Softwaretechnik I

3. Semester

Prof. Dr. Ing. J. Matevska

HSB
Hochschule Bremen
City University of Applied Sciences

Laborversuch 5: Test

Gruppenmitglieder:

Laura Weischer, Bastian Silies, Florian Duchow

Datum: 23.01.18

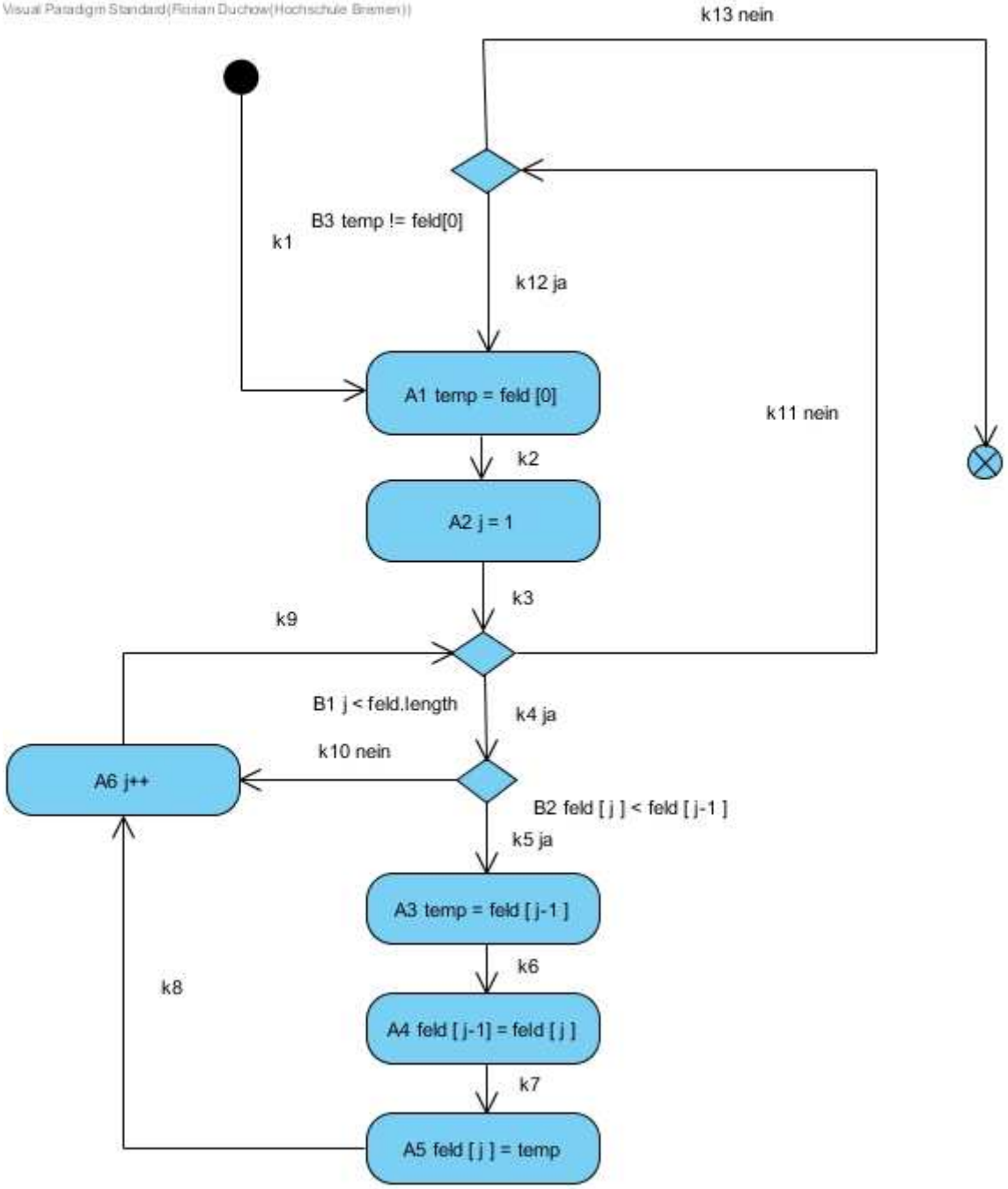
Inhaltsverzeichnis

Beschreibung der gegebenen Methode

Das gegebene Programm abc() sortiert das als Parameter übergebende Array. Dazu werden zwei Schleifen verschachtelt. Die innere Schleife läuft durch das gesamte Array. Dabei wird das aktuelle Feld mit dem vorherigen Feld verglichen. Wenn dieses größer ist als das aktuelle, werden die Werte der Felder mit einander vertauscht. Dazu wird der größere Wert in einer temporären Variable gespeichert. Danach wird der kleinere Wert der Beiden auf die Position vor dem aktuellen Feld gespeichert und der temporäre Wert in dem aktuellen Feld. Dies geschieht bis zum Letzten Feld des Arrays. Dadurch steht nach einem Schleifendurchgang der äußeren Schleife der größte Wert am Ende des Arrays. Dies wird so oft wiederholt, wie der temporäre Wert nicht dem Wert des Feldes an der 0. Stelle entspricht. Dieses Verfahren ähnelt sehr dem BubbleSort, bei dem die Werte wie in dem gegebenen Beispiel sortiert werden, allerdings ist die Bedingung der äußeren Schleife eine andere. Denn dort wird die äußere Schleife durch eine for-Schleife dargestellt, die von der letzten Position des Array bis zur ersten läuft. Dadurch werden die letzten Werte, die bereits sortiert sind nicht mehr mit in die Sortierung einbezogen.

White-Box Test

Kontrollflussgraph



Anweisungsüberdeckung und Kanten- Zweigüberdeckung

Eingabe	Ausgabe	Anweisungsabdeckung	Verhältnis Anweisungsabdeckung	Kantenabdeckung	Verhältnis Kantenabdeckung
672345	234567	A1, A2, A3, A4, A5, A6	100%	k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11,k12,k13	100%
268932	223689	A1, A2, A3, A4, A5, A6	100%	k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11,k12,k13	100%
123456	123456	A1,A2,A6	50%	k1,k2,k3,k4,k9,k10,k11,k13	62%
324523	223345	A1, A2, A3, A4, A5, A6	100%	k1,k2,k3,k4,k5,k6,k7,k8,k9,k10,k11,k12,k13	100%
111111	111111	A1,A2,A6	50%	k1,k2,k3,k4,k9,k10,k11,k13	62%

Anhand der Anweisungs- und Kantenüberdeckung ist zu erkennen, dass in dem vorliegenden Programm kein Programmteil vorliegt der nicht mindestens in einem Fall angesprochen wird. Für das vorliegende Beispiel ist zu erkennen, dass in 3 von 5 Fällen alle Anweisungen und Kanten verwendet werden. Die Beispiele die nicht die gesamte Abdeckung erbringen sind entweder und sortiert oder bestehen nur aus gleichen Werten.

Mehrfache Bedingungsüberdeckung

j	Feld.length	Feld[j]	Feld[j - 1]	Temp	Feld[0]	J < feld.length	feld[j] < feld[j - 1]	temp != feld[0]
1	6	1	9	1	9	T	T	T
1	6	0	2	2	2	T	T	F
1	6	8	3	6	3	T	F	T
1	6	6	4	4	4	T	F	F
6	6	-	2	9	5	F	T	T
6	6	-	6	6	6	F	T	F
6	6	-	3	2	7	F	F	T
6	6	-	9	8	8	F	F	F

Pfadüberdeckung

Bei einer Pfadabdeckung wird jeder mögliche Pfad von Start- bis zu Endpunkt beschreiben werden. Bei zwei verschachtelten Schleifen ist die Anzahl zu groß um jeden einzelnen Pfad aufzuschreiben. Deshalb wir jede Schleife nur einmal betreten und in dieser Schleife werden alle weiteren Pfade einmal betrachtet.

Testfall 1:

Die äußere und innere Schleife wird betreten und die If Anweisung ist true:

K1-k2-k3-k4-k5-k6-k7-k8-k9-k11-k13

Testfall 2:

Die äußere und innere Schleife wird betreten und die If Anweisung ist false:

K1-k2-k3-k4-k10-k9-k11-13

Testfall 3:

Die äußere Schleife wird betreten, die innere aber nicht:

K1-k2-k3-k11-k13

Black-Box-Test

Testplanung

Getestet werden soll die Methode abc() aus der Laboraufgabe „05 – Test“, die den Bubble-Sort implementiert.

Getestet wird mittels des Black-Box-Testverfahrens. Hierbei kommen Äquivalenzklassen, Grenzwertanalyse und Zufallstests zum tragen.

Als Testwerkzeug wird das Framework Junit verwendet.

Testdesign

Es sollen sechs Zahlen eingegeben werden. Alle nicht zufallsbasierten Tests werden daher mit Zahlen aus dem Zahlenbereich 1-6 ausgeführt

Äquivalenzklassen

1. Aufsteigend sortiert
    1. keine mehrfachen Werte
    2. eine Zahl mehrfach
    3. zwei Zahlen mehrfach
    4. drei Zahlen mehrfach
2. Absteigend sortiert
    1. keine mehrfachen Werte
    2. eine Zahl mehrfach
    3. zwei Zahlen mehrfach
    4. drei Zahlen mehrfach
3. Unsortiert
    1. keine mehrfachen Werte
    2. eine Zahl mehrfach
    3. zwei Zahlen mehrfach
    4. drei Zahlen mehrfach

Grenzwerte

Als Grenzwerte gelten jeweils die höchste und niedrigste Zahl der Zahlen des Testdatums und die Ränder des Arrays.

Zufallstest

Für den Test mit Zufallswerten, werden Werte aus dem Zahlenbereich 0-99 verwendet.

Testdaten

Äquivalenzklasse	**Werte**	**1. Testdatum**	**2. Testdatum**	**3. Testdatum**	**4. Testdatum**
1.1	Aufsteigend ohne mehrfache	1,2,3,4,5,6			
1.2	Aufsteigend, eine mehrfach	1,1,2,3,4,5	1,2,3,5,5,5	1,3,3,3,3,4,	3,3,3,3,3,3
1.3	Aufsteigend, zwei mehrfach	1,1,2,3,4,4	1,1,2,2,2,4	2,2,2,4,4,4	1,1,2,2,3,4
1.4	Aufsteigend, drei mehrfach	1,1,2,2,3,3			
2.1	Absteigend, ohne mehrfache	6,5,4,3,2,1			
2.2	Absteigend, eine mehrfach	6,6,5,4,3,2	6,5,4,3,3,3	6,5,5,5,5,4	
2.3	Absteigend, zwei mehrfach	6,6,5,4,3,3	6,6,5,5,5,4	6,6,6,5,5,5	6,5,5,4,3,3
2.4	Absteigend, drei mehrfach	6,6,5,5,4,4			
3.1	Unsortiert, ohne mehrfache	3,6,4,1,5,2			
3.2	Unsortiert, eine mehrfach	3,4,1,5,1,2	2,6,6,1,6,3	5,4,1,4,6,4	2,2,5,3,4,2
3.3	Unsortiert, zwei mehrfach	1,6,1,2,6,5	1,6,1,5,2,1	4,4,6,4,2,6	3,5,4,3,4,3
3.4	Unsortiert, drei mehrfach	6,3,1,6,3,1	2,2,6,6,1,1	5,4,5,3,3,4	

Testprotokoll

Jede Äquivalenzklasse wurde als eigene Testmethode programmiert

Der Zufallstest erzeugt zufällig ein Array mit sechs Werten. Dieses Array wird einmal über abc() sortiert und einmal über die java.util.Arrays-Methode sort(). Diese beiden Werte werden miteinander verglichen. Insgesamt wird dies 1.000.000 Mal wiederholt.

Fehlgeschlagen sind die Tests zu den Aquivalenzklassen 2.2, 2.3, 2.4, 3.3, 3.4.

Daraufhin wurde in der Testmethode zu Aquivalenzklasse 3.3 ein weiterer Testfall eingefügt (4,4,6,4,5,6) um den Fall zu testen, dass zwei gleiche Zahlen am Anfang des Arrays stehen, aber keine weitere kleinere Zahl im Array ist. Dieser Testfall ist jedoch erfolgreich durchlaufen.

Auch in Äquivalenzklasse 4 wurde ein weiterer Testfall eingefügt.

Im Anschluss wurde nur vereinzelt die Reihenfolge, in denen die Ergebnisse geprüft wurden leicht verändert um eine Vermutung, die in der Auswertung näher beschrieben wird, zu bestätigen.

Auswertung

Alle Testfälle der Äquivalenzklassen 1 sind erfolgreich gewesen, woraus sich schließen, lässt, dass bereits sortierte Arrays keine fehlerhaften Ausgaben erzeugen.

In den Äquivalenzklassen 2 sind alles Test der Äquivalenzklassen in denen Duplikate vorlagen, fehlgeschlagen, ebenso war dies bei zwei von drei Äquivalenzklassen der Äquivalenzklasse 3 der Fall. Ursache für das Scheitern dieser Tests waren jeweils Paare[1] der gleichen Zahl im Eingabe-Array, die keine kleineren Werte an einer Position davor stehen hatten, sowie eine einzelne oder ein Paar gleicher Zahlen, die kleiner sind als das erstgenannte Paar und nach diesem Paar im Array stehen.

[1]oder mehr gleiche aufeinander folgende Zahlen. Der Einfachheit halber auch zukünftig als Paare bezeichnet.