TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP MÔN HỌC

Kiểm thử và đảm bảo chất lượng phần mềm

BÀI TẬP: KIỂM THỬ CHỨC NĂNG

Sinh viên: Đỗ Hữu Hoàng Tùng

Mã sinh viên: 22026551

1. Đặc tả bài toán

Chương trình mô phỏng đơn giản quá trình giao dịch trong ngân hàng, nhận 3 thành phần chính là đầu vào:

1. balance: Số tiền hiện tại có trong tài khoản
2. transaction_type: Loại giao dịch, có thể nhận 1 trong 2 giá trị:
    a. deposit: Giao dịch bỏ tiền vào tài khoản
    b. withdrawal: Giao dịch rút tiền khỏi tài khoản
3. amount: Số tiền sử dụng vào giao dịch

Tùy vào loại giao dịch, số balance trong tài khoản sẽ được thêm vào hoặc bớt đi 1 lượng tương ứng với amount. Chương trình có 1 số ràng buộc nhất định như sau:

1. amount và balance đều phải là 1 số tự nhiên
2. transaction_type chỉ có thể nhận 1 trong 2 giá trị là deposit và withdrawal
3. Do vấn đề an ninh, chỉ cho phép số lượng amount tối đa trong mỗi giao dịch là 1 triệu
4. Nếu transaction_type là withdrawal thì amount không được lớn hơn balance

2. Cài đặt chương trình

Sử dụng ngôn ngữ Python để viết hàm cho chương trình

```
def bank_transaction(balance: int, transaction_type:str, amount:int) ->
int:
    """
    Perform a bank transaction (deposit or withdrawal) on the current
amount.

    Parameters:
    balance (int): The current amount in the bank account.
    transaction_type (str): The type of transaction ('deposit' or
'withdrawal').
    amount (int): The amount to deposit or withdraw.

    Returns:
    int: The new amount in the bank account after the transaction.
```

```
    """
    if amount < 0:
        raise ValueError("Input not valid.")
    if amount == 0:
        return balance
    if amount > 1000000:
        raise ValueError("Input not valid.")
    if transaction_type == 'deposit':
        balance = balance + amount
    elif transaction_type == 'withdrawal':
        if amount > balance:
            raise ValueError("Input not valid.")
        balance = balance - amount
    else:
        raise ValueError("Input not valid.")
    return balance
```

1. Đoạn code chương trình giao dịch ngân hàng

3. Kiểm thử chức năng

3.1 Kiểm thử giá trị biên

Tại đây ta sử dụng phương pháp kiểm thử giá trị biên thông thường và single boundary với giá trị đầu vào là amount. Từ đặc tả ta biết được amount nằm trong khoảng [0, 1000000] nên ta có được các giá trị sau:

- Giá trị biến amount nhỏ nhất: 0
- Giá trị biến amount cận giá trị nhỏ nhất: 1
- Giá trị biến amount thông thường: 500
- Giá trị biến amount lớn nhất: 1000000
- Giá trị cận biến amount lớn nhất: 999999

Từ phân tích trên, ta thấy khi sử dụng single boundary sẽ có thể lựa chọn 5 khả năng cho biến amount, từ đó sinh được 5 test case tương ứng như sau:

STT	Input	Expected Output
1	balance: 1000 transaction_type: deposit amount: 0	1000
2	balance: 1000 transaction_type: deposit amount: 500	1500
3	balance: 1000 transaction_type: deposit amount: 1000000	1001000
4	balance: 1000 transaction_type: deposit amount: 1	1001
5	balance: 1000 transaction_type: deposit amount: 999999	1000999

Bảng 1. Bảng kiểm thử theo giá trị biên

```
class TestBankTransactionBoundaryMethod(unittest.TestCase):
    def test_1(self):
        self.assertEqual(bank_transaction(1000, 'deposit', 0), 1000)

    def test_2(self):
        self.assertEqual(bank_transaction(1000, 'deposit', 500), 1500)

    def test_3(self):
        self.assertEqual(bank_transaction(1000, 'deposit',  1000000),
1001000)

    def test_4(self):
        self.assertEqual(bank_transaction(1000, 'deposit', 1), 1001)

    def test_5(self):
        self.assertEqual(bank_transaction(1000, 'deposit', 999999),
1000999)
```

2. Đoạn code xây dựng các ca kiểm thử theo giá trị biên

3.2 Kiểm thử theo bảng quyết định

Dựa vào đặc tả, ta xác định được các điều kiện và hành động sau

- Điều kiện:
  - C1: balance < 0?
  - C2: transaction_type là deposit?
  - C3: transaction_type là withdrawal?
  - C4: amount > 0?
  - C5: amount > balance?
  - C6: amount > 1000000?
- Hành động:
  - E1: Input không hợp lệ
  - E2: Thực hiện giao dịch bỏ tiền vào tài khoản
  - E3: Thực hiện giao dịch rút tiền

Dựa vào các điều kiện và hành động xác định ở trên, ta lập được bảng quyết định dưới đây

		R1	R2	R3	R4	R5	R6	R7	R8	R9
Điều kiện	C1: balance < 0	T	F	F	F	F	F	F	F	F
	C2: Transaction_type là deposit	T	T	T	T	T	F	F	F	F
	C3: Transaction_type là withdrawal	_	_	_	_	_	T	T	T	F
	C4: amount < 0	F	T	F	F	F	F	F	F	F
	C5: amount > balance	_	_	T	T	F	T	T	F	F
	C6: amount > 1000000	_	_	T	F	F	T	F	F	F
Hành động	E1: Input không hợp lệ	v	v	v			v	v		v
	E2: Thực hiện giao dịch bỏ tiền vào tài khoản				v	v				
	E3: Thực hiện giao dịch rút tiền								v	

Bảng 2. Bảng quyết định

Trong bảng 2 có 9 cột từ R1 đến R9, tương đương với đó là 9 ca kiểm thử được tạo ra dựa trên bảng quyết định:

STT	Input	Expected Output
1	balance: -10 transaction_type: deposit amount: 500	Raise Value Error
2	balance: 10 transaction_type: deposit amount: -500	Raise Value Error
3	balance: 10 transaction_type: deposit amount: 1000001	Raise Value Error
4	balance: 1000 transaction_type: deposit amount: 5000	6000
5	balance: 1000 transaction_type: deposit amount: 500	1500
6	balance: 1000 transaction_type: withdrawal amount: 1000001	Raise Value Error
7	balance: 1000 transaction_type: withdrawal amount: 1500	Raise Value Error
8	balance: 1000 transaction_type: withdrawal amount: 500	500
9	balance: 1000 transaction_type: transfer	Raise Value Error

	amount: 500	

Bảng 3. Bảng kiểm thử theo bảng quyết định

```
class TestBankTransactionDecisionTable(unittest.TestCase):
                        def test_1(self):
                    with self.assertRaises(ValueError):
                     bank_transaction(-10, 'deposit', 500)

                        def test_2(self):
                    with self.assertRaises(ValueError):
                     bank_transaction(10, 'deposit', -500)

                        def test_3(self):
                    with self.assertRaises(ValueError):
                   bank_transaction(10, 'deposit', 1000001)

                        def test_4(self):
      self.assertEqual(bank_transaction(1000, 'deposit', 5000), 6000)

                        def test_5(self):
       self.assertEqual(bank_transaction(1000, 'deposit', 500), 1500)

                        def test_6(self):
                    with self.assertRaises(ValueError):
                bank_transaction(1000, 'withdrawal', 1000001)

                        def test_7(self):
                    with self.assertRaises(ValueError):
                  bank_transaction(1000, 'withdrawal', 1500)

                        def test_8(self):
        self.assertEqual(bank_transaction(1000, 'withdrawal', 500),
                                 500)

                        def test_9(self):
                    with self.assertRaises(ValueError):
                    bank_transaction(1000, 'transfer', 500)
```

2. Đoạn code xây dựng các ca kiểm thử theo bảng quyết định

3.3 Kết quả kiểm thử

Sử dụng công cụ kiểm thử được tích hợp sẵn Pyunit của Python để tạo và chạy các ca kiểm thử vừa được sinh ra

```
.....F........
======================================================================
FAIL: test_1 (test.TestBankTransactionDecisionTable.test_1)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\ADMIN\Documents\code\KTPM\test.py", line 22, in test_1
    with self.assertRaises(ValueError):
AssertionError: ValueError not raised

----------------------------------------------------------------------
Ran 14 tests in 0.029s
```

Hình 1. Kết quả kiểm thử

STT	Input	Expected Output	Actual Output	Status
1	balance: 1000 transaction_type: deposit amount: 0	1000	1000	Passed
2	balance: 1000 transaction_type: deposit amount: 500	1500	1500	Passed
3	balance: 1000 transaction_type: deposit amount: 1000000	1001000	1001000	Passed
4	balance: 1000 transaction_type: deposit amount: 1	1001	1001	Passed
5	balance: 1000 transaction_type: deposit amount: 999999	1000999	1000999	Passed

Bảng 4. Bảng báo cáo kiểm thử theo giá trị biên V1

STT	Input	Expected Output	Actual Output	Status
1	balance: -10 transaction_type: deposit amount: 500	Raise Value Error	490	Failed
2	balance: 10 transaction_type: deposit amount: -500	Raise Value Error	Raise Value Error	Passed
3	balance: 10 transaction_type: deposit amount: 1000001	Raise Value Error	Raise Value Error	Passed
4	balance: 1000 transaction_type: deposit amount: 5000	6000	6000	Passed
5	balance: 1000 transaction_type: deposit amount: 500	1500	1500	Passed
6	balance: 1000 transaction_type: withdrawal amount: 1000001	Raise Value Error	Raise Value Error	Passed
7	balance: 1000 transaction_type: withdrawal amount: 1500	Raise Value Error	Raise Value Error	Passed
8	balance: 1000 transaction_type: withdrawal amount: 500	500	500	Passed
9	balance: 1000	Raise Value	Raise Value	Passed

	transaction_type: transfer amount: 500	Error	Error	

Bảng 5. Bảng báo cáo kiểm thử theo bảng quyết định V1

4. Nhận xét và kết luận

Từ kết quả kiểm thử ở trên, ta thấy với kiểm thử giá trị biên toàn bộ ca kiểm thử đều qua, tuy nhiên do mới chỉ sử dụng single boundary và chưa sử dụng thêm các giá trị biên mạnh nên khả năng xác định lỗi sai tiềm ẩn trong chương trình còn hạn chế. Với phương pháp bảng quyết định, đã có 1 ca kiểm thử không được qua chính là ca đầu tiên với trường hợp balance < 0, sau khi phân tích thì nguyên nhân nằm ở logic chương trình còn thiếu sót, cụ thể chưa có đoạn mã kiểm tra giá trị balance dẫn đến sai sót như trên. Sau khi chỉnh sửa và chạy lại kiểm thử thì toàn bộ ca kiểm thử đều đã thành công như 2 bảng dưới đây

```
if amount < 0 or balance < 0:
        raise ValueError("Input not valid.")
```

3. Thêm logic kiểm tra giá trị balance

STT	Input	Expected Output	Actual Output	Status
1	balance: 1000 transaction_type: deposit amount: 0	1000	1000	Passed
2	balance: 1000 transaction_type: deposit amount: 500	1500	1500	Passed
3	balance: 1000 transaction_type: deposit amount: 1000000	1001000	1001000	Passed
4	balance: 1000 transaction_type: deposit	1001	1001	Passed

	amount: 1			
5	balance: 1000 transaction_type: deposit amount: 999999	1000999	1000999	Passed

Bảng 4. Bảng báo cáo kiểm thử theo giá trị biên V2

STT	Input	Expected Output	Actual Output	Status
1	balance: -10 transaction_type: deposit amount: 500	Raise Value Error	Raise Value Error	Passed
2	balance: 10 transaction_type: deposit amount: -500	Raise Value Error	Raise Value Error	Passed
3	balance: 10 transaction_type: deposit amount: 1000001	Raise Value Error	Raise Value Error	Passed
4	balance: 1000 transaction_type: deposit amount: 5000	6000	6000	Passed
5	balance: 1000 transaction_type: deposit amount: 500	1500	1500	Passed
6	balance: 1000 transaction_type: withdrawal amount: 1000001	Raise Value Error	Raise Value Error	Passed
7	balance: 1000 transaction_type: withdrawal amount: 1500	Raise Value Error	Raise Value Error	Passed

8	balance: 1000 transaction_type: withdrawal amount: 500	500	500	Passed
9	balance: 1000 transaction_type: transfer amount: 500	Raise Value Error	Raise Value Error	Passed

Bảng 7. Bảng báo cáo kiểm thử theo bảng quyết định V2

Để đảm bảo phát hiện lỗi bằng kiểm thử giá trị biên, cần áp dụng kiểm thử multiple boundary, nhưng điều này làm tăng đáng kể số lượng ca kiểm thử so với bảng quyết định. Vì vậy, việc kết hợp nhiều phương pháp kiểm thử giúp tối ưu hóa quá trình: ngay cả khi chỉ áp dụng kiểm thử giá trị biên đơn giản, vẫn có thể giảm thiểu số lượng ca kiểm thử mà không lo bỏ sót lỗi.