СКИТ – Лабораториска вежба 1

За функцијата дадена во продолжение треба да се измоделира влезниот домен и тоа според двата начина, **interface based** и **functionality based**.

```
/**
 * The method should act as follows:
 *          if ukVisaApplications or usaVisaApplications is null throw NullPointerException
 *          else, return a (non-null) list of applicants (applicant ids) that have applied both for UK and USA Visa
 *          else, if there are no such applicants return null
 */
1 usage
public static List<String> applicantsForBothVisas(List<String> ukVisaApplications,
                                                   List<String> usaVisaApplications) {
```

Откако ќе ги дефинирате карактеристиките треба да ги поделите во блокови и за секој од нив да се одговорат следните прашања:

а) Дали партиционирањето на влезните параметри го задоволува својството дисјунктност? Зошто? Ако не, напишете измени за да се задоволи својството дисјунктност.

б) Дали партиционирањето на влезните параметри го задоволува својството комплетност? Зошто? Ако не, напишете измени за да се задоволи својството комплетност.

в) Изберете основен тест, и за него наведете ги сите тестови за задоволување на **Base Choice Coverage** (**BCC**) критериумот. Колкав е бројот на тестови што треба да се направи?

г) Напревете junit тестови според BCC критериумот за покривање на ISP.

I. Влезен домен за ф-ја public static List<String> applicantsFor BothVisas ( )

Interface-based карактеристики:

C1: ukVisaApplications не е null

C2: ukVisaApplications не е празна низа

C3: usaVisaApplications не е null

C4: usaVisaApplications не е празна низа

Functionality-based карактеристики:

C5: има најмалку еден елемент коишто истовремено се наоѓа во двете низи (и во низата ukVisaApplications и во низата usaVisaApplications)

C6: крајната листа од applicantsForBothVisas не е null

II. Поделба во блокови на карактеристиките

Карактеристиките C1, C2, C3, C4, C5 и C6 се точно/неточно (TRUE/FALSE) тип на изрази. Партиционирањето на овие карактеристики ќе биде на 2 блока - TRUE (T) и FALSE (F).

Со ваквата поделба сме осигурани дека овие карактеристики и нивните блокови го задоволуваат својствата на дисјунктност (нешто не може да биде точно и неточно истовремено), и комплетност (или е точно или е неточно, не може да биде нешто трето).

Карактеристика	*C1*	*C2*	*C3*	*C4*	*C5*	*C6*
Партиционирање	{True, False}	{True, False}	{True, False}	{True, False}	{True, False}	{True, False}

а) Дали партиционирањето на влезните параметри го задоволува својството дисјунктност? Зошто? Ако не, напишете измени за да се задоволи својството дисјунктност.

Партиционирањето на сите влезните параметри го задоволува својството на дисјунктност, бидејќи за секоја од влезните параметри може да биде или True или False, но не и двете истовремено.

б) Дали партиционирањето на влезните параметри го задоволува својството комплетност? Зошто? Ако не, напишете измени за да се задоволи својството комплетност.

Партиционирањето на сите влезните параметри го задоволува својството на комплетност, бидејќи за секоја од влезните параметри може да биде или True или False, но не и нешто друго.

III. Тестови за Base Choice Coverage (BCC) критериум

III.1 Невозможни тестови

Сега, од сите овие комбинации имаме тестови коишто се невозможни. Па така, сите тестови во форматот:

T F _ _ _ _, F T _ _ _ _, _ _ T F _ _ и _ _ F T _ _

Се невозможни тестови бидејќи истовремено не е возможно иста листа истовремено да е null и да не е празна, и обратно, да не е null и да е празна.

Исто така, невозможни се и следниве тестови:

_ _ _ _ T F и _ _ _ _ F T

Се невозможни тестови бидејќи не е возможно во крајната листа истовремено да имаме најмалку еден елемент и крајната листа да е null, и обратно, крајната листа истовремено да нема ниту еден елемент и крајната листа да не е null.

Невозможни тестови се следниве:

T T F F T T, F F T T T T

Се невозможни тестови бидејќи не е возможно една од двете почетни листи да е празна, а во крајната листа да имаме барем еден елемент.

Заклучно, невозможни тестови се:

- **T F _ _ _ _**
- **F T _ _ _ _**
- **_ _ T F _ _**
- **_ _ F T _ _**
- **_ _ _ _ T F**
- **_ _ _ _ F T**
- **T T F F T T**
- **F F T T T T**

III.2 Можни тестови

По сите невозможни тестови, да ги разгледаме можните тестови:

- **T T T T T T**
- **T T F F F F**
- **F F T T F F**
- **T T T T F F**

C1	*C2*	*C3*	*C4*	*C5*	*C6*		*Сценарио*
T	T	T	T	T	T		Низите ukVisaApplications и usaVisaApplications не се празни и не се null, како и има барем еден елемент во финалната низа и не е null.
T	T	F	F	F	F		Низата ukVisaApplications не е празна и не е null, низата usaVisaApplications е празна и е е null, како и нема ниту еден елемент во финалната низа и е null.
F	F	T	T	F	F		Низата usaVisaApplications не е празна и не е null, низата ukVisaApplications е празна и е е null, како и нема ниту еден елемент во финалната низа и е null.
T	T	T	T	F	F		Низите ukVisaApplications и usaVisaApplications не се празни и не се null, но нема ниту еден елемент во финалната низа и низата е null.

T T T T T T

-> Низите ukVisaApplications и usaVisaApplications не се празни и не се null, како и има барем еден елемент во финалната низа и не е null.

-> Од овде произлегува дека имаме барем еден елемент кој истовремено се наоѓа и во низата ukVisaApplications и во usaVisaApplications, поради што во финалната низа (ид на апликантите кои аплицирали и за двете низи) имаме еден или повеќе елементи.

T T T T F F

-> Низите ukVisaApplications и usaVisaApplications не се празни и не се null, но нема ниту еден елемент во финалната низа и низата е null.

-> Од овде произлегува дека немаме ниту еден елемент кој истовремено се наоѓа и во низата ukVisaApplications и во usaVisaApplications, поради што во финалната низа (ид на апликантите кои аплицирали и за двете низи) немаме елементи.

T T F F F F и F F T T F F

-> Една од низите (ukVisaApplications или usaVisaApplications) не е празна и не е null, а другата низа е празна и е е null, поради што нема ниту еден елемент во финалната низа и финалната низа е null.

-> Поради едната празна низа, произлегува дека немаме ниту еден елемент во финалната низа (невозможно е да имаме елементи во финалната низа).

Потребно е овие карактеристики да се комбинираат и со некој критериум за покривање. Во овој случај ќе употребиме Base Choice Coverage (BCC).

За таа цел, најпрво треба да се одбере еден Base Choice од којшто ќе се градат сите други тестови. Доколку ги гледаме подредено карактеристиките (C1 C2 C3 C4 C5 C6), и дефинираните блокови, во овој случај base choice би бил сите карактеристики да се true, односно првиот тест би бил T T T T T T (сите карактеристики се True). Се избира овој случај бидејќи ова е таканаречен best case scenario.

Base Choice: T T T T T T -> тест 1
F T T T T T -> тест 2
T F T T T T -> тест 3
T T F T T T -> тест 4
T T T F T T -> тест 5
T T T T F T -> тест 6
T T T T T F -> тест 7

Од горенаведените тестови, невозможни се: тест 2, тест 3, тест 4, тест 5 и тест 7.

Тест 2 е невозможен тест, бидејќи не може истовремено ukVisaApplications да е null и да не е празна. Овој тест го заменуваме со F F T T T T. Но и со оваа промена тестот сеуште е невозможен, бидејќи не може ukVisaApplications да е null и празна и истовремено да постои најмалку еден елемент во финалната низа. **Конечно новиот тест 2 е F F T T F F.**

Тест 3 е невозможен тест, бидејќи не може ukVisaApplications да е празна и да постои најмалку еден елемент во финалната низа. **Тест 3 го заменуваме со T F T T F F.**

Исто како и тест 2, тест 4 е невозможен тест, бидејќи не може истовремено usaVisaApplications да е null и да не е празна. Овој тест го заменуваме со T T F F T T. Но и со оваа промена тестот сеуште е невозможен, бидејќи не може ukVisaApplications да е null и празна и истовремено да постои најмалку еден елемент во финалната низа. **Конечно новиот тест 4 е T T F F F F.**

Исто како и тест 3, тест 5 е невозможен тест, бидејќи не може usaVisaApplications да е празна и да постои најмалку еден елемент во финалната низа. **Тест 5 го заменуваме со T T T F F F.**

Тест 6 е исто така невозможен тест, бидејќи финалната низа не може истовремено да има најмалку еден елемент и да е null. Па така, **тест 6 го заменуваме со T T T T F F**.

Од ова, ново добиеното сценарио е следно:

Base Choice: T T T T T T -> тест 1
F F T T F F -> тест 2
T F T T F F -> тест 3
T T F F F F -> тест 4
T T T F F F -> тест 5
T T T T F F -> тест 6
T T T T T F -> тест 7

IV. Junit тестови за покривање на ISP

```
package junit;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;


public class VisaApplications {
    public static List<String> applicantsForBothVisas(List<String>
ukVisaApplications, List<String> usaVisaApplications) {

        Set<String> ukSet = new HashSet<>(ukVisaApplications);
        Set<String> usaSet = new HashSet<>(usaVisaApplications);

        Set<String> bothVisasSet = new HashSet<>(ukSet);
        bothVisasSet.retainAll(usaSet);


        return new ArrayList<>(bothVisasSet);
    }
}
```

```
package junit;
import org.junit.jupiter.api.Test;
import java.util.Arrays;
import java.util.List;
import static org.junit.Assert.assertNotEquals;
import static org.junit.Assert.assertThrows;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class VisaApplicationsTest {


    @Test
    //Test1 T T T T T T
    public void testApplicantsForBothVisas() {

        List<String> ukApplications = Arrays.asList("ID1", "ID2", "ID3");
        List<String> usaApplications = Arrays.asList("ID2", "ID3", "ID4");

        List<String> expectedResult = Arrays.asList("ID2", "ID3");

        List<String> result =
VisaApplications.applicantsForBothVisas(ukApplications, usaApplications);

        assertEquals(expectedResult, result);
    }
```

```
    @Test
    //Test T F T F F F
    public void testApplicantsForBothVisasEmptyLists() {

        List<String> ukApplications = Arrays.asList();
        List<String> usaApplications = Arrays.asList();

        List<String> expectedResult = Arrays.asList();

        List<String> result =
VisaApplications.applicantsForBothVisas(ukApplications, usaApplications);

        assertEquals(null, result);
    }


    @Test
    //Test3 T F T T F F AND Test5 T T T F F F
    public void testApplicantsForBothVisasEmptyList() {

        List<String> ukApplications = Arrays.asList();
        List<String> usaApplications = Arrays.asList("ID1", "ID2", "ID3");

        List<String> expectedResult = Arrays.asList();

        List<String> result =
VisaApplications.applicantsForBothVisas(ukApplications, usaApplications);

        assertEquals(null, result);
    }


    @Test
    //Test6 T T T T F F
    public void testApplicantsForBothVisasNoCommonIDs() {

        List<String> ukApplications = Arrays.asList("ID1", "ID2");
        List<String> usaApplications = Arrays.asList("ID3", "ID4");

        List<String> expectedResult = Arrays.asList();

        List<String> result =
VisaApplications.applicantsForBothVisas(ukApplications, usaApplications);

        assertEquals(null, result);
    }
```

```
    @Test()
    //Test2 F F T T F F AND Test4 T T F F F F
    public void testApplicantsForBothVisasNullList() {

        List<String> ukApplications = Arrays.asList("ID6", "ID7", "ID8");
        List<String> usaApplications = null;

        List<String> result =
VisaApplications.applicantsForBothVisas(ukApplications, usaApplications);

        assertThrows(NullPointerException.class, () -> {
            VisaApplications.applicantsForBothVisas(ukApplications,
usaApplications);
        });
    }


    @Test
    public void testApplicantsForBothVisasExtraIDsInResult() {
        List<String> ukApplications = Arrays.asList("ID1", "ID2", "ID3");
        List<String> usaApplications = Arrays.asList("ID2", "ID3", "ID4");

        List<String> expectedResult = Arrays.asList("ID2", "ID3", "ID4");
        List<String> result =
VisaApplications.applicantsForBothVisas(ukApplications, usaApplications);

        assertEquals(expectedResult, result);
    }


    @Test
    public void testApplicantsForBothVisasUnorderedResult() {
        List<String> ukApplications = Arrays.asList("ID1", "ID2", "ID3");
        List<String> usaApplications = Arrays.asList("ID2", "ID3", "ID4");

        List<String> expectedResult = Arrays.asList("ID2", "ID3");
        List<String> result =
VisaApplications.applicantsForBothVisas(ukApplications, usaApplications);

        assertNotEquals(expectedResult, result);
    }
}
```

Изработил:
Бојан Ристов (211151)