軟體測試期中考_1

介紹 Introduce

設計一套可以計算當前票價的程式，並加以測試

資料設計 DataDesign

第二欄的設計理念，因為假日、套票、季後賽的價格互不影響，因此把它設計為同一個欄位
Holiday -> 假日, Post -> 季後賽, Package -> 套票

Price	no/isHoliday/isPost/isPackage	isVip
票的價格	票的模式	是否為 Vip

這邊要特別對出現 (套票 + Vip) 進行例外處理

程式設計 Programing

變數的理念

因為票價是固定的，因此將其資料型態統一設計為 `final private static double`
這樣日後票價若出現漲價、降價，也比較好一併處理

```
// 基本的票價
private final static double BASIC_PRICE = 500;
// 假日的票價
private final static double HOLIDAY_PRICE = 600;
// 季後賽的票價
private final static double POST_PRICE = 1000;
// 套票的票價
private final static double PACKAGE_PRICE = 250;
// VIP的票價
private final static double VIP_BONUS = 0.8;
```

程式邏輯 + Preventing Program

對 counter 傳入 mode, isVip 的值
首先判斷是否 mode 介於設計的 0 ~ 3 之間，若沒有則拋出例外

```
if (mode < 0 || mode > 3)
    throw new IllegalAccessException("mode 應該介於 0~3之間");
```

透過 mode 得到一個價格
接著去判斷是否出現 VIP + (mode == 3) 的情況

```
boolean bothVipAndModeThree = isVIP && mode == 3;
if (bothVipAndModeThree)
    throw new IllegalAccessException("不能為套票又使用 VIP");
```

判斷是否為 VIP，若是則打折

先透過 log 輸出本次輸入、輸出

```
log.info("mode {} isVIp {} 最後算出 {}", mode, isVIP, result);
```

最後回傳結果

PMD 的處理

將所有遇上的問題通通解決掉
並且有將原始的程式產生的 PMD 以及修改後的 PMD 各別放上資料夾的第一層
名稱為 originPmd.html, afterPmd.html

放上一些我這之中學習到的知識

- MethodNamingConventions The static method name 'Counter'
  function 的命名方式應該為駱駝式的命名
- UseUtilityClass All methods are static. Consider using a utility class instead.
  這邊因為 function 都是 static 的，因此建議將其設置為一個不可被繼承的 class
  避免在日後我們的程式碼被人呼叫成以下

  ```
  CounterMLB counter = new CounterMLB();
  ```

  期望的是直接呼叫

  ```
  CounterMLB.counter()
  ```

- AvoidLiteralsInIfCondition
  因為我的 mode 設計為 0~3 所以我當時是寫了

  ```
  if (mode == 0) {

  } else if (mode == 1){

  }.....
  ```

  這會造成日後，閱讀程式碼的人還要先理解 0 ~ 3 的含意是甚麼
  因此我把它設計為 switch 並且額外設立一個函數

同時避免掉一個 function 有太多判斷的問題

```
public static double judgePrice(int mode) {
    return switch (mode) {
        case 1 -> HOLIDAY_PRICE;
        case 2 -> POST_PRICE;
        case 3 -> PACKAGE_PRICE;
        default -> BASIC_PRICE;
    };
}
```

測試 Testing

首先透過本課學習到的強涵蓋的知識，設計一個 csv 檔案

Price	no/isHoliday/isPost/isPackage	isVip
500	0	FALSE
600	1	FALSE
1000	2	FALSE
250	3	FALSE
400	0	TRUE
480	1	TRUE
800	2	TRUE

1. 強涵蓋，那這邊若用數學去計算應該為 4*2

```
@ParameterizedTest(name = "票價測試 #{index} - 預期票價:{0}, 票種:{1}, VIP:{2}")
@DisplayName("測試 .csv 內設計的所有正常可能資料")
@CsvFileSource(resources = "/Data.csv", numLinesToSkip = 1)
void counter(double expected, int mode, boolean isVip) throws IllegalAccessEx
    double result = CounterMLB.counter(mode, isVip);
    log.info("預期: {} 測試結果: {}",expected , result);
    assertEquals(expected, result);
}
```

2. 為何 -1？

   因為必須再額外寫一個測試檔案去針對出現 VIP + package 的情況

```
@ParameterizedTest
@DisplayName("測試 套票 + VIP 的組合出現")
@ValueSource(ints = {3})
void counterForCheckVIPAndPackage(int mode) {
    IllegalAccessException illegalAccessException = assertThrows(IllegalAcces
```

```
    log.info("測試拋出接受到的 message {}", illegalAccessException.getMessage())
    assertEquals("不能為套票又使用 VIP", illegalAccessException.getMessage());
}
```

3. 最後再針對是否我們有對 mode 的邊界進行拋出例外的測試

```
@ParameterizedTest
@DisplayName("測試 mode 非規範內的拋出")
@ValueSource(ints = {-1,4,5})
void checkModeInZeroToThree(int mode) {
    IllegalAccessException illegalAccessException = assertThrows(IllegalAcces
    log.info("測試拋出接受到的 message {}", illegalAccessException.getMessage())
    assertEquals("mode 應該介於 0~3之間", illegalAccessException.getMessage());
}
```

結果

自評

(O) 能夠設計豐富而有效的測試案例（以表格規劃之）
(O) 完成程式可以執行，並以 JUnit 測試
(O) 成功完成 PMD 程式靜態檢測的報告
(O) 可以依據 PMD 建議，改善程式碼