Black Box Testing

Mahtab EzzatiKarami

Table of Contents

Bank.FeesCalculater

The focus of this class is to check the correct calculation of fees while performing withdrawal, deposit, and transfer.

Withdrawal

The test cases for the withdrawal portion of the ATM program were derived by using Robust Worst-Case Boundary Value Analysis. The equivalence classes are as follows:

Variable	Different Values			
accountBalance	<0	[0-1000]	(1000-10000]	>10000
Student	True	False		
DayOfWeek	Saturday	Sunday	Any other day	

The variable values used for the test cases include:

1. Student
    a. True
    b. False

2. DayOfWeek
    a. Saturday
    b. Sunday
    c. Any other day

3. AccountBalance
    a. Below the minimum: -1
    b. Minimum: 0
    c. Above the minimum: 1
    d. Medium: 500
    e. Below the maximum: 999
    f. Maximun: 1000
    g. Above the maximum: 1001
    h. Medium: 5000
    i. Below the maximum: 9999
    j. Maximum: 10000
    k. Above the maximum: 10001

Test Cases

The tests are as follows:

Test Case#	AccountBalance	Student	DayOfWeek
1	-1	T	Saturday
2	0	T	Saturday
3	1	T	Saturday
4	500	T	Saturday
5	999	T	Saturday
6	1000	T	Saturday
7	1001	T	Saturday
8	5000	T	Saturday
9	9999	T	Saturday
10	10000	T	Saturday
11	10001	T	Saturday
12	-1	T	Wednesday
13	0	T	Wednesday
14	1	T	Wednesday
15	500	T	Wednesday
16	999	T	Wednesday
17	1000	T	Wednesday
18	1001	T	Wednesday
19	5000	T	Wednesday
20	9999	T	Wednesday
21	10000	T	Wednesday
22	10001	T	Wednesday
23	-1	T	Sunday

24	0	T	Sunday
25	1	T	Sunday
26	500	T	Sunday
27	999	T	Sunday
28	1000	T	Sunday
29	1001	T	Sunday
30	5000	T	Sunday
31	9999	T	Sunday
32	10000	T	Sunday
33	10001	T	Sunday
34	-1	F	Saturday
35	0	F	Saturday
36	1	F	Saturday
37	500	F	Saturday
38	999	F	Saturday
39	1000	F	Saturday
40	1001	F	Saturday
41	5000	F	Saturday
42	9999	F	Saturday
43	10000	F	Saturday
44	10001	F	Saturday
45	-1	F	Wednesday
46	0	F	Wednesday
47	1	F	Wednesday
48	500	F	Wednesday
49	999	F	Wednesday

50	1000	F	Wednesday
51	1001	F	Wednesday
52	5000	F	Wednesday
53	9999	F	Wednesday
54	10000	F	Wednesday
55	10001	F	Wednesday
56	-1	F	Sunday
57	0	F	Sunday
58	1	F	Sunday
59	500	F	Sunday
60	999	F	Sunday
61	1000	F	Sunday
62	1001	F	Sunday
63	5000	F	Sunday
64	9999	F	Sunday
65	10000	F	Sunday
66	10001	F	Sunday

Results

The Junit test results are as follows:

Test Case#	Test Result
1	Passed
2	Passed
3	Passed
4	Passed
5	Passed
6	Passed
7	Passed
8	Passed
9	Passed
10	Passed
11	Passed
12	Passed
13	Passed
14	Passed
15	Passed
16	Passed
17	Passed
18	Passed
19	Passed
20	Passed
21	Passed
22	Passed
23	Passed

24	Passed
25	Passed
26	Passed
27	Passed
28	Passed
29	Passed
30	Passed
31	Passed
32	Passed
33	Passed
34	Passed
35	Passed
36	Passed
37	Passed
38	Passed
39	Passed
40	Passed
41	Passed
42	Passed
43	Passed
44	Passed
45	Passed
46	Passed
47	Passed
48	Passed
49	Passed

50	Passed
51	Passed
52	Passed
53	Passed
54	Passed
55	Passed
56	Passed
57	Passed
58	Passed
59	Passed
60	Passed
61	Passed
62	Passed
63	Passed
64	Passed
65	Passed
66	Passed

Deposit

The test cases for the deposit portion of the ATM program were derived by using Weak Robust Equivalence Class Analysis. The cases for valid input were chosen so that for each variable, one value from each invalid equivalence class was selected. So that in each case, only one of the variables had an invalid value and the rest were valid. This resulted in eight test cases. Four of them were for valid inputs and others were for invalid inputs. The following assumptions were made:

The accountBalance variable has a maximum value of 15,000 and the amount variable has a maximum value of 1000, and both variables have a minimum value of zero.

The equivalence classes are as follows:

Variable	Equivalence Classes					
Amount	<0	[0-100]	(100-500]	(500-1000]	>1000	
AccountBalance	<0	[0-1000]	(1000-5000]	(5000-10000]	(10000-15000]	>15000
Student	True	False				

The variable values used for the test cases include:

1. Amount
    a. Class: <0 => -5
    b. Class: [0-100] => 20
    c. Class: (100-500] => 150
    d. Class: (500-1000] => 700
    e. Class: >1000 => 1300

2. AccountBalance
    a. Class: <0 => -5
    b. Class: [0-1000] => 500
    c. Class: (1000-5000] => 3000
    d. Class: (5000-10000] => 8000
    e. Class: (10000-15000] => 12000
    f. >15000 => 16000

3. Student
    a. True
    b. False

Test Cases

The tests are as follows:

Test Case#	amount	accountBalance	student
1	20	500	T
2	150	3000	F
3	700	8000	F
4	20	12000	F
5	-5	500	F
6	1300	500	F
7	20	-5	F
8	20	16000	F

Results

The Junit test results are as follows:

Test Case#	Test Result
1	Pass
2	Pass
3	Pass
4	Pass
5	Pass
6	Pass
7	Pass
8	Pass

Transfer

The test cases for the transfer portion of the ATM program were derived by using Decision Table Analysis. For each of the rows in the decision table provided, there is a test case. So, in total, we have 16 test cases.

The decision table is as follows:

Condition	Case1	Case2	Case3	Case4	Case5	Case6	Case7	Case8
Amount	<100	<100	<100	<100	>=100	>=100	>=100	>=100
FromAccountBalance	<1000	<1000	>=1000	>=1000	<1000	<1000	>=1000	>=1000
ToAccountBalance	<1000	>=1000	<1000	>=1000	<1000	>=1000	<1000	>=1000
Student	T	T	T	T	T	T	T	T

Condition	Case9	Case10	Case11	Case12	Case13	Case14	Case15	Case16
Amount	<100	<100	<100	<100	>=100	>=100	>=100	>=100
FromAccountBalance	<1000	<1000	>=1000	>=1000	<1000	<1000	>=1000	>=1000
ToAccountBalance	<1000	>=1000	<1000	>=1000	<1000	>=1000	<1000	>=1000
Student	F	F	F	F	F	F	F	F

The variable values used for the test cases include:

1. Amount
    a. Class: <100 => 90
    b. Class: >=100 => 100

2. fromAccountBalance
    a. Class: <1000 => 900
    b. Class: >=1000 => 1000

3. toAccountBalance
    a. Class: <1000 => 900
    b. Class: >=1000 => 1000

4. Student
    a. True
    b. False

Test cases

The tests are as follows:

Test Case#	amount	fromAccountBalance	toAccountBalance	Student
1	90	900	900	T
2	90	900	1000	T
3	90	1000	900	T
4	90	1000	1000	T
5	100	900	900	T
6	100	900	1000	T
7	100	1000	900	T
8	100	1000	1000	T
9	90	900	900	F
10	90	900	1000	F
11	90	1000	900	F
12	90	1000	1000	F
13	100	900	900	F
14	100	900	1000	F
15	100	1000	900	F
16	100	1000	1000	F

Results

The Junit test results are as follows:

Test Case#	Test Result
1	Passed
2	Passed
3	Passed
4	Passed
5	Passed
6	Passed
7	Passed
8	Passed
9	Passed
10	Passed
11	Passed
12	Passed
13	Passed
14	Passed
15	Passed
16	Failed

ATM.Session

The focus for this class is to check for the correct calculation of fees while performing withdrawal.

Withdrawal

The test cases for the withdrawal portion of the ATM program were derived by using Robust Worst-Case Boundary Value Analysis. The equivalence classes are as follows:

Variable	Different Values		
Pin	3 digits	4 digits	5 digits
Amount	<20	[20-1000]	>1000

The variable values used for the test cases include:

1. Pin
    a. 3 digit: 153 =>invalid
    b. 4 digit: 1534 => valid
    c. 5 digit: 15346 => invalid

2. Amount
    a. Below the minimum: 19
    b. Minimum: 20
    c. Above the minimum: 21
    d. Medium: 500
    e. Below the maximum: 999
    f. Maximum: 1000
    g. Above the maximum: 1001

We also test other valid amounts of products of 20 and 50 between 20 and 100 and invalid amounts of 70 and 90.

    a. 40: valid
    b. 50: valid
    c. 60: valid
    d. 70: invalid
    e. 80: valid
    f. 90: invalid
    g. 100: valid

For tests with 4-digit pin numbers, we were expecting invalidAmountException for the ones with invalid amount and no exceptions for the ones with amounts which were product of 20 or 50. But all of these 14 tests failed with invalidPINException. It shows that 4-digit pins are not working. For the other 14 tests for which we used 3 digit pins, we were expecting invalidPINException. All of them failed because 3-digit pins are actually working fine with the code given to use. So none of them had invalidPINException.

The only 14 tests for which the results were as expected were the ones with 5-digit pins. For these tests, we expected invalidPINExceptin and that was exactly the case. For the reasons mentioned above, the first 28 tests failed, and we designed extra 14 tests for which we assume 3-digit pins are legal. So we do not expect invalidPINException. Instead we directly test the amount values.

P.S. We did not have to do these extra tests, but this way it is easier and more clear whether there is anything wrong with amount values or not.

The results show that for valid amounts of 60 and 80, the program should work fine, because these values are products of 20. But the tests show they are invalid amounts. So it seemed like the program is working fine with the amount more than 20 which are also products of 50 but not product of 20. So we also check for other amounts such as 70 and 90. The program should have an InvalidAmountException but unexpectedly it is fine with these two invalid values. So the reason is because it first deducts products of 50 from the amount. If the rest is product of 20 then it is accepted as valid. If it is not, then it is counted as an invalid amount. For example, when you deduct 50 from 60, 10 is not a product of 20. The same goes with 80. That is why they have InvalidAmountException. For 70 and 90, when we deduct 50, the rest is a product of 20 and that is why they are counted as valid amounts although they are not.

Test Cases

The tests are as follows:

Test Case#	Pin	Amount	Expected Result
1	1534	19	InvalidAmountException
2	1534	20	No Exception
3	1534	21	InvalidAmountException
4	1534	500	No Exception
5	1534	999	InvalidAmountException
6	1534	1000	No Exception
7	1534	1001	InvalidAmountException
8	1534	40	No Exception
9	1534	50	No Exception
10	1534	60	No Exception
11	1534	70	InvalidAmountException
12	1534	80	No Exception
13	1534	90	InvalidAmountException
14	1534	100	No Exception
15	153	19	InvalidPINException
16	153	20	InvalidPINException
17	153	21	InvalidPINException
18	153	500	InvalidPINException
19	153	999	InvalidPINException
20	153	1000	InvalidPINException
21	153	1001	InvalidPINException
22	153	40	InvalidPINException

23	153	50	InvalidPINException
24	153	60	InvalidPINException
25	153	70	InvalidPINException
26	153	80	InvalidPINException
27	153	90	InvalidPINException
28	153	100	InvalidPINException
29	15346	19	InvalidPINException
30	15346	20	InvalidPINException
31	15346	21	InvalidPINException
32	15346	500	InvalidPINException
33	15346	999	InvalidPINException
34	15346	1000	InvalidPINException
35	15346	1001	InvalidPINException
36	15346	40	InvalidPINException
37	15346	50	InvalidPINException
38	15346	60	InvalidPINException
39	15346	70	InvalidPINException
40	15346	80	InvalidPINException
41	15346	90	InvalidPINException
42	15346	100	InvalidPINException
43	153	19	InvalidAmountException
44	153	20	No Exception
45	153	21	InvalidAmountException
46	153	500	No Exception
47	153	999	InvalidAmountException
48	153	1000	No Exception

49	153	1001	InvalidAmountException
50	153	40	No Exception
51	153	50	No Exception
52	153	60	No Exception
53	153	70	InvalidAmountException
54	153	80	No Exception
55	153	90	InvalidAmountException
56	153	100	No Exception

Results

The Junit test results are as follows:

Test Case#	Test Result
1	Failed
2	Failed
3	Failed
4	Failed
5	Failed
6	Failed
7	Failed
8	Failed
9	Failed
10	Failed
11	Failed
12	Failed
13	Failed
14	Failed
15	Failed
16	Failed
17	Failed
18	Failed
19	Failed
20	Failed
21	Failed
22	Failed
23	Failed

24	Failed
25	Failed
26	Failed
27	Failed
28	Failed
29	Passed
30	Passed
31	Passed
32	Passed
33	Passed
34	Passed
35	Passed
36	Passed
37	Passed
38	Passed
39	Passed
40	Passed
41	Passed
42	Passed
43	Passed
44	Passed
45	Passed
46	Passed
47	Passed
48	Passed
49	Passed

50	Passed
51	Passed
52	Failed
53	Failed
54	Failed
55	Failed
56	Passed