Contents

money_spec

[coding_record]

[tdd]

[unittests]

需求

- 相同數值視為等值
- 可提供貨幣單位匯率轉換
- 可以加總、相減、乘法

測試案例

- Money.cs
- CurrencyLookup.cs

將貨幣單位相關部份由 CurrencyLookup(domain service) 來負責，money 僅透過介面去取得 currency ，使 money 物件不需要擔心如何正確建立 currency，該部份會由 CurrencyLookup 操作，money 只需注意取得的 currency 是否 InUse 與傳入的數值是否符合其 currency 的小數點進位

1. example
    1. two_of_same_amount_should_be_equal()
    2. two_of_same_amount_but_different_currencies
    3. fromstring_and_fromdecimal_should_be_equal
    4. sum_of_money_gives_full_amount
    5. unused_currency_should_not_be_allowed
    6. unknown_currency_should_not_be_allowed
    7. throw_when_too_many_decimal_places
    8. throws_on_adding_different_currencies
    9. throws_on_substracting_different_currencies

從閱讀測試案例名稱可得知，規格為：- 值相等性 - 不同貨幣單位但同數值者視為不同 - 提供 string 與 decimal 建立物件 - 不得使用已棄用貨幣單位 - 非法字串無法轉換為貨幣單位 - 每種貨幣單位有其各自的小數點進位 - 不同貨幣單位無法進行相加與相減

2. practice：個人練習拆為 3 部份 money 只處理有關如何建立正確貨幣，有關匯率轉換計算部份由 ExchangeService 負責，Pair 提供建立各種貨幣單位對應，並另外實作不同貨幣可指定單位進行加總
    - Money.cs
    - Currency.cs
    - Pair.cs
    - ICurrencyExpression.cs
    - IOperationExpression.cs
    - ExchangeService.cs

1. moneytests
    1. test_gives_string_should_be_transfer_amount
    2. test_gives_invalid_currency_should_be_throw_exception
    3. test_useless_currency_should_not_be_allowed
    4. test_throw_when_too_many_decimal_places
    5. test_get_currency
    6. test_money_with_same_amount_should_be_equality
2. pairtests
    1. test_pair_equality
    2. test_null_value_should_throw_exception
3. ExchangeServiceTest
    1. test_currency_exchange_to_another_currency
    2. test_add_rate_than_list_should_be_added
    3. test_add_same_pair_and_value_should_be_throw_exception
    4. test_get_rate
    5. test_sum_of_money_gives_full_amount
    6. test_subtraction_of_money_gives_correct_amount
    7. test_currency_times_n_then_return_amount_multiplied_by_n

由以上練習得知，若需求規格無文件化，或是無法與程式碼與時俱進時，測試程式碼的重要性就會突顯，可以透過良好的函式名稱表達受測物件有何限制，可以達到什麼功能，對了解一個系統與商業邏輯上有很大的幫助，且與測試程式與受測物件習習相關，可達到 living documents。

- 應尋求方法可將函式名稱、註解、案例、結果進行文件化，提供非開發人員檢閱測試是否符合需求規範，減少開發人員與測試人員、等其他職能之間對商業邏輯的誤解，促進產生共識。
- 從另一個方向思考，若開始初期即進行協作 example by specification(需求規格實例化)，在未實際開發前即在規格上有共識，並商討測試案例，有助於開發人員的開發速度，有明確規範的指引下利用 tdd 的方式使得功能會大幅減少測試與重工。
- 實驗流程
    1. event storming 了解領域知識，並識別 domain model, domain event, 流程等
    2. design-level storming ，針對以上特定區塊進行細部討論，進行 design modeling 工作
    3. 進行協作 example by specifications，在測試案例與 edge case 有共識，並產生實際規格
    4. 開發人員按照規格進行開發/修改 (tdd)
    5. demo ，並討論是否與實際有落差，或規格有遺漏的部份
    6. 進行 3. 步驟，直到驗收完成。