

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



Báo cáo môn học:
Kiểm thử và đảm bảo chất lượng phần mềm

Chủ đề: Kiểm thử chức năng

Giảng viên: ThS. Nguyễn Thu Trang

Mã lớp môn học: INT3117 7

Sinh viên: Nguyễn Hữu Đồng

Mã sinh viên: 21020760

Lớp: K66I-IT1

Hà Nội – 2024

Mục lục

Mục lục	iii
1 Mô tả yêu cầu	1
1.1 Mô tả bài toán	1
1.2 Input	1
1.3 Output	1
2 Phân tích và sinh ca kiểm thử	2
2.1 Phân tích yêu cầu	2
2.2 Sinh ca kiểm thử - Kiểm thử biên	2
2.2.1 Phân tích	2
2.2.2 Ca kiểm thử	2
2.3 Sinh ca kiểm thử - Bảng quyết định	3
2.3.1 Phân tích	3
2.3.2 Bảng quyết định	3
2.3.3 Ca kiểm thử	3
3 Mã nguồn	6
4 Phân tích kết quả	10
4.1 Kiểm thử biên	10
4.2 Kiểm thử với bảng quyết định	10
4.3 Phân tích lỗi	10
4.4 Phân tích kết quả	10

Chương 1

Mô tả yêu cầu

1.1 Mô tả bài toán

Bài toán: Tính số tiền cần phải thanh toán cho dịch vụ taxi. Công ty X cung cấp dịch vụ taxi với mức giá cụ thể như sau:

- Phí mở cửa: 30,000 đồng cho quãng đường ban đầu (dưới 1km)
- Từ km thứ 2 đến km thứ 20: 20,000 đồng cho mỗi km tiếp theo
- Từ km thứ 21 trở đi: 10,000 đồng cho mỗi km
- Mỗi người thêm vào (từ người thứ 2) sẽ tăng thêm 20% vào tổng chi phí cơ bản của 1 người.

Ngoài ra, do giới hạn của phương tiện và để đảm bảo chất lượng phục vụ, dịch vụ taxi của công ty X chỉ vận chuyển tối đa 6 người trên xe (bao gồm cả tài xế), và giới hạn quãng đường di chuyển là 200km.

1.2 Input

Đầu vào của bài toán gồm 1 số nguyên là số người và 1 số thực là số km di chuyển.

1.3 Output

Cho biết số tiền mà nhóm người cần phải trả cho chuyến xe.

Chương 2

Phân tích và sinh ca kiểm thử

2.1 Phân tích yêu cầu

Do biến đầu vào của bài toán là số người trên xe và quãng đường đi được, trong đó số người trên xe đã được bài toán giới hạn là 6 (bao gồm cả tài xế), và quãng đường đi được có thể chấp nhận là 0 (giá mở cửa). Do đó, các biến đầu vào có ràng buộc lần lượt là:

$$1 \leq \text{passengers} \leq 5$$

$$0 \leq \text{distance} \leq 200$$

2.2 Sinh ca kiểm thử - Kiểm thử biên

2.2.1 Phân tích

Với $1 \leq \text{passengers} \leq 5$ ta có: $\min_{\text{passengers}} = 1$, $\min_{+\text{passengers}} = 2$, $\text{norm}_{\text{passengers}} = 3$, $\text{max}_{-\text{passengers}} = 4$, $\text{max}_{\text{passengers}} = 5$

Với $0 \leq \text{distance} \leq 200$ ta có: $\min_{\text{distance}} = 0$, $\min_{+\text{distance}} = 0.1$, $\text{norm}_{\text{distance}} = 100$, $\text{max}_{-\text{distance}} = 199.9$, $\text{max}_{\text{distance}} = 200$

2.2.2 Ca kiểm thử

Từ phân tích trên, ta chọn các ca kiểm thử sau:

Bảng 2.1: Test case cho kiểm thử biên

Test case	Passengers	Distance (km)	Expected Output
TC1	1	100	1,210,000
TC2	2	100	1,452,000
TC3	3	100	1,694,000
TC4	4	100	1,936,000
TC5	5	100	2,178,000
TC6	3	0	42,000
TC7	3	0.1	42,000
TC8	3	199.9	3,092,600
TC9	3	200	3,094,000

2.3 Sinh ca kiểm thử - Bảng quyết định

2.3.1 Phân tích

Với $1 \leq passengers \leq 5$ ta có: $passengers \in (-\infty, 0] \cup \{1\} \cup (1, 5] \cup (5, \infty]$

Với $0 \leq distance \leq 200$ ta có: $distance \in (-\infty, 0) \cup [0, 1] \cup (1, 200] \cup (200, \infty)$

2.3.2 Bảng quyết định

Từ những phân tích trên, ta có bảng quyết định sau:

2.3.3 Ca kiểm thử

Từ phân tích trên, ta chọn các ca kiểm thử sau:

Bảng 2.2: Bảng quyết định (1)

		R01	R02	R03	R04	R05	R06
Điều kiện	C1: passengers ≤ 0	F	F	F	F	F	F
	C2: passengers = 1	-	T	T	T	T	T
	C3: $1 < \text{passengers} \leq 5$	-	-	-	-	-	-
	C4: passengers > 5	-	-	-	-	-	-
	C5: distance < 0	F	T	F	F	F	F
	C6: $0 \leq \text{distance} \leq 1$	F	-	T	F	F	F
	C7: $1 < \text{distance} \leq 20$	F	-	-	T	F	F
	C8: $20 < \text{distance} \leq 200$	T	-	-	-	T	F
	C9: distance > 200	-	-	-	-	-	T
Hành động	E1: Tính giá mở cửa			x	x	x	
	E2: Tính giá từ 1km đến 20km				x	x	
	E3: Tính giá từ 20km					x	
	E4: Tính phụ phí thêm người						
	E5: Exception	x	x				x

Bảng 2.3: Bảng quyết định (2)

		R07	R08	R09	R10	R11	R12
Điều kiện	C1: passengers ≤ 0	F	F	F	F	F	F
	C2: passengers = 1	F	F	F	F	F	F
	C3: $1 < \text{passengers} \leq 5$	F	T	T	T	T	T
	C4: passengers > 5	T	-	-	-	-	-
	C5: distance < 0	F	T	F	F	F	F
	C6: $0 \leq \text{distance} \leq 1$	F	-	T	F	F	F
	C7: $1 < \text{distance} \leq 20$	F	-	-	T	F	F
	C8: $20 < \text{distance} \leq 200$	T	-	-	-	T	F
	C9: distance > 200	-	-	-	-	-	T
Hành động	E1: Tính giá mở cửa			x	x	x	
	E2: Tính giá từ 1km đến 20km				x	x	
	E3: Tính giá từ 20km					x	
	E4: Tính phụ phí thêm người			x	x	x	
	E5: Exception	x	x				x

Bảng 2.4: Test case cho bảng quyết định

Test case	Passengers	Distance (km)	Expected Output
TC1	-1	100	Exception
TC2	1	-1	Exception
TC3	1	0	30,000
TC4	1	10	210,000
TC5	1	100	1,210,000
TC6	1	300	Exception
TC7	6	100	Exception
TC8	2	-1	Exception
TC9	2	0	36,000
TC10	2	10	252,000
TC11	2	100	1,452,000
TC12	2	300	Exception

Chương 3

Mã nguồn

Dưới đây là mã nguồn giải bài toán, mã nguồn kiểm thử hộp đen và kết quả kiểm thử. Chi tiết mã nguồn có sẵn tại: https://github.com/huudong03uet/Testing/tree/main/black_box_testing

```
PS G:\Code\Testing\Code\black_box_testing> pytest -v
===== test session starts =====
platform win32 -- Python 3.10.5, pytest-8.3.3, pluggy-1.5.0 -- C:\Users\ADMIN\AppData\Local\Programs\Python\Python310\python.exe
cachedir: .pytest_cache
rootdir: G:\Code\Testing\Code\black_box_testing
plugins: anyio-3.6.1, hydra-core-1.3.2, typeguard-2.13.3
collected 21 items

test_taxi_fare.py::TestTaxiFareCalculator::test_tc1 PASSED [ 4%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc2 PASSED [ 9%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc3 PASSED [ 14%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc4 PASSED [ 19%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc5 PASSED [ 23%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc6 PASSED [ 28%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc7 PASSED [ 33%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc8 PASSED [ 38%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc9 PASSED [ 42%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc10 PASSED [ 47%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc11 PASSED [ 52%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc12 PASSED [ 57%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc13 PASSED [ 61%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc14 PASSED [ 66%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc15 PASSED [ 71%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc16 FAILED [ 76%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc17 PASSED [ 80%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc18 PASSED [ 85%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc19 PASSED [ 90%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc20 PASSED [ 95%]
test_taxi_fare.py::TestTaxiFareCalculator::test_tc21 PASSED [100%]

===== FAILURES =====
----- TestTaxiFareCalculator.test_tc16 -----

self = <test_taxi_fare.TestTaxiFareCalculator object at 0x00000122531CC700>

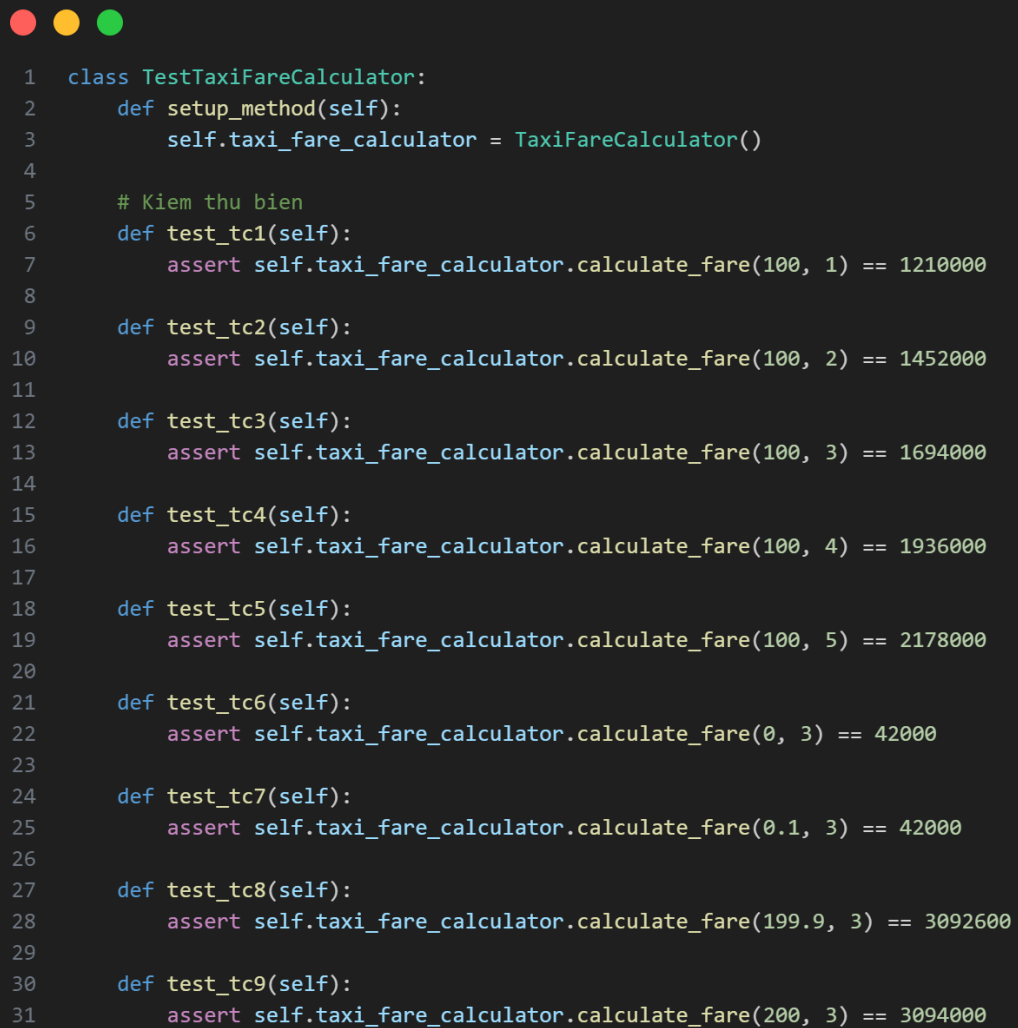
    def test_tc16(self):
>       with pytest.raises(ValueError):
E       Failed: DID NOT RAISE <class 'ValueError'>

test_taxi_fare.py:59: Failed
===== short test summary info =====
FAILED test_taxi_fare.py::TestTaxiFareCalculator::test_tc16 - Failed: DID NOT RAISE <class 'ValueError'>
===== 1 failed, 20 passed in 0.14s =====
```

Hình 3.1: Kết quả kiểm thử



```
1 class TaxiFareCalculator:
2     def __init__(self):
3         self.opening_fare = 30000
4         self.fare_up_to_20km = 20000
5         self.fare_above_21km = 10000
6
7     def calculate_fare(self, distance: float, num_people: int) -> float:
8         if distance < 0 or distance > 200:
9             raise ValueError("Khoảng cách phải từ 0 đến 200 km")
10
11         if num_people < 1 or num_people > 6:
12             raise ValueError("Số lượng người phải lớn hơn 1 và dưới 6 người")
13
14         fare_for_one = self.opening_fare
15
16         if distance > 1:
17             if distance <= 20:
18                 fare_for_one += (distance - 1) * self.fare_up_to_20km
19             else:
20                 fare_for_one += 19 * self.fare_up_to_20km
21                 fare_for_one += (distance - 20) * self.fare_above_21km
22
23         fare_taxi = 0
24
25         if num_people > 1:
26             fare_taxi = fare_for_one + fare_for_one * 0.2 * (num_people - 1)
27         else:
28             fare_taxi = fare_for_one
29         return fare_taxi
```

Hình 3.2: Mã nguồn giải quyết bài toán



```
1 class TestTaxiFareCalculator:
2     def setup_method(self):
3         self.taxi_fare_calculator = TaxiFareCalculator()
4
5     # Kiểm thử biến
6     def test_tc1(self):
7         assert self.taxi_fare_calculator.calculate_fare(100, 1) == 1210000
8
9     def test_tc2(self):
10        assert self.taxi_fare_calculator.calculate_fare(100, 2) == 1452000
11
12    def test_tc3(self):
13        assert self.taxi_fare_calculator.calculate_fare(100, 3) == 1694000
14
15    def test_tc4(self):
16        assert self.taxi_fare_calculator.calculate_fare(100, 4) == 1936000
17
18    def test_tc5(self):
19        assert self.taxi_fare_calculator.calculate_fare(100, 5) == 2178000
20
21    def test_tc6(self):
22        assert self.taxi_fare_calculator.calculate_fare(0, 3) == 42000
23
24    def test_tc7(self):
25        assert self.taxi_fare_calculator.calculate_fare(0.1, 3) == 42000
26
27    def test_tc8(self):
28        assert self.taxi_fare_calculator.calculate_fare(199.9, 3) == 3092600
29
30    def test_tc9(self):
31        assert self.taxi_fare_calculator.calculate_fare(200, 3) == 3094000
```

Hình 3.3: Mã nguồn kiểm thử bài toán (1)



```
1 # Kiem thu bang quyet dinh
2 def test_tc10(self):
3     with pytest.raises(ValueError):
4         self.taxi_fare_calculator.calculate_fare(100, -1)
5 def test_tc11(self):
6     with pytest.raises(ValueError):
7         self.taxi_fare_calculator.calculate_fare(-1, 1)
8
9 def test_tc12(self):
10     assert self.taxi_fare_calculator.calculate_fare(0, 1) == 30000
11
12 def test_tc13(self):
13     assert self.taxi_fare_calculator.calculate_fare(10, 1) == 210000
14
15 def test_tc14(self):
16     assert self.taxi_fare_calculator.calculate_fare(100, 1) == 1210000
17
18 def test_tc15(self):
19     with pytest.raises(ValueError):
20         self.taxi_fare_calculator.calculate_fare(300, 1)
21
22 def test_tc16(self):
23     with pytest.raises(ValueError):
24         self.taxi_fare_calculator.calculate_fare(100, 6)
25
26 def test_tc17(self):
27     with pytest.raises(ValueError):
28         self.taxi_fare_calculator.calculate_fare(-1, 2)
29
30 def test_tc18(self):
31     assert self.taxi_fare_calculator.calculate_fare(0, 2) == 36000
32
33 def test_tc19(self):
34     assert self.taxi_fare_calculator.calculate_fare(10, 2) == 252000
35
36 def test_tc20(self):
37     assert self.taxi_fare_calculator.calculate_fare(100, 2) == 1452000
38
39 def test_tc21(self):
40     with pytest.raises(ValueError):
41         self.taxi_fare_calculator.calculate_fare(300, 2)
```

Hình 3.4: Mã nguồn kiểm thử bài toán (2)

Chương 4

Phân tích kết quả

4.1 Kiểm thử biên

Chương trình pass hết 9 test case và không gây ra lỗi. Ta kết luận rằng chương trình không gặp lỗi với bộ test case này.

4.2 Kiểm thử với bảng quyết định

Đối với kiểm thử với bảng quyết định, chương trình xuất hiện failure tại test case 16. Trong trường hợp này, expected output là Exception ValueError, nhưng chương trình lại không trả ra được Exception.

4.3 Phân tích lỗi

Sau khi kiểm tra lại, cho thấy lập trình viên đã xử lý thiếu trường hợp. Đề bài giới hạn số người trên xe là 6 (bao gồm cả tài xế), tức số người trên xe tối đa là 5. Tuy nhiên, mã nguồn được lập trình $num_people > 6$, điều này đã gây ra lỗi cho chương trình.

4.4 Phân tích kết quả

Ta thấy, mặc dù chương trình pass hết các test đối với Kiểm thử biên, tuy nhiên lại không pass hết các test đối với Kiểm thử với bảng quyết định. Điều đó cho thấy trong trường hợp này, bộ test case của Kiểm thử biên chưa tốt, ta có thể giải quyết bằng cách sử dụng Kiểm thử biên mạnh (thêm 2 giá trị là $max+$ và $min-$) hoặc sử dụng Kiểm thử phân hoạch tương đương để có thể phát hiện được lỗi.