Bài tập tuần 2

Phạm Văn Phúc
21020782@vnu.edu.vn

Giới thiệu

Bài toán xác định liệu khách hàng có khả năng mua xe hơi hay không? Để đơn giản, ta chỉ xét hai yếu tố là tuổi và thu nhập. Dựa vào số liệu thống kê, ta có bảng các trường hợp bên dưới.

Tuổi	Thu nhập	Kết luận
<0	Bất kỳ	Lỗi
0-17	Bất kỳ	Không
18-100	Dưới 25 triệu	Không
18-100	25-50 triệu	Có
18-100	Trên 50 triệu	Có
> 100	Bất kỳ	Lỗi
Bất kỳ	< 0	Lỗi

Bảng 1: Bảng luật lệ

Ta tượng trưng bài toán bằng 1 hàm:

```go
type Customer struct {
  Name   string
  Age    uint8
  Salary uint64
}

func CanBuyCar(customer Customer) (bool, error) {

  return true, nil
}
```

Chương trình 1: Hàm cần kiểm thử

Xây dựng bảng quyết định

Điều kiện và hành động

- age < 0 ? Không thể xảy ra với kiểu tĩnh `uint8`
- salary < 0 ? Không thể xảy ra với `uint64`
- $0 \leq$ age < 18 ?
- $18 \leq$ age ≤ 100 ?
- age > 100 ?
- salary $< 25.000.000$?
- $25.000.000 \leq$ salary $\leq 50.000.000$?

- salary $>$ 50.000.000 ?

- Buy car?

Bảng quyết định

$0 \leq a < 18$	$18 \leq a \leq 100$	$a > 100$	$s < 25m$	$25m \leq s \leq 50m$	$s > 50m$	**Buy car**
T	F	F	-	-	-	**F**
F	T	F	T	F	F	**F**
F	T	F	F	T	F	**T**
F	T	F	F	F	T	**T**
F	F	T	-	-	-	**Error**

Bảng 2: Bảng quyết định (s: salary, a: age)

Từ bảng quyết định, ta có được các test case sau (đính kèm báo cáo):

```
func TestCanBuyCar(t *testing.T) {
  testCases := []struct {
    Name           string
    Age            uint8
    Salary         uint64
    ExpectedResult bool
    ExpectedError  error
  }{
    // Test case 1: 0 <= Age < 18, Salary < 25m, ExpectedResult: false, ExpectedError:
nil
     {Name: "John", Age: 10, Salary: 20000000, ExpectedResult: false, ExpectedError:
nil},

   // Test case 2: 18 <= Age <= 100, Salary < 25m, ExpectedResult: false, ExpectedError:
nil
     {Name: "Alice", Age: 25, Salary: 15000000, ExpectedResult: false, ExpectedError:
nil},

    // Test case 3: Age > 100, Salary < 25m, ExpectedResult: false, ExpectedError: nil
    {Name: "Elderly", Age: 110, Salary: 20000000, ExpectedResult: false, ExpectedError:
InvalidAgeError},

     // Test case 4: 0 <= Age < 18, 25m <= Salary <= 50m, ExpectedResult: false,
ExpectedError: nil
   {Name: "Young Rich", Age: 15, Salary: 35000000, ExpectedResult: false, ExpectedError:
nil},

     // Test case 5: 18 <= Age <= 100, 25m <= Salary <= 50m, ExpectedResult: false,
ExpectedError: nil
    {Name: "Adult Rich", Age: 45, Salary: 45000000, ExpectedResult: true, ExpectedError:
nil},

   // Test case 6: Age > 100, 25m <= Salary <= 50m, ExpectedResult: false, ExpectedError:
nil
       {Name: "Elderly Rich", Age: 105, Salary: 48000000, ExpectedResult: false,
ExpectedError: InvalidAgeError},
```

```
    // Test case 7: 0 <= Age < 18, Salary > 50m, ExpectedResult: true, ExpectedError:
nil
     {Name: "Young Millionaire", Age: 12, Salary: 75000000, ExpectedResult: false,
ExpectedError: nil},
  }
```

Phân hoạch tương đương

Phân tích Chương trình 1, ta thấy có hai đầu vào là `age` và `salary`. Dựa vào bảng Bảng 1, ta thấy có thể chia thành các lớp sau:

`age`: [0, 18); [18, 100], [100, $+\infty$]

`salary`: [$-\infty$, 25m); [25m,50m]; [50m, ∞]

Từ các miền đã phân lớp, ta có phân hoạch mạnh sau:

```
func TestCanBuyCarEquivalent(t *testing.T) {
  testCases := []struct {
    Name           string
    Age            uint8
    Salary         uint64
    ExpectedResult bool
    ExpectedError  error
  }{
    // Test case 1: Age in [0, 18), Salary in [-infinity, 25m), ExpectedResult: false,
ExpectedError: nil
     {Name: "Teenager Low Income", Age: 10, Salary: 15000000, ExpectedResult: false,
ExpectedError: nil},

     // Test case 2: Age in [0, 18), Salary in [25m, 50m], ExpectedResult: false,
ExpectedError: nil
    {Name: "Teenager Middle Income", Age: 17, Salary: 35000000, ExpectedResult: false,
ExpectedError: nil},

    // Test case 3: Age in [0, 18), Salary in [50m, infinity), ExpectedResult: true,
ExpectedError: nil
    {Name: "Teenager High Income", Age: 16, Salary: 75000000, ExpectedResult: false,
ExpectedError: nil},

   // Test case 4: Age in [18, 100], Salary in [-infinity, 25m), ExpectedResult: false,
ExpectedError: nil
     {Name: "Adult Low Income", Age: 25, Salary: 15000000, ExpectedResult: false,
ExpectedError: nil},

     // Test case 5: Age in [18, 100], Salary in [25m, 50m], ExpectedResult: true,
ExpectedError: nil
     {Name: "Adult Middle Income", Age: 45, Salary: 35000000, ExpectedResult: true,
ExpectedError: nil},

    // Test case 6: Age in [18, 100], Salary in [50m, infinity), ExpectedResult: false,
ExpectedError: nil
     {Name: "Adult High Income", Age: 75, Salary: 75000000, ExpectedResult: true,
ExpectedError: nil},

    // Test case 7: Age in [100, infinity), Salary in [-infinity, 25m), ExpectedResult:
false, ExpectedError: nil
```

```
    {Name: "Elderly Low Income", Age: 105, Salary: 15000000, ExpectedResult: false,
ExpectedError: InvalidAgeError},

   // Test case 8: Age in [100, infinity), Salary in [25m, 50m], ExpectedResult: false,
ExpectedError: InvalidAgeError
    {Name: "Elderly Middle Income", Age: 110, Salary: 35000000, ExpectedResult: false,
ExpectedError: InvalidAgeError},

    // Test case 9: Age in [100, infinity), Salary in [50m, infinity), ExpectedResult:
false, ExpectedError: InvalidAgeError
    {Name: "Elderly High Income", Age: 120, Salary: 75000000, ExpectedResult: false,
ExpectedError: InvalidAgeError},
  }
```