Софтверски квалитет и тестирање

Домашна задача 1

183016

A) Interface based approach: доколку моделирањето на влезниот домен е според овој пристап, за секој влезен параметар од дадената функција, дефинираме посебна карактеристика, којашто ќе се мапира единствено во соодветниот влезен параметар и нема да се однесува на повеќе параметри.

Според ова, се дефинираат 2 карактеристики:

C_1: set1 е non-null објект.

C_2: set2 е non-null објект.

Овие две карактеристики може да бидат партиционирани на ист начин, т.е. според вистинитоста на дефинираните искази и затоа секоја од нив се дели во 2 домени: true и false.

Карактеристика	B1	B2	Base case
C_1: set1 е non-null објект.	true	false	true
C_2: set2 е non-null објект.	true	false	true

Ваквото партиционирање на карактеристиките во неколку домени го задоволува својството за дисјунктност: влезните параметри set1, set2 може да имаат елементи или може да бидат празни, но не и двете истовремено. Исто така задоволено е и својството за комплетност: дефинираните блокови за двете карактеристики го покриваат целиот домен и не постои вредност од влезниот домен којашто не е вклучена во било кој од блоковите.

Откако карактеристиките ќе бидат партиционирани во домени, потребно е да се одбере критериум за покривање, според кој ќе ги одредиме специфичните тест барања кои ќе се користат за генерирање на jUnit тестови. Во овој случај го користеме **Base Choice Coverage (BCC)** критериумот и поради тоа најпрво потребно е за секоја од карактеристиките да одредеме *base choice блок*.

За base choice блок за секоја од карактеристиките ќе го одбереме оној блок чиишто вредности ќе резултираат со нормално (очекувано) однесување на програмата.

Тоа се блоковите B1 за секоја од карактеристиките.

Метод	Карактеристики	Исклучоци	Тест барања
setDifference(Set set1, Set set2)	C_1,C_2	NullPointerException	TT,TF,FT

Base choice test за методот setDifference(Set set1, Set set2) ќе биде (T,T) и во превод значи дека доколку set1 објектот содржи елементи и доколку set2 објектот содржи елементи, методата ќе врати резултантен објект од тип Set (доколку постојат елементи во разликата), којшто ќе ја содржи разликата меѓу set1 и set2, инаку ќе врати null.

Останатите non-base тестови се генерираат според дефиницијата на BCC критериумот и вкупниот број потребни тестови за да се задоволи критериумот е 3. Останатите 2 тестови треба да резултираат со фрлање на исклучок од тип NullPointerException бидејќи во секој од случаите едно од множествата е null, а во спецификацијата на методата е наведено дека со такви влезни параметри методата треба да фрли исклучок од тој тип.

Сите потребни jUnit тестови кои одговараат на дефинираните тест барања се наоѓаат во соодветната .java датотека.

B) Functionality based approach: според овој пристап на моделирање на влезниот домен, карактеристиките се дефинираат според целта и функционалноста на методата којашто ја тестираме. Во овој случај можеме да ги земеме во предвид релациите помеѓу влезните параметри и една карактеристика е можно да се мапира во повеќе влезни параметри. Затоа, на некој начин дефинирањето на ваквите карактеристики може да биде поспецифично и да бара повеќе внимание.

Според ова генерираме 3 карактеристики:

C_1: Типот на објект којшто методата го враќа.

C_2: Бројот на елементи во set1.

C_3: Бројот на елементи во set2.

Првата карактеристика може да биде поделена на 3 домени, бидејќи методата којашто ја тестираме или ќе врати објект од тип Set, во којшто објект ќе се наоѓа разликата меѓу пратените множества, или ќе врати null објект, доколку разликата е 0 или ќе резултира со NullPointerException, доколку некој од влезните параметри е празно множество. Блоковите за C_1 се дисјунктни бидејќи не е можно истовремено да бидат вратени сите типови објекти и се комплетни бидејќи тоа се единствените типови објекти коишто методата може да ги врати.

Карактеристика	**B1**	**B2**	**B3**	**Base case**
C_1: Типот на објект којшто методата го враќа.	Set	null	NullPointer Exception	Set
C_2: Бројот на елементи во set1.	>0	0	/	>0
C_3: Бројот на елементи во set2.	>0	0	/	>0

Следните две карактеристики се партиционираат на ист начин и тоа: бројот на елементи во било кое од влезните множества може да биде 0 (празно множество) или може да биде поголемо од 0. Ваквото партиционирање на карактеристики резултира со дисјунктни блокови, бидејќи множествата може да бидат празни или со елементи, но не и двете истовремено и блоковите се комплетни бидејќи се покриени сите можни вредности за големината на множествата.

Исто како во првиот пристап критериумот за покривање којшто ќе го користеме е **Base Choice Coverage (BCC)** и затоа следно што треба да направиме е да ги дефинираме *base choice блоковите* за секоја од карактеристиките – тоа ќе бидат блоковите чиишто вредности ќе резултираат со нормално (очекувано) однесување на програмата, т.е методата ќе ја постигне својата цел. Тоа се блоковите B1 за секоја од карактеристиките.

Метод	**Карактеристики**	**Исклучоци**	**Тест барања**
setDifference(Set set1, Set set2)	C_1,C_2,C_3	NullPointerException	(Set, >0, >0) (null,>0,>0) (NPE,>0,>0) (Set,0,>0) (Set,>0,0)

Base choice test за оваа метода ќе биде (Set, >0, >0) и тоа би значело дека доколку како влезни параметри бидат предадени две множества коишто имаа елементи и постои разлика меѓу првото и второто множество, методата ќе врати резултантно множество кое ќе ги содржи елементите од set1, но не и од set2.

Останатите non-base тестови се генерираат според дефиницијата на BCC критериумот и вкупниот број потребни тестови за да се задоволи критериумот е 5: еден base choice test и по еден тест за секој постоечки non-base блок.

Во овој случај, важно е да се забележи дека неколку тест барања се невозможни и бидејќи одбраниот критериум за покривање е BCC, наметнато правило е дека доколку постојат невозможни тест барања, потребно е секое од нив да биде заменето со некое возможно тест барање, кое се уште не постои во множеството на барања.

Невозможни тест барања	**Ревизирани тест барања**
(NPE,>0,>0) , (Set,0,>0) , (Set,>0,0)	(NPE, >0, 0) , (NPE, 0, >0) , (NPE, 0, 0)

На пример, барањето (NPE, >0, >0) е невозможно бидејќи не е можно методата да резултира со фрлање на исклучок, доколку двете пратени множества имаат елементи. Затоа ваквото барање може да се замени со (NPE, >0, 0). Друго невозможно барање е (Set, 0, >0) во овој случај методата треба да резултира со

фрлање исклучок и не може да врати објект од тип Set, можно тест барање би било (NPE, 0, >0). Последното тест барање коешто е невозможно е (Set, >0, 0) и соодветна замена би била (NPE, 0, 0).

Сите потребни jUnit тестови кои одговараат на дефинираните тест барања се наоѓаат во соодветната .java датотека.