

Софтверски квалитет и тестирање

Домашна задача број 3

Ема Спирова 163067

```
public Set intersection(Set s1, Set s2)
//Effects: If s1 or s2 is null throw NullPointerException
//else return a (non null) Set to the intersection
// of Sets s1 and s2
```

Interface based approach

Два параметри: Set s1, Set s2.

Карактеристики за s1:

c1 = “s1 е празно множество“

c2 = “s1 е null“

c3 = “ s1 има барем еден елемент“

Карактеристики за s2:

c4 = “s2 е празно множество“

c5 = “s2 е null“

c6 = “ s2 има барем еден елемент“

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

- За да се партиции потребно е да се дисјунктни и комплетни. Партиционирањето на влезните параметри го задоволува својството дисјунктност, бидејќи карактеристиките/блоковите не се поклопуваат. Нема партиција која ќе припаѓа и на c1 и на c2, c3, односно пресекот на карактеристиките е празно множество.

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

- Партиционирањето на влезните параметри го задоволува својството комплетност, бидејќи блоковите го покриваат целиот домен D.

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?

- Основен тест: c2, c5.

Број на тестови: $1 + (3-1) + (3-1) = 1 + 2 + 2 = 5$.

Останати тестови:

c1, c5 c2, c4

c3, c5 c2, c6

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

JUnit тест за c2, c6

```
@Test
public void test1() {
    Set<Integer> set = new HashSet<Integer>();
    set.addAll(Arrays.asList(new Integer[] {1}));

    assertThrows(NullPointerException.class, () -> intersection.intersection(null, set));
}
```

JUnit тест за c3, c5

```
@Test
public void test2() {
    Set<Integer> set = new HashSet<Integer>();
    set.addAll(Arrays.asList(new Integer[] {1}));

    assertThrows(NullPointerException.class, () -> intersection.intersection(set, null));
}
```

JUnit тест за c2, c5

```
@Test
public void test3() {
    Set<Integer> set = new HashSet<Integer>();

    assertThrows(NullPointerException.class, () -> intersection.intersection(null, null));

}
```

Functionality based approach

Два параметри: Set s1, Set s2.

Карактеристики за s1 во однос на s2:

c1 = “пресекот $s1 \cap s2$ е унија“

c2 = “ $s1 \cap s2$ е празно множество“

c3 = “s2 е подмножество на s1“

c4 = “s1 е подмножество на s2“

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

- За да се партиции потребно е да се дисјунктни и комплетни. Партиционирањето на влезните параметри го задоволува својството дисјунктност, бидејќи карактеристиките/блоковите не се поклопуваат. Нема партиција која ќе припаѓа и на c1 и на c2, односно пресекот на карактеристиките е празно множество.

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

- Партиционирањето на влезните параметри го задоволува својството комплетност, бидејќи блоковите го покриваат целиот домен D.

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?

- Основен тест: c1.

Број на тестови: $1 + (4 - 1) = 1 + 3 = 4$.

Останати тестови: c2, c3, c4.

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

JUnit тест за c1, кога двете множества имаат исти елементи

```
@Test
public void equalSets(){

    Set<Integer> s1 = new HashSet<Integer>();
    s1.addAll(Arrays.asList(new Integer[] {1, 2}));
    Set<Integer> s2 = new HashSet<Integer>();
    s2.addAll(Arrays.asList(new Integer[] {1, 2}));

    assertEquals(s1, intersection.intersection(s1, s2));
```

```
}
```

JUnit тест за c2

```
@Test
```

```
    public void emptySets(){

        Set<Integer> s1 = new HashSet<Integer>();
        s1.addAll(Arrays.asList(new Integer[] {}));
        Set<Integer> s2 = new HashSet<Integer>();
        s2.addAll(Arrays.asList(new Integer[] {}));

        assertEquals(s1, intersection.intersection(s1, s2));
    }
```

JUnit тест за c3

```
@Test
```

```
    public void s2SubsetOfs1() {

        Set<Integer> s1 = new HashSet<Integer>();
        s1.addAll(Arrays.asList(new Integer[] {1, 2, 3, 4, 5}));
        Set<Integer> s2 = new HashSet<Integer>();
        s2.addAll(Arrays.asList(new Integer[] {1, 2, 3}));

        assertEquals(s2, intersection.intersection(s1,s2));
    }
```

JUnit тест за c4

```
@Test
```

```
    public void s1SubsetOfs2() {

        Set<Integer> s1 = new HashSet<Integer>();
        s1.addAll(Arrays.asList(new Integer[] {1, 2, 3}));
        Set<Integer> s2 = new HashSet<Integer>();
        s2.addAll(Arrays.asList(new Integer[] {1, 2, 3, 4, 5}));

        assertEquals(s1, intersection.intersection(s1,s2));
    }
```