Лабораториска вежба 1 – група А (211055)

Interface-based approach:

Овие карактеристики се однесуваат директно на природата на влезните параметри на функцијата, и обично се идентификуваат од самиот опис на функцијата.

Имаме 2 влезни параметри: ukVisaApplications и usaVisaApplications. Оттука може да дефинираме неколку карактеристики кои се однесуваат на истите:

C1: ukVisaApplications не е null: b1=true, b2=false;

C2: usaVisaApplications не е null: b1=true, b2=false;

C3: ukVisaApplications не е празна листа: b1=true, b2=false;

C4: usaVisaApplications не е празна листа: b1=true, b2=false;

Functionality-based approach:

Овој тип на карактеристики се однесуваат на самите влезни параметри, но повеќе би се однесувале на поврзаност помеѓу нив и како влијаат на крајниот излез на функцијата.

Овде таква карактеристика може да биде:

C5: Апликанти кои аплицирале и за ukVisaApplications и за usaVisaApplications:

C5.1 = барем еден аплицирал и за ukVisaApplications и за usaVisaApplications

C5.2 = нема ниту еден кој аплицирал и за двете визи

А) Карактеристиките C1,C2, C3 и C4 се точно/неточно (true/false) тип на изрази. Партиционирањето на овие 2 карактеристики ќе биде на 2 блока - true (T) и false (F). Со ваквата поделба сме сигурни дека овие карактеристики и нивните блокови го задоволуваат својствата на дисјунктност (нешто не може да биде точно и неточно во исто време)

Карактеристиката C5 исто така го задоволува својството за дисјунктност затоа што или ќе имаме барем еден апликант кој аплицирал за двете визи или ќе немаме ниту еден, но двете во исто време не можат.

Б) Карактеристиките C1,C2, C3 и C4 го задоволуваат својството за комплетност бидејќи се точно/неточно (true/false) тип на изрази, па не може да биде нешто трето.

Исто така и C5 го задоволуваат својството за комплетност бидејќи или ќе имаме еден апликант или ниту еден.

В) Во овој случај Base Choice Coverage би било карактеристиките C1,C2,C3 и C4 да бидат true(T), а C5 да биде дека барем еден аплицирал и за двете визи што во нашиот случај е C5.1. Па така нашиот happy path и прв тест е **T T T T C5.1**.

Другите тестови се добиваат на тој начин што секоја карактеристика ќе се менува во сите други можни блокови (вредности), а другите ќе бидат фиксирани. Па така ги добиваме следните тестови:

F T T T C5.1 – тест 2

T F T T C5.1 – тест 3

T T F T C5.1 – тест 4

T T T F C5.1 – тест 5

T T T T C5.2 – тест 6

Сега од овие тестови треба да видиме дали има некои што се невозможни (infeasible):

- Имаме infeaseblie случај во вториот тест, бидејќи ukVisaApplication не може истовремено да биде и null и празна листа што значи ќе мора да го измениме со F T F T C5.1 но сеуште тестот е невозможен бидејќи ќе ни падне последниот тест затоа што ukVisaApplication ќе биде празна и null, но поради тоа ќе нема заеднички апликанти со usaVisaApplicattion, па во овој случај заменуваме F T F T C5.2.

- Ист случај имаме и во третиот тест, само овој пат за usaVisaApplications, па решението ќе биде исто како и во вториот односно T F T F C5.2.

- Кај четвртиот тест имаме празна листа во ukVisaApplications, но и барем еден што аплицирал за двете визи што е невозможно, па затоа ќе замениме за C5 односно тестот ќе биде T T F T C5.2.

- Кај петтиот тест исто така имаме празна листа за usaVisaApplications, но и барем еден што аплицирал за двете визи, па како и во погорниот случај ќе замениме за C5 што би било T T T F C5.2 .

- И кај шестиот тест иако имаме апликанти кои аплицирале и за usaVisaApplication и за ukVisaApplications сепак немаме заеднички кои аплицирале и за двете истовремено, но сепак тој е **валиден** тест.

На крај ги имаме следните тестови:

T T T T C5.1 – тест 1 BS
F T F T C5.2 – тест 2
T F T F C5.2 – тест 3
T T F T C5.2 – тест 4
T T T F C5.2 – тест 5
T T T T C5.2 – тест 6

Наредно треба уште да дефинираме вредности што одговараат за секој од тестовите:

Тест 1, BS(T T T T C5.1):
ukVisaApplications: x1, x2, x3, x4
usaVisaApplication : x2, x3, x4, x5
Очекуван излез: 3

Тест 2(F T F T C5.2):
ukVisaApplications: null
usaVisaApplication : x1, x2
Очекуван излез: NullPointerException

Тест 3(T F T F C5.2):
ukVisaApplications: x1, x2
usaVisaApplication : null
Очекуван излез: NullPointerException

Тест 4(T T F T C5.2):
ukVisaApplications: [ ]
usaVisaApplication : x1, x2
Очекуван излез: null

Тест 5(T T T F C5.2):
ukVisaApplications: x1, x2
usaVisaApplication : [ ]
Очекуван излез: null

Тест 6(T T T T C5.2):
ukVisaApplications: x1, x2, x3
usaVisaApplication : x4, x5, x6
Очекуван излез: null

```java
import java.util.ArrayList;
import java.util.List;

5 usages
public class VisaApplications {

    5 usages
    public static List<String> applicantsForBothVisas(List<String> ukVisaApplications,
                                                       List<String> usaVisaApplications){
        if (ukVisaApplications == null || usaVisaApplications == null){

            throw new NullPointerException("ukVisaApplications or usaVisaApplications can't be null");
        }

        List<String> commonApplicants = new ArrayList<>(ukVisaApplications);
        commonApplicants.retainAll(usaVisaApplications);

        return commonApplicants.isEmpty() ? null : commonApplicants;

    }
}
```

```java
import org.junit.jupiter.api.Test;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.*;


public class VisaApplicationsTest {

    @Test
    //Test 1 Base choice
    public void test1(){
        List<String> ukVisaApplication = Arrays.asList("x1", "x2", "x3", "x4");
        List<String> usaVisaApplication = Arrays.asList("x2","x3","x4", "x5");

        List<String> commonApplications = VisaApplications.applicantsForBothVisas(ukVisaApplication, usaVisaApplication);
        assertEquals(Arrays.asList("x2", "x3", "x4"), commonApplications);
    }
    @Test
    //Test 2 uk=null
    public void test2(){
        assertThrows(NullPointerException.class, () ->{
            VisaApplications.applicantsForBothVisas( ukVisaApplications: null, Arrays.asList("x1", "x2"));
        });
    }
    @Test
    //Test 3 usa=null
    public void test3(){
        assertThrows(NullPointerException.class, () ->{
            VisaApplications.applicantsForBothVisas(Arrays.asList("x1", "x2"), usaVisaApplications: null);
        });
    }
```

```java
    }
    @Test
    //Test 4 uk=[ ]
    public void test4(){
        List<String> ukVisaApplications = new ArrayList<>();
            assertNull(VisaApplications.applicantsForBothVisas(ukVisaApplications, Arrays.asList("x1", "x2")));
    }
    @Test
    //Test 5 usa=[ ]
    public void test5(){
        List<String> usaVisaApplications = new ArrayList<>();
        assertNull(VisaApplications.applicantsForBothVisas(Arrays.asList("x1", "x2"), usaVisaApplications));
    }
    @Test
    //Test 6
    public void test6(){
        List<String> ukVisaApplication = Arrays.asList("x1", "x2", "x3");
        List<String> usaVisaApplication = Arrays.asList("x4","x5","x6");

        assertNull(VisaApplications.applicantsForBothVisas(ukVisaApplication, usaVisaApplication));
    }
}

```