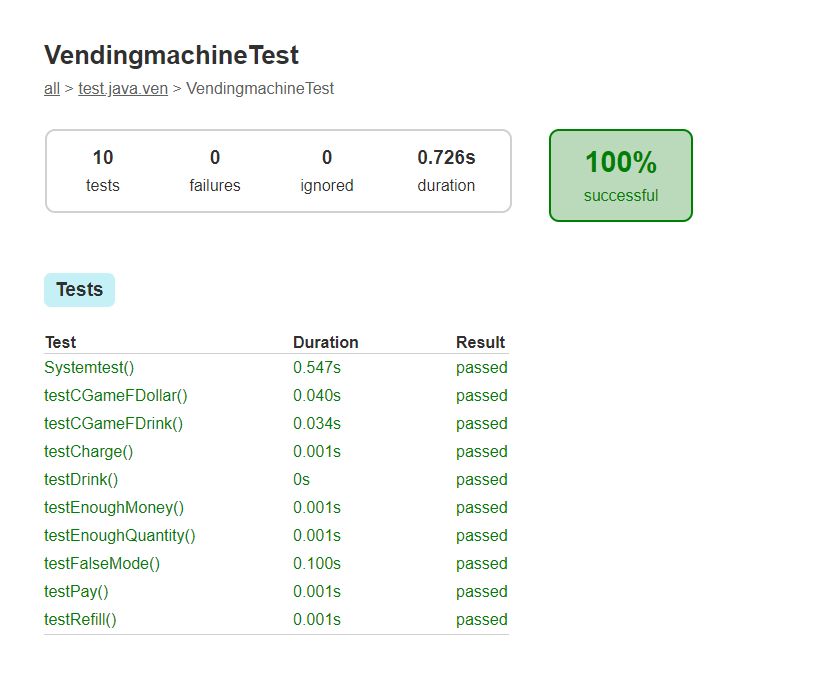
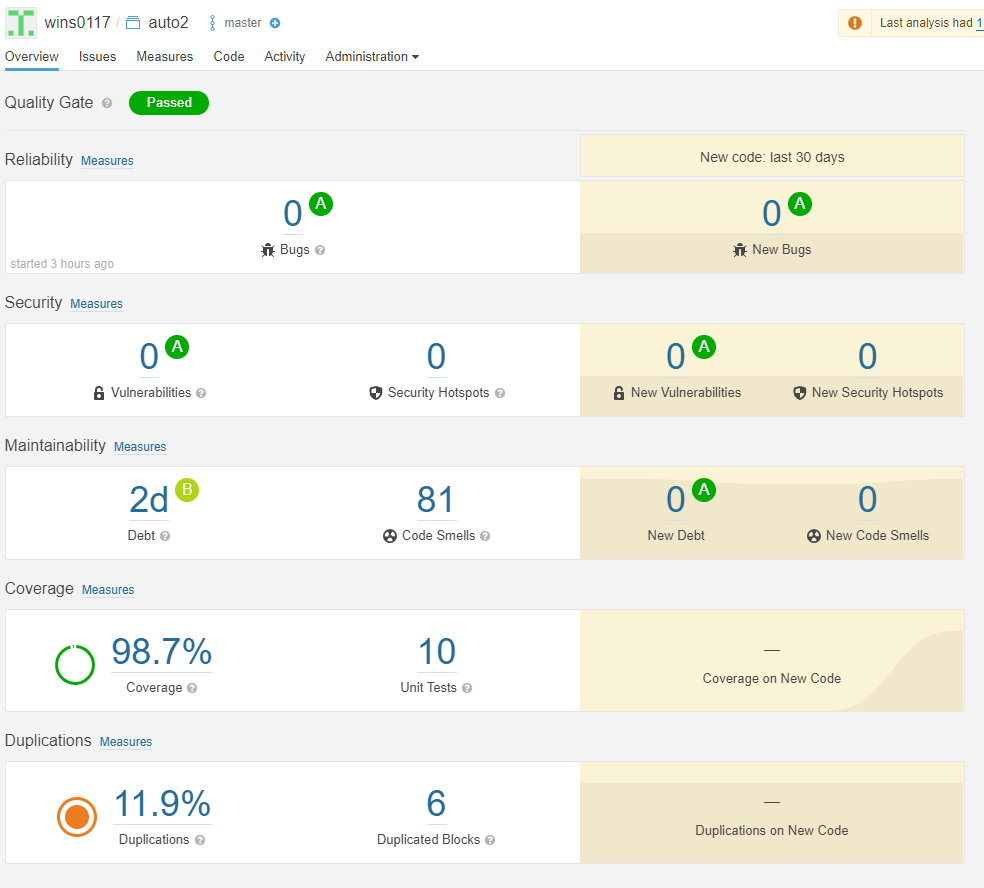
2.3系統測試

**Systemtest()**設計了一個情境，使用者對販賣機一連串的操作下是否販賣機能夠正確的回應



* 1. 測試結果



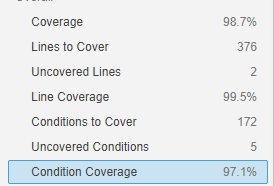
  

1. 自評
   1. 功能的完整性 (包含程式碼行數）

功能皆已完成且無bug



* 1. 測試的完整性 (包含測試案例數，涵蓋度等量化指標）



測試案例數為十項，覆蓋率達98.7%

* 1. 測試的自動化程度 (包含測試碼行數）

藉助了gradle、sonarqube幫助測試，因時間因素無再整合至jenkins中