Softwarequalitätssicherung

Black-Box Testing

Gruppenmitglieder: Dennis Jongebloed
Julian Hinxlage
Johannes Theiner

Semester: Sommersemester 2020
letzte Änderung: 23. April 2020

1 Black-Box-Test

1.1 Eignung des Verfahrens

- Die Äquivalenzklassenbildung eignet sich nur für die Ausgabewerte, da die Eingans Variable nicht unabhängig voneinander sind. Außerdem wird keine hohe Zweigabdeckung erreicht.
- Die Grenzwertanalyse eignet sich, erreicht aber keine hohe Abdeckung, da die Variablen nicht unabhängig sind.
- Der Zustandsbezogene Test eignet sich nicht, da der zustand nicht bekannt ist.
- Die Ursachen-Wirkungs-Graph-Analyse/Entscheidungstabelle eignet sich nicht, da keine konkreten Werte festgelegt werden können.
- Der Anwendungsfallbasierte Test eignet sich nicht, da keine Anwendungsfallspezifikationen vorhanden ist.

1.2 Testfälle

Testfall	**Äquivalenzklasse**	**Ausgewählte Testdaten**		
		a	**b**	**c**
T1	NO_TRIANGLE	-1	1	2
T2	EQUILATERAL_TRIANGLE	2	2	2
T3	ISOSCELESS_TRIANGLE	2	2	3
T4	SCALENE_TRIANGLE	2	3	4

1.3 Entscheidungstabelle

Entscheidungstabelle		**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**T11**
Bedingungen												
	a <= 0	T	-	-	F	F	F	F	F	F	F	F
	b <= 0	-	T	-	F	F	F	F	F	F	F	F
	c <= 0	-	-	T	F	F	F	F	F	F	F	F
	a == b	-	-	-	-	-	-	T	T	F	F	F
	a == c	-	-	-	-	-	-	T	F	T	F	F
	b == c	-	-	-	-	-	-	T	F	F	T	F
	a + b > c	-	-	-	F	-	-	T	T	T	T	T
	a + c > b	-	-	-	-	F	-	T	T	T	T	T
	b + c > a	-	-	-	-	-	F	T	T	T	T	T
Aktionen												
	NO_TRIANGLE	T	T	T	T	T	T	F	F	F	F	F
	EQUILATERAL_TRIANGLE	F	F	F	F	F	F	T	F	F	F	F
	ISOSCELESS_TRIANGLE	F	F	F	F	F	F	F	T	T	T	F
	SCALENE_TRIANGLE	F	F	F	F	F	F	F	F	F	F	T

2 Unit-Tests

```java
public class EquivalenceClassTest {

    @Test
    public void noTriangle() {
        TriangleChecker tc = new TriangleChecker();
        assertEquals(TriangleType.NO_TRIANGLE, TriangleChecker.checkTriangle(-1, 1, 2));
    }

    @Test
    public void equilateralTriangle() {
        TriangleChecker tc = new TriangleChecker();
        assertEquals(TriangleType.EQUILATERAL_TRIANGLE, TriangleChecker.checkTriangle(2, 2, 2));
    }

    @Test
    public void isocelessTriangle() {
        TriangleChecker tc = new TriangleChecker();
        assertEquals(TriangleType.ISOSCELESS_TRIANGLE, TriangleChecker.checkTriangle(2, 2, 3));
    }

    @Test
    public void scaleneTriangle() {
        TriangleChecker tc = new TriangleChecker();
        assertEquals(TriangleType.SCALENE_TRIANGLE, TriangleChecker.checkTriangle(2, 3, 4));
    }

}
```

3 Implementierung

```
public class TriangleChecker {

    public static TriangleType checkTriangle(int a, int b, int c) {
        if (a <= 0 || b <= 0 || c <= 0)
            return TriangleType.NO_TRIANGLE;

        if ((long)a + (long)b > (long)c &&
            (long)a + (long)c > (long)b &&
            (long)b + (long)c > (long)a) {
            if (a == b && a == c)
                return TriangleType.EQUILATERAL_TRIANGLE;

            if (a == b || a == c || b == c)
                return TriangleType.ISOSCELESS_TRIANGLE;

            return TriangleType.SCALENE_TRIANGLE;
        }


        return TriangleType.NO_TRIANGLE;
    }

}
```