Exercise Sheet 4

Birgit Pohl 574353 (MO. 9-11)
Kevin Trogant 572451 (Mo. 15-17)
Ronja Enseleit 572404 (Mo. 15-17)
Dustin Hebecker 571271 (MO. 9-11)

Group: *Iron Man*

20. November 2016

Team: *Iron Man*
B. Pohl, K. Trogant, R. Enseleit, D. Hebecker 20. November 2016

Aufgabe 1

Der Akzeptanztest zielt darauf ab beim Kunden Vertrauen in das Produkt zu erzeugen, während der Funktionstest überprüft ob das entwickelte Produkt den Anforderungen der Anforderungsdefinition entspricht.
Im Gegensatz zum Funktionstest wird der Akzeptanztest vom Kunden in der Umgebung durchgeführt, in der das Produkt eingesetzt werden soll. Da vor dem Akzeptanztest bereits durch den Funktionstest sichergestellt wurde, dass das Produkt der Anforderungsdefinition entspricht, sollte im Akzeptanztest keine Abweichung vom definierten Verhalten mehr entdeckt werden. Das heißt, im Akzeptanztest wird vor allem überprüft ob das Produkt auf die Weise benutzt werden kann, die sich der Kunde vorgestellt hat.
Schlägt der Funktionstest fehl, heißt das, dass das Produkt eine funktionale Anforderung nicht erfüllt (also zum Beispiel eine falsche Ausgabe liefert). Schlägt der Akzeptanztest fehl, heißt das, dass das Produkt vom Kunden nicht benutzt werden kann (zB. weil die Bedienung zu umständlich ist).

Aufgabe 2

Bildung von gültigen und ungültigen Äquivalenzklassen:

Eingabe	gültige Äquivalenzklassen	ungültige Äquivalenzklassen
1. String	1) String aus Buchstaben und/oder Ziffern, der der gleiche String wie der 2. String ist 2) String aus Buchstaben und/oder Ziffern, der ein echter Substring von dem 2. String ist 3) String aus Buchstaben und/oder Ziffern, der kein Substring von dem 2. String ist 4) String mit Sonderzeichen, der der gleiche String wie der 2. String ist 5) String mit Sonderzeichen, der ein echter Substring von dem 2. String ist 6) String mit Sonderzeichen, der kein Substring von dem 2. String ist 7) Leerer String 8) String mit Leerzeichen (vlt. auch Sonderzeichen), der der gleiche String wie der 2. String ist 9) String mit Leerzeichen (vlt. auch Sonderzeichen), der ein echter Substring von dem 2. String ist 10) String mit Leerzeichen (vlt. auch Sonderzeichen), der kein Substring von dem 2. String ist	11) Kein String
2. String	12) String aus Buchstaben und/oder Ziffern, der der gleiche String wie der 1. String ist 13) String aus Buchstaben und/oder Ziffern, der ein echter Substring von dem 1. String ist 14) String aus Buchstaben und/oder Ziffern, der kein Substring von dem 1. String ist 15) String mit Sonderzeichen, der der gleiche String wie der 1. String ist 16) String mit Sonderzeichen, der ein echter Substring von dem 1. String ist 17) String mit Sonderzeichen, der kein Substring von dem 1. String ist 18) Leerer String 19) String mit Leerzeichen (vlt. auch Sonderzeichen), der der gleiche String wie der 1. String ist 20) String mit Leerzeichen (vlt. auch Sonderzeichen), der ein echter Substring von dem 1. String ist 21) String mit Leerzeichen (vlt. auch Sonderzeichen), der kein Substring von dem 1. String ist	22) Kein String

Aquivalenzklassentestschema:

Strings sind hier der Übersichtlichkeit halber von Anführungszeichen umrandet.

Testfall-nummer	Getestete Äquivalenzklassen	Eingabe 1	Eingabe 2	Ergebnis/Kommentar
0.	1, 12	„0d3r“	„0d3r“	Kein Substring
1.	2, 14	„f5jk“	„ghf5jk79“	1. String ist ein Substring des 2.
2.	2, 14	„allo“	„hallo“	1. String ist ein Substring des 2.
3.	2, 14	„hall“	„hallo“	1. String ist ein Substring des 2.
4.	2, 14	„a“	„hallo“	1. String ist ein Substring des 2.
5.	2, 17	„allo“	„%hallo+“	1. String ist ein Substring des 2.
6.	2, 21	„Hallo“	„Hallo Welt!“	1. String ist ein Substring des 2.
7.	3, 13	„ghf5jk79“	„hf5“	2. String ist ein Substring des 1.
8.	3, 14	„ghf5jk79“	„796g“	Kein Substring
9.	3, 17	„ghf5jk79“	„hf5&“	Kein Substring
10.	3, 18	„ghf5jk79“	„“	Kein Substring
11.	3, 21	„ghf5jk79“	„1 0h“	Kein Substring
12.	3, 22	„ghf5jk79“	42	Falsche Eingabe
13.	4, 15	„Hallo!“	„Hallo!“	Kein Substring
14.	5, 17	„Hallo!“	„Hallo!!!“	1. String ist ein Substring des 2.
15.	5, 21	„!Hallo“	„!Hallo Welt!“	1. String ist ein Substring des 2.
16.	6, 13	„Hallo!“	„Hallo“	2. String ist ein Substring des 1.
17.	6, 14	„Hallo!“	„hallo0“	Kein Substring
18.	6, 16	„Hallo,d“	„Hallo,“	2. String ist ein Substring des 1.
19.	7, 17	„“	„1%“	Kein Substring
20.	7, 22	„“	1.56	Falsche Eingabe
21.	8, 19	„Hallo Welt!“	„Hallo Welt!“	Kein Substring
22.	9, 21	„Hallo “	„Hallo Welt!“	1. String ist ein Substring des 2.
23.	10, 20	„Hallo Welt!“	„Hallo “	2. String ist ein Substring des 1.
24.	11, 14	127	„127“	Falsche Eingabe
25.	11, 22	1337	1337	Falsche Eingabe

Es ergeben sich 25 Tests. Diese Tests können alle für den Funktionstest verwendet werden, da sie die Funktionalität eines bestimmten Moduls testen.

Aufgabe 3

Äquivalenzklassen für den Modi "Bearbeitung"

Eingabe	gültige Äquivalenzklassen	ungültige Äquivalenzklassen
Stadt	1) Noch nicht im System	2) Schon im System 3) Leerer Wert 4) Numerischer Wert
Route Zeit	5) > 0	6) < 0 7) 0 8) Leer 9) Alphabetisch
Route Distanz	10) > 0	11) < 0 12) 0 13) Leer 14) Alphabetisch
Route Kosten	15) Numerisch	16) Alphabetisch 17) Leer
Route Stadt 1/2	18) 2 Unterschiedliche Städte aus dem Sytstem	19) Leer 20) 2 Identische Städte 21) Stadt nicht im System 22) Numerischer Wert

Tests für die Äquivalenzklassen des Modi "Bearbeitung"

Testfall-nummer	Getestete Äquivalenzk.	Eingaben	Ergebnis/Kommentar
1	1	Berlin	Erste Stadt im System
2	2	Berlin	Fehler Duplikat
3	3	„ “	Fehler leerer String
4	4	1234	Fehler ungültiger Wert
5	5, 10, 15, 18	T=1,S=100,V=10,C1=Köln,C2=Bonn	Erfolg, angenommen Die Städte sind im System
6	6, 10, 15, 18	T=-1,S=100,V=10,C1=Köln,C2=Bonn	Fehler negative Zeit
7	7, 10, 15, 18	T=0,S=100,V=10,C1=Köln,C2=Bonn	Fehler Instantane Reise
8	8, 10, 15, 18	T=,S=100,V=10,C1=Köln,C2=Bonn	Fehler Keine Zeit gegeben
9	9, 10, 15, 18	T=abc,S=100,V=10,C1=Köln,C2=Bonn	Fehler falscher Zeitwert
10	5, 11, 15, 18	T=1,S=-1,V=10,C1=Köln,C2=Bonn	Fehler negative Distanz
11	5, 12, 15, 18	T=1,S=0,V=10,C1=Köln,C2=Bonn	Fehler Städte haben keinen Abstand
12	5, 13, 15, 18	T=1,S=,V=10,C1=Köln,C2=Bonn	Fehler kein Städteabstand gegeben
13	5, 14, 15, 18	T=1,S=abc,V=10,C1=Köln,C2=Bonn	Fehler falscher Abstandswert
14	5, 10, 16, 18	T=1,S=100,V=abc,C1=Köln,C2=Bonn	Fehler falscher Preiswert
15	5, 10, 17, 18	T=1,S=100,V=,C1=Köln,C2=Bonn	Fehler kein Preis gegeben
16	5, 10, 15, 19	T=1,S=100,V=10,C1=,C2=Bonn	Fehler ein oder beide Städte nicht angegeben (beides testen)
17	5, 10, 15, 20	T=1,S=100,V=10,C1=Köln,C2=Köln	Fehler Start und Ziel identisch
18	5, 10, 15, 21	T=1,S=100,V=10,C1=Köln,C2=Moon	Fehler eine oder beide Städte nicht im System
19	5, 10, 15, 22	T=1,S=100,V=10,C1=Köln,C2=123	Fehler falscher Städtewert

T=Zeit
S=Distanz
V=Kosten
C=Stadt
F=Fehler

Äquivalenzklassen für den Modi "Abfrage"

Alle folgenden Klassen gibt es für jede Abfrage (Zeit, Distanz, Kosten, Agony). Weiterhin wird angenommen die Abfragen sind so gestalltet, dass die Korrekte Antowrt bekannt ist bzw. das System expliziet für diese Abfrage gestaltet wurde.

Eingabe	gültige Äquivalenzklassen	ungültige Äquivalenzklassen
2 Städte	1) Beide im System 2) 2 oder Mehr routen möglich. 3) Kürzeste Route führt über mehrere/alle anderen Städte im System	4) Nicht im System 5) Start und Ziel identisch 6) Leere Abfrage 7) Es existiert keien Verbindung zwischen den Eingaben

Tests für die Äquivalenzklassen des Modi "Abfrage"

Im folgenden wird angenommen es existiert ein System mit folgenden Städten:

- Berlin
- Bonn
- Köln
- Düsseldorf
- New York

Als Agony wird der Einfachheit halber die Summe aus Zeit und Kosten verwendet. Das Produkt wäre jedoch realistischer.
Weiterhin werden folgende verbindungen angenommen:

1. Berlin - Bonn , 5h, 500 km, 5 Euro, Agony 10
2. Berlin - Düsseldorf , 5h, 500 km, 5 Euro, Agony 10
3. Berlin - Köln , 4h, 400 km, 4 Euro, Agony 8
4. Köln - Bonn , 1h, 100 km, 1 Euro, Agony 2
5. Köln - Düsseldorf , 10h, 1000 km, 10 Euro, Agony 20

Testfall-nummer	Getestete Äquivalenzk.	Eingaben	Ergebnis/Kommentar
1	1	Köln - Bonn	Take route 4
2	2, (3)	Berlin - Bonn	Take route 1 or (3 and 4)
3	3	Köln - Düsseldorf	Take route 3 and 2
4	4	Bielefeld - Berlin	Fehler Bielefeld existiert nicht
5	5	Berlin - Berlin	Fehler Start und Ziel identisch
6	6	- Berlin	Fehler leerer Startort
7	7	Berlin - New York	Fehler es gibt keine Verbindung

Im allgemeinen sollte dieser Test noch etwas ausfühlerlicher sein und z.B. Strecken mit $1 - n$ routen testen, welche sich bei einem schlechten Programierstiel utnerschiedlich verhalten könnten. Angenommen es wird sich an die gängigsten und sinnvollsten Techniken gehalten, sollte dieser Test jedoch ausreichen. Insbesondere zum test der Wegfindung (möglicherweise stochastischer Algorithmus)

sollten dynamische verschiedene Modelle erstellt werden mit bekannten Lösungen und getestet werden. Ein sollcher Test sollte jedoch ohne Hintergrundwissen über die Code Struktur des Projektes erstellt werden (blind test).