Institut für Informatik
Jun.-Prof. Dr.-Ing. Elke Pulvermüller
B. Sc. Sergey Krutikov
B. Sc. Alexander Brandt

Universität Osnabrück, 06.11.2012
http://www-lehre.inf.uos.de/~sq

Testat bis Freitag, 16.11.2012, 19:00 Uhr

Übungen zu Software-Qualität

Wintersemester 2012/2013

Übungsblatt 4

Aufgabe 4.1 (Syntaxtest, 8 Punkte)

Die Ziffern zur Darstellung von römischen Zahlen lauten dem Wert nach absteigend: M, D, C, L, X, V, I. Diese Ziffern haben jeweils einen festen Wert (Zehnerpotenzen als Basiswerte und fünffache Hilfsbasiswerte):

$$
\begin{aligned}
M &= 1000 \\
D &= 500 \\
C &= 100 \\
L &= 50 \\
X &= 10 \\
V &= 5 \\
I &= 1
\end{aligned}
$$

Die nachfolgenden Regeln einer Grammatik definieren, wie die römischen Zahlen bis 3999 gebildet werden. Nichtterminale Symbole sind in spitzen Klammern dargestellt, λ bezeichnet das leere Wort:

<S> → <T> <H> <Z> <E>
<T> → λ | MMM | MM | M
<H> → λ | CM | DCCC | DCC | DC | D | CD | CCC | CC | C
<Z> → λ | XC | LXXX | LXX | LX | L | XL | XXX | XX | X
<E> → λ | IX | VIII | VII | VI | V | IV | III | II | I

a) Transformieren Sie die gegebene Grammatik in eine entsprechende Hierarchie von Syntaxdiagrammen, die sowohl die Nicht-Terminalsymbole als auch die Terminalsymbole enthalten. (5 Punkte)

b) Wieviele Testfälle werden mindestens benötigt, wenn Knotenabdeckung (Durchlaufen sämtlicher Terminalsymbole) erreicht werden soll? (1 Punkt)

c) Wieviele Testfälle werden für Kantenabdeckung (Durchlauf durch alle Kanten der Diagramme) mindestens benötigt? (1 Punkt)

d) Wieviele Testfälle werden für vollständige Pfadabdeckung der Syntaxdiagramme benötigt? (1 Punkt)

Aufgabe 4.2 (Anweisungsüberdeckungstest, 10 Punkte)

Machen Sie sich mit der Methode `analyseList()` auf der folgenden Seite vertraut. Sie finden die Methode auch in in der JAVA-Klasse unter der folgenden Internet-Adresse:

`http://www-lehre.inf.uos.de/~sq/2012/aufgaben/blatt04/ListAnalyser.java`

a) Erzeugen Sie den Kontrollflussgraphen für die Methode `analyseList()`. Tragen Sie *alle* theoretisch ausführbaren Zweige ein. (6 Punkte)

b) Berechnen Sie den „Anweisungsüberdeckungsgrad“ (C_0-Test), für den Fall, dass die Methode mit den folgenden Eingaben aufgerufen wird: (4 Punkte)

   1. Parameter `in_lstList = Arrays.asList("0", "1", "2", "3")`
   2. Parameter `out_aiResults = new int[0]`

Aufgabe 4.3 (Zweigüberdeckungstest, 14 Punkte)

a) Geben Sie Eingaben und Testfälle an, mit denen für die Methode `analyseList()` 100% „Zweigüberdeckung“ (C_1-Test) erreicht wird. Halten Sie dabei die Anzahl der Testfälle möglichst gering. (5 Punkte)

b) Nennen Sie ein Beispiel für einen „primitiven Zweig“ anhand des Kontrollflussgraphen zu der Methode `analyseList()` und begründen Sie. (3 Punkte)

c) Warum werden primitive Zweige zur Berechnung des Überdeckungsmaßes von Zweigüberdeckungstests eingesetzt? (2 Punkte)

d) Benennen Sie die wesentlichen Merkmalsunterschiede zwischen dem Anweisungsüberdeckungstest (C_0-Test) und Zweigüberdeckungstest (C_1-Test). (4 Punkte)

Aufgabe 4.4 (Pfadüberdeckungstest, 6 Punkte)

Berechnen Sie die Anzahl aller möglichen, vollständigen Pfade der Methode `analyseList()` für den Fall, dass sichergestellt ist, dass die übergebene Liste 1000 Elemente enthält. Geben Sie die Berechnungsvorschrift und das Ergebnis an. (6 Punkte)

Aufgabe 4.5 (Offener Frageteil, 6 Punkte)

Beantworten Sie Ihrer Tutorin bzw. Ihrem Tutor Fragen zur Veranstaltung „Software-Qualität“.

– bitte wenden –

```
01    /** Method to analyse the elements of a given list of strings. */
02    public boolean analyseList(List<String> in_lstList, int[] out_aiResults) {
03        boolean bResult = false;
04        if (in_lstList != null && !in_lstList.isEmpty()) {
05            // initialize counters
06            int iNumberCounter = 0;
07            int iEvenNumberCounter = 0;
08            int iWordCounter = 0;
09            int iEvenWordCounter = 0;
10            // go through all elements of the given list
11            for (String strElement : in_lstList) {
12                // ignore 'null' and empty strings
13                if (strElement != null && !strElement.isEmpty()) {
14                    int iNumber = 0;
15                    boolean bNumberFound = false;
16                    // try to parse the current list element to an int value
17                    try {
18                        iNumber = Integer.parseInt(strElement);
19                        bNumberFound = true;
20                    } catch (NumberFormatException e) {
21                    // do nothing
22                    }
23                    // check if a number was found
24                    if (bNumberFound) {
25                        iNumberCounter++;
26                        // check if number is even or odd
27                        if (iNumber % 2 == 0) {
28                            iEvenNumberCounter++;
29                            System.out.println(iEvenNumberCounter +
30                                    ". even number found: " + iNumber);
31                        } else {
32                            System.out.println(iNumberCounter +
33                                    ". number found: " + iNumber);
34                        }
35                    } else { // no number found
36                        iWordCounter++;
37                        // check if the number of characters of the current
38                        // list element is even or odd
39                        if (strElement.length() % 2 == 0) {
40                            iEvenWordCounter++;
41                            System.out.println(iEvenWordCounter + ". word " +
42                                    "with even number of characters found: " +
43                                    strElement);
44                        } else {
45                            System.out.println(iWordCounter + ". word found: " +
46                                    strElement);
47                        } // end if
48                    } // end if
49                } // end if
50            } // end for
51            // put the results into output array
52            if (out_aiResults != null && out_aiResults.length >= 4) {
53                out_aiResults[0] = iNumberCounter;     // numbers found
54                out_aiResults[1] = iEvenNumberCounter; // even numbers found
55                out_aiResults[2] = iWordCounter;       // words found
56                out_aiResults[3] = iEvenWordCounter;   // words with even number
57                bResult = true;                        // ...of characters found
58            } // end if
59        }
60        return bResult;
61    }
```