CS 589: SOFTWARE TESTING AND ANALYSIS

Final Project Report
Fall 2013

Submitted By,
Nickhil Revu
A20299434

Content:

1. Introduction:

The main aim of this project is to develop a set of transactions available in a Banking System with respect to the given EFSM model diagram and the account class, also to test the program using Pair transactions testing, Default Transaction and also using Multiple condition testing to test the predicates of multiple condition statements in operation.

2. Model Based Testing

Testing of all Transition Pairs

States in FSM of Account:

- Idle
- Check Pin
- Ready
- Overdrawn
- Locked

Transition Pairs in EFSM of Account

A. Idle State:

Incoming: T1, T5, T6, T7, T9, T10

Outgoing: T2, T7

Pairs: (T1,T2); (T5,T2); (T6,T2); (T7,T2); (T9,T2); (T10,T2)
(T1,T7); (T5,T7); (T6,T7); (T7,T7); (T9,T7); (T10,T7)

B. Check Pin State:

Incoming: T2, T3

Outgoing: T3, T4, T5, T6, T8

Pairs: (T2,T3); (T2,T4); (T2,T5); (T2,T6); (T2,T8)
(T3,T3); (T3,T4); (T3,T5); (T3,T6); (T3,T8)

C. Ready State:

Incoming: T4, T11, T12, T13, T15, T17

Outgoing: T10, T11, T12, T13, T14, T16

Pairs: (T4,T10); (T4,T11); (T4,T12); (T4,T13); (T4,T14); (T4,T16)
(T11,T10); (T11,T11); (T11,T12); (T11,T13); (T11,T14); (T11,T16)
(T12,T10); (T12,T11); (T12,T12); (T12,T13); (T12,T14); (T12,T16)
(T13,T10); (T13,T11); (T13,T12); (T13,T13); (T13,T14); (T13,T16)
(T15,T10); (T15,T11); (T15,T12); (T15,T13); (T15,T14); (T15,T16)
(T17,T10); (T17,T11); (T17,T12); (T17,T13); (T17,T14); (T17,T16)

D. **Overdrawn State:**

Incoming: T8, T14, T19, T21, T22

Outgoing: T9, T15, T20, T21, T22

Pairs: (T8,T9); (T8,T15); (T8,T20); (T8,T21); (T8,T22)
(T14,T9); (T14,T15); (T14,T20); (T14,T21); (T14,T22)
(T19,T9); (T19,T15); (T19,T20); (T19,T21); (T19,T22)
(T21,T9); (T21,T15); (T21,T20); (T21,T21); (T21,T22)
(T22,T9); (T22,T15); (T22,T20); (T22,T21); (T22,T22)

E. **Locked State:**

Incoming: T16, T18, T20

Outgoing: T17, T18, T19

Pairs: (T16,T17); (T16,T18); (T16,T19)
(T18,T17); (T28,T18); (T18,T19)
(T20,T17); (T20,T18); (T20,T19)

Pair Transition Testing

TEST #	**PAIRS COVERED**
TEST #1	(T1,T7); (T7,T7); (T7,T2); (T2,T3); (T3,T4); (T4,T10); (T10,T2); (T2,T5)
TEST#2	(T1,T2); (T2,T8); (T8,T21); (T21,T22); (T22,T15); (T15,T10); (T10,T7)
TEST #3	(T5,T7); (T8,T9); (T9,T7)
TEST #4	(T5,T2); (T3,T6); (T6,T7); (T6,T2)
TEST #5	(T9,T2); (T8,T22); (T22,T9)
TEST #6	(T3,T5); (T3,T8); (T22,T22); (T22,T21); (T21,T21); (T21,T9)
TEST #7	(T8,T15); (T15,T12) (T12,T14); (T14,T9)
TEST #8	(T2,T4); (T4,T11); (T11,T12); (T12,T3); (T13,T16); (T16,T18); (T18,T17); (T17,T14); (T14,T22); (T15,T11); (T11,T13); (T13,T11); (T11,T16); (T16,T17); (T17,T13); (T13,T10)
TEST #9	(T4,T14); (T14,T20); (T20,T18); (T18,T19); (T19,T20); (T20,T19); (T19,T9)
TEST #10	(T4,T12); (T12,T11); (T12,T10)
TEST #11	(T4,T13); (T13,T13); (T13,T12); (T12,T16); (T17,T10)
TEST #12	(T4,T16); (T17,T16); (T17,T11); (T11,T11)
TEST #13	(T12,T12); (T13,T14); (T14,T21); (T21,T15); (T15,T16); (T17,T12)
TEST #14	(T11,T14); (T14,T15); (T15,T13)
TEST #15	(T21,T20); (T19,T22); (T22,T20); (T19,T21)
TEST #16	(T19,T15); (T15,T14)

Non Executable Pairs:

Check Pin State:

(T2,T6)

(Incoming) T2: login(x) [y==id] / attempts=0

(Outgoing) T6: pin(x) [(x!=pn) && (attempts==1)]

Reason : For an incoming edge T2 attempts is set to value 0 and the very next step in incorrect pin the value of attempts should be 1, That is not possible in the given EFSM model.

Check Pin State:

(T3,T3)

(Incoming) T3: pin(x)[(x!=pn)&&(attempts<1)] /attempts++

(Outgoing) T3: pin(x)[(x!=pn)&&(attempts<1)] /attempts++

Reason : For an incoming edge T3 attempts is set to value 1 (attempts is 0 initially and is incremented by 1)and the very next step in incorrect pin the value of attempts should be less than 1, That is not possible in the given EFSM model.

Locked State:

(T16,T19)

(Incoming) T16: lock(x)[x==pn]

(Outgoing) T19: unlock(x)[(x==pn)&&(b<=999)]

Reason : The incoming edge T16 is coming from a Ready state which means the balance is greater than 999 and the outgoing transition T19 is going to Overdrawn state where the balance is <=999. This Transition pair is not executed because the value of balance is unchanged in Locked state

Locked State:

(T20,T17)

(Incoming) T20: lock(x)[x==pn]

(Outgoing) T17: unlock(x)[(x==pn)&&(b>999)]

Reason : The incoming edge T16 is coming from a Ready state which means the balance is greater than 999 and the outgoing transition T19 is going to Overdrawn state where the balance is <=999. This Transition pair is not executed because the value of balance is unchanged in Locked state

3. Default Transitions:

Default Transitions are the transition which when executed does not affect the state in the EFSM model.

Default Transition Testing

State	Default Transitions	Test Case
Start state	login(x); pin(x); deposit(d); withdraw(w); balance(); lock(x); unlock(x); logout()	TEST #17
Idle State	open(x,y,z); pin(x); deposit(d); withdraw(w); balance(); lock(x); unlock(x); logout()	TEST #18
Check Pin	open(x,y,z); login(x); deposit(d); withdraw(w); balance(); lock(x); unlock(x)	TEST #19
Ready state	open(x,y,z); login(x); pin(x); deposit(d);withdraw(w); balance(); unlock(x)	TEST #20
Overdrawn state	open(x,y,z); login(x); pin(x); withdraw(w); deposit(d); balance(); unlock(x)	TEST #21
Locked State	open(x,y,z); login(x); pin(x); deposit(d); withdraw(w); lock(x); logout()	TEST #22

4. Multiple condition Testing

Testing all the conditional Statements

Line 14: if ((x > 0) && (x4 == -1))

(x>0)	(X4==-1)	TEST CASE
T	T	TEST #1
T	F	**Not Executable:** x4 is initialized to -1 and that cannot be false in the first statement. However if we execute logout we can change the value of x4 to 0 and later we may execute open method with initial value > 0. But Test Suite cannot accept a logout operation prior to open, So this branch is not executable.
F	T	TEST #23
F	F	**Not Executable:** x4 is initialized to -1 and that cannot be false in the first statement. However if we execute logout we can change the value of x4 to 0 and later we may execute open method with initial value 0. But Test Suite cannot accept a logout operation prior to open, So this branch is not executable.

Line 26: if (x4!=2)

(X4==-1)	TEST CASE
T	TEST #24
F	TEST #10

Line 30: if (x2!=0)

(X4==-1)	TEST CASE
T	TEST #19
F	TEST #10

Line 34: if ((x1 <x7) && (d>0)

(x1 < x7)	(d>0)	TEST CASE
T	T	TEST #2
T	F	TEST #26
F	T	TEST #10
F	F	TEST #27

Line 41: if (d>0)

(d>0)	TEST CASE
T	TEST #10
F	TEST #27

Line 51: if (x2!=0)

(X4!=2)	TEST CASE
T	TEST #28
F	TEST #10

Line 55: if (x2!=0)

(X4==-1)	TEST CASE
T	TEST #29
F	TEST #10

Line 59: if ((x1 > w) && (w>0)

(x1 > w)	(w>0)	TEST CASE
T	T	TEST #10
T	F	TEST #30
F	T	TEST #31
F	F	**Not Executable:** Both (x1>w) and (w>0) cannot be false at the same time, because x1 is account balance which is always a positive number and if (w>0) is false means w is negative and x1>w means x1 is also negative and smaller than w. So this cannot be executed.

Line 61: if (x1<x7)

(x1<x7)	TEST CASE
T	TEST #32
F	TEST #10

Line 69: if (x1<x7)

(x1<x7)	TEST CASE
T	TEST #9
F	TEST #10

Line 77: if (x4!=2)

(X4!=2)	TEST CASE
T	TEST #33
F	TEST #8

Line 87: if (x4!=0)

(x4!=0)	TEST CASE
T	TEST #34
F	TEST #9

Line 91: if (x1!=3)

(x1!=3)	TEST CASE
T	TEST #35
F	TEST #9

Line 95: if (x2==0)

(X2==0)	TEST CASE
T	TEST #9
F	TEST #48

Line 107: if (x4!=2)

(X4!=2)	TEST CASE
T	TEST #36
F	TEST #9

Line 111: if ((x2!=0) && (x==x3)

(x2!=0)	(x==x3)	TEST CASE
T	T	TEST #9
T	F	TEST #37
F	T	TEST #38
F	F	TEST #39

Line 123: if (x4!=0)

(X4!=0)	TEST CASE
T	TEST #40
F	TEST #16

Line 127: if (x5==x)

(X5==x)	TEST CASE
T	TEST #16
F	TEST #41

Line 137: if ((x4==0) && (x2==1)

(x4==0)	(x2==1)	TEST CASE
T	T	TEST #42
T	F	TEST #43
F	T	TEST #44
F	F	TEST #16

Line 146: if (x4!=1)

(x4!=1)	TEST CASE
T	TEST #45
F	TEST #5

Line 150: if (x==x3)

(x==x3)	TEST CASE
T	TEST #5
F	TEST #46

Line 159: if (k>=num)

(k>=num)	TEST CASE
T	TEST #47
F	TEST #46

5. Test suite and Execution Results

TEST SUITE

Test#1: open 1000 123 555 login 444 login 333 login 555 pin 124 pin 123 logout login 123
Test#2: open 400 123 666 login 666 pin 123 deposit 50 balance deposit 600 logout login 777
Test#3: open 500 123 666 login 666 logout login 777 login 666 pin 123 logout login 555
Test#4: open 500 123 666 login 666 logout login 666 pin 124 pin 125 login 777 login 666 pin 124 pin 125 login 666 logout
Test#5: open 500 123 666 login 666 pin 123 logout login 666 pin 123 balance logout
Test#6: open 500 123 666 login 666 pin 124 logout login 666 pin 124 pin 123 balance balance deposit 50 deposit 100 logout
Test#7: open 500 123 666 login 666 pin 123 deposit 510 deposit 200 withdraw 300 logout
Test#8: open 1100 111 555 login 555 pin 111 withdraw 50 deposit 500 balance lock 111 balance balance unlock 111 withdraw 700 balance deposit 20 deposit 1000 withdraw 50 balance withdraw 50 lock 111 unlock 111 balance logout
Test#9: open 1100 111 555 login 555 pin 111 withdraw 200 lock 111 balance unlock 111 lock 111 unlock 111 logout
Test#10: open 1000 111 555 login 555 pin 111 deposit 50 withdraw 45 deposit 100 logout
Test#11: open 2000 222 456 login 456 pin 222 balance balance deposit 200 lock 222 unlock 222 logout
Test#12: open 2000 222 456 login 456 pin 222 lock 222 unlock 222 lock 222 unlock 222 withdraw 500 withdraw 100 logout
Test#13: open 1500 999 9876 login 9876 pin 999 deposit 50 deposit 50 balance withdraw 700 deposit 10 deposit 300 lock 999 unlock 999 deposit 100 logout
Test#14: open 1500 999 9876 login 9876 pin 999 withdraw 500 withdraw 500 deposit 600 balance logout
Test#15: open 600 333 689 login 689 pin 333 deposit 100 lock 333 unlock 333 balance lock 333 unlock 333 deposit 120 logout
Test#16: open 600 333 689 login 689 pin 333 lock 333 unlock 333 deposit 500 withdraw 300 logout
Test#17: open -1 123 123 login 123 pin 123 deposit 50 withdraw 50 balance lock 111 unlock 111 logout
Test#18: open 100 123 123 login 321 pin 123 deposit 50 withdraw 50 balance lock 111 unlock 111 logout
Test#19: open 100 123 123 login 123 pin 321 deposit 50 withdraw 50 balance lock 111 unlock 111 logout

Test#20: open 1500 123 123 login 123 pin 123 withdraw 50 deposit 50 lock 111 unlock 123 open 1 1 1 login 1 pin 1
Test#21: open 500 123 123 login 123 pin 123 withdraw 50 deposit 50 lock 111 unlock 123 open 1 1 1 login 1 pin 1
Test#22: open 1000 123 123 login 123 pin 123 lock 123 unlock 321 lock 123 open 1 1 1 withdraw 20 deposit 20 pin 123 login 234 logout
Test#23: open -1 5678 222
Test#24: open 2000 5678 222 login 222 deposit 50
Test#25: open 1100 1234 222 login 222 pin 1234 lock 1234 deposit 100
Test#26: open 400 666 888 login 888 pin 666 deposit -100
Test#27: open 1100 111 222 login 222 pin 111 deposit -100
Test#28: open 500 111 555 login 555 withdraw 50
Test#29: open 1100 1234 222 login 222 pin 1234 lock 1234 withdraw 50
Test#30: open 400 666 888 login 888 pin 666 withdraw -100
Test#31: open 500 111 222 login 222 pin 111 withdraw 1000
Test#32: open 500 111 222 login 222 pin 111 withdraw 50 logout
Test#33: open 2000 678 876 login 876 pin 876 balance
Test#34: open 2000 678 876 login 876 pin 876 lock 678
Test#35: open 500 1 1 login 1 pin 1 lock 2
Test#36: open 1000 123 456 login 456 pin 1234 unlock 123
Test#37: open 1000 123 456 login 456 pin 123 lock 123 unlock 321
Test#38: open 1000 123 456 login 456 pin 123 unlock 123
Test#39: open 1000 123 456 login 456 pin 123 lock 1 unlock 1
Test#40: open 0 111 333 login 333
Test#41: open 1100 777 333 login 33
Test#42: open 7000 111 777 lock 111 logout
Test#43: open 100 111 777 logout
Test#44: open 5000 111 777 login 777 pin 111 lock 111 logout
Test#45: open 1000 221 123 pin 221
Test#46: open 1000 212 123 login 123 pin 213
Test#47: open 1000 212 123 login 123 pin 213 pin 123
Test#48: open 1000 212 123 login 123 pin 212 lock 212 lock 212
$$

Result of Test Suite in Test suite checker:

Test#1:
  open(1000,123,555) method
  login(444) method
  login(333) method
  login(555) method
  pin(124) method
  pin(123) method
  logout() method
  login(123) method

Test#2:
  open(400,123,666) method
  login(666) method
  pin(123) method
  deposit(50) method
  balance() method
  deposit(600) method
  logout() method
  login(777) method

Test#3:
  open(500,123,666) method
  login(666) method
  logout() method
  login(777) method
  login(666) method
  pin(123) method
  logout() method
  login(555) method

Test#4:
  open(500,123,666) method
  login(666) method
  logout() method
  login(666) method
  pin(124) method
  pin(125) method
  login(777) method
  login(666) method
  pin(124) method
  pin(125) method

login(666) method
logout() method

Test#5:
open(500,123,666) method
login(666) method
pin(123) method
logout() method
login(666) method
pin(123) method
balance() method
logout() method

Test#6:
open(500,123,666) method
login(666) method
pin(124) method
logout() method
login(666) method
pin(124) method
pin(123) method
balance() method
balance() method
deposit(50) method
deposit(100) method
logout() method

Test#7:
open(500,123,666) method
login(666) method
pin(123) method
deposit(510) method
deposit(200) method
withdraw(300) method
logout() method

Test#8:
open(1100,111,555) method
login(555) method
pin(111) method
withdraw(50) method
deposit(500) method
balance() method
lock(111) method

balance() method
balance() method
unlock(111) method
withdraw(700) method
balance() method
deposit(20) method
deposit(1000) method
withdraw(50) method
balance() method
withdraw(50) method
lock(111) method
unlock(111) method
balance() method
logout() method

Test#9:
open(1100,111,555) method
login(555) method
pin(111) method
withdraw(200) method
lock(111) method
balance() method
unlock(111) method
lock(111) method
unlock(111) method
logout() method

Test#10:
open(1000,111,555) method
login(555) method
pin(111) method
deposit(50) method
withdraw(45) method
deposit(100) method
logout() method

Test#11:
open(2000,222,456) method
login(456) method
pin(222) method
balance() method
balance() method
deposit(200) method
lock(222) method

unlock(222) method
logout() method

Test#12:
open(2000,222,456) method
login(456) method
pin(222) method
lock(222) method
unlock(222) method
lock(222) method
unlock(222) method
withdraw(500) method
withdraw(100) method
logout() method

Test#13:
open(1500,999,9876) method
login(9876) method
pin(999) method
deposit(50) method
deposit(50) method
balance() method
withdraw(700) method
deposit(10) method
deposit(300) method
lock(999) method
unlock(999) method
deposit(100) method
logout() method

Test#14:
open(1500,999,9876) method
login(9876) method
pin(999) method
withdraw(500) method
withdraw(500) method
deposit(600) method
balance() method
logout() method

Test#15:
open(600,333,689) method
login(689) method
pin(333) method

deposit(100) method
lock(333) method
unlock(333) method
balance() method
lock(333) method
unlock(333) method
deposit(120) method
logout() method

Test#16:
open(600,333,689) method
login(689) method
pin(333) method
lock(333) method
unlock(333) method
deposit(500) method
withdraw(300) method
logout() method

Test#17:
open(-1,123,123) method
login(123) method
pin(123) method
deposit(50) method
withdraw(50) method
balance() method
lock(111) method
unlock(111) method
logout() method

Test#18:
open(100,123,123) method
login(321) method
pin(123) method
deposit(50) method
withdraw(50) method
balance() method
lock(111) method
unlock(111) method
logout() method

Test#19:
open(100,123,123) method
login(123) method

pin(321) method
deposit(50) method
withdraw(50) method
balance() method
lock(111) method
unlock(111) method
logout() method

Test#20:
open(1500,123,123) method
login(123) method
pin(123) method
withdraw(50) method
deposit(50) method
lock(111) method
unlock(123) method
open(1,1,1) method
login(1) method
pin(1) method

Test#21:
open(500,123,123) method
login(123) method
pin(123) method
withdraw(50) method
deposit(50) method
lock(111) method
unlock(123) method
open(1,1,1) method
login(1) method
pin(1) method

Test#22:
open(1000,123,123) method
login(123) method
pin(123) method
lock(123) method
unlock(321) method
lock(123) method
open(1,1,1) method
withdraw(20) method
deposit(20) method
pin(123) method
login(234) method

  logout() method

Test#23:
  open(-1,5678,222) method

Test#24:
  open(2000,5678,222) method
  login(222) method
  deposit(50) method

Test#25:
  open(1100,1234,222) method
  login(222) method
  pin(1234) method
  lock(1234) method
  deposit(100) method

Test#26:
  open(400,666,888) method
  login(888) method
  pin(666) method
  deposit(-100) method

Test#27:
  open(1100,111,222) method
  login(222) method
  pin(111) method
  deposit(-100) method

Test#28:
  open(500,111,555) method
  login(555) method
  withdraw(50) method

Test#29:
  open(1100,1234,222) method
  login(222) method
  pin(1234) method
  lock(1234) method
  withdraw(50) method

Test#30:
  open(400,666,888) method
  login(888) method

pin(666) method
withdraw(-100) method

Test#31:
open(500,111,222) method
login(222) method
pin(111) method
withdraw(1000) method

Test#32:
open(500,111,222) method
login(222) method
pin(111) method
withdraw(50) method
logout() method

Test#33:
open(2000,678,876) method
login(876) method
pin(876) method
balance() method

Test#34:
open(2000,678,876) method
login(876) method
pin(876) method
lock(678) method

Test#35:
open(500,1,1) method
login(1) method
pin(1) method
lock(2) method

Test#36:
open(1000,123,456) method
login(456) method
pin(1234) method
unlock(123) method

Test#37:
open(1000,123,456) method
login(456) method
pin(123) method

lock(123) method
unlock(321) method

Test#38:
open(1000,123,456) method
login(456) method
pin(123) method
unlock(123) method

Test#39:
open(1000,123,456) method
login(456) method
pin(123) method
lock(1) method
unlock(1) method

Test#40:
open(0,111,333) method
login(333) method

Test#41:
open(1100,777,333) method
login(33) method

Test#42:
open(7000,111,777) method
lock(111) method
logout() method

Test#43:
open(100,111,777) method
logout() method

Test#44:
open(5000,111,777) method
login(777) method
pin(111) method
lock(111) method
logout() method

Test#45:
open(1000,221,123) method
pin(221) method

```
Test#46:
   open(1000,212,123) method
   login(123) method
   pin(213) method

Test#47:
   open(1000,212,123) method
   login(123) method
   pin(213) method
   pin(123) method

Test#48:
   open(1000,212,123) method
   login(123) method
   pin(212) method
   lock(212) method
   lock(212) method

The test suite has been checked.
No errors have been detected.

    Press any key to continue
```

Result of Execution:

Test #1

Expected Outcome: The final state should reach Idle from Check Pin, login 123 should return -1, attempts left should be 1 after executing T2, T3.
Actual Outcome: The final state is Idle and the prior state is Check Pin and it returns -1 on executing login 123, the number of attempts left is 1 and can be known by show_attempts().
Result: Pass

Test #2

Expected Outcome: The final state should reach Idle state from Ready state, login 777 should return -1
Actual Outcome: the final state is Idle and the prior state is Ready state and it returns -1 on executing login 777
Result: Pass

Test #3

Expected Outcome: The final state should reach Idle state from Overdrawn state, login 555 should return -1

Actual Outcome: the final state is Idle and the prior state is Overdrawn state and it returns -1 on executing login 555

Result: Pass

Test #4

Expected Outcome: The final state should reach Idle state, on executing pin(125) and pin(125) the number of trials left should be 0

Actual Outcome: the final state is Idle as expected, the number of trials after executing the above methods the number of trials is 0

Result: Pass

Test #5

Expected Outcome: Executing balance from overdrawn must display the balance as 500

Actual Outcome: The account balance is displayed as 500 when the above series of operations are executed.

Result: Pass

Test #6

Expected Outcome: By executing the balance method 500 should be displayed and after two deposits are executed the balance should be 630.

Actual Outcome: While executing a series of transactions as listed in above test when balance is executed the balance displayed is 500 after two deposits we may use test oriented methods of show_balance() to check the balance and the balance displayed as 630.

Result: Pass

Test #7

Expected Outcome: After executing all the given transactions the balance should be 890

Actual Outcome: After executing the series of transactions and is show_balance() is executed the result is 890.

Result: Pass

Test #8

Expected Outcome: executing balance for the first time should display 1550.

Actual Outcome: As per the execution the execution of balance for the first time displayed 1550.

Result: Pass

Test #9

Expected Outcome: The Balance should be 890

Actual Outcome: After executing all the above transactions and executing show_balance returned 890

Result: Pass

Test #10

Expected Outcome: The final state should reach Idle state from ready state. The balance should be 1105

Actual Outcome: the final state is Idle and the prior state is Ready state. The balance is 1105 as expected

Result: Pass

Test #11

Expected Outcome: The final state should reach Idle state from ready state, login 777 should return -1

Actual Outcome: the final state is Idle and the prior state is Ready state and it returns -1 on executing login 777

Result: Pass

Test #12

Expected Outcome: After executing lock the state should be in locked and at the end of the inputs the balance should be 1500

Actual Outcome: The final balance is 1500

Result: Pass

Test #13

Expected Outcome: on executing the balance the output should be 1600, lock and unlock should be executed successfully.

Actual Outcome: lock and unlock are executed successfully.

Result: Pass

Test #14

Expected Outcome: on executing balance command 490 should be displayed.

Actual Outcome: The balance is 490 as expected.

Result: Pass

Test #15

Expected Outcome: on executing balance command 690 should be displayed.

Actual Outcome: The balance is 690 as expected.

Result: Pass

Test #16

Expected Outcome: When opening balance is 600 and 500 is deposited the Total balance should be 1090 because it is moved from overdrawn to ready where $10 is penalty

Actual Outcome: The balance is 490 as expected.

Result: Pass

Test #17

Expected Outcome: These are the default transitions for the Idle state

Actual Outcome: Access Denied! (return =-1)is the message displayed for all the methods executed except for logout.

Result: Fail

Explanation: As per the EFSM model there is no logout transaction from Start. But when we execute using account class it is returning Successful (return : 0)

Test #18

Expected Outcome: These are the default transitions for the Start state

Actual Outcome: Access Denied! (return =-1)is the message displayed for all the methods executed once reaching the start state.

Result: Pass

Test #19

Expected Outcome: These are the default transitions for the check pin state

Actual Outcome: Access Denied! (return =-1)is the message displayed for all the methods executed once reaching the check pin state.

Result: Pass

Test #20

Expected Outcome: These are the default transitions for the Ready state. We stay in ready state once if we reach there.

Actual Outcome: Access Denied! (return =-1)is the message displayed for all the methods executed once reaching the ready state.

Result: Pass

Test #21

Expected Outcome: These are the default transitions for the overdrawn state. We stay in overdrawn state once if we reach there.

Actual Outcome: Access Denied! (return =-1)is the message displayed for all the methods executed once reaching the overdrawn state.

Result: Pass

Test #22

Expected Outcome: These are the default transitions for the lock state. We stay in locked state once if we reach there.

Actual Outcome: Access Denied! (return =-1)is the message displayed for all the methods executed once reaching the locked state.

Result: Pass

Test #23

Expected Outcome: On executing this test should not change state

Actual Outcome: Access Denied!(-1) is returned when the test is executed.

Result: Pass

Test #24

Expected Outcome: Executing deposit should fail

Actual Outcome: Access Denied (-1) is returned when deposit is executed because there is no valid transaction from check pin state which will execute deposit.

Result: Pass

Test #25

Expected Outcome: lock command should not be executed.

Actual Outcome: Executing lock command will return Access Denied!(-1)

Result: Pass

Test #26

Expected Outcome: Deposit method should not be executed as deposit amount is less than 0

Actual Outcome: Executing deposit command return Access Denied!(-1)

Result: Pass

Test #27

Expected Outcome: Deposit method should not be executed as deposit amount is less than 0

Actual Outcome: Executing deposit command return Access Denied!(-1)

Result: Pass

Test #28

Expected Outcome: withdraw method should not be executed as the transaction is not valid from check pin state

Actual Outcome: Executing withdraw command return Access Denied!(-1)

Result: Pass

Test #29

Expected Outcome: lock command moves control to locked state withdraw should not be executed.

Actual Outcome: Executing withdraw command will return Access Denied!(-1)

Result: Pass

Test #30

Expected Outcome: withdraw method should not be executed as withdraw amount is less than 0

Actual Outcome: Executing withdraw command return Access Denied!(-1)

Result: Pass

Test #31

Expected Outcome: withdraw method should not be executed as there is no withdraw method in overdrawn state

Actual Outcome: Executing withdraw command will return Access Denied!(-1)

Result: Pass

Test #32

Expected Outcome: withdraw method should not be executed as there is no withdraw method in overdrawn state

Actual Outcome: Executing withdraw command will return Access Denied!(-1)

Result: Pass

Test #33

Expected Outcome: Executing pin with wrong pin should reduce the number of attempts and also balance cannot be executed from check pin

Actual Outcome: After Executing wrong pin and checking number of attempts left 1 is displayed, executing balance results in Access Denied!

Result: Pass

Test #34

Expected Outcome: Executing pin with wrong pin should reduce the number of attempts and also lock cannot be executed from check pin

Actual Outcome: After Executing wrong pin and checking number of attempts left 1 is displayed, executing lock results in Access Denied!

Result: Pass

Test #35

Expected Outcome: Using a wrong pin to lock doesn’t lock the account

Actual Outcome: Executing lock method using a wrong pin command will return Access Denied!(-1)

Result: Pass

Test #36

Expected Outcome: Executing pin with wrong pin should reduce the number of attempts and also unlock cannot be executed from check pin

Actual Outcome: After Executing wrong pin and checking number of attempts left 1 is displayed, executing unlock results in Access Denied!

Result: Pass

Test #37

Expected Outcome: Executing unlock with wrong pin should remain in same locked state

Actual Outcome: After Executing unlock with wrong pin Access Denied! Is returned.

Result: Pass

Test #38

Expected Outcome: Unlock cannot be executed without entering locked state.

Actual Outcome: After entering into check pin state and executing unlock Access Denied is returned.

Result: Pass

Test #39

Expected Outcome: Unlock cannot be executed without entering locked state.

Actual Outcome: After entering into check pin state and executing unlock Access Denied is returned.

Result: Pass

Test #40

Expected Outcome: Account cannot be created with 0 balance

Actual Outcome Access Denied! Is returned when trying to open an account with balance <1

Result: Pass

Test #41

Expected Outcome: Account cannot be created with 0 balance

Actual Outcome Access Denied! Is returned when trying to open an account with balance <1

Result: Pass

Test #42

Expected Outcome: Account cannot be locked from Idle state and obviously logout cannot happen.

Actual Outcome Access Denied! Is returned when trying to open an account try to lock immediately.

Result: Pass

Test #43

Expected Outcome: logout cannot be executed from Idle state.

Actual Outcome Access Denied! Is returned when trying to open an account and logout immediately

Result: Pass

Test #44

Expected Outcome: Logout cannot be executed from lock state

Actual Outcome Access Denied! Is returned when trying to logout an account after executing lock.

Result: Pass

Test #45

Expected Outcome: pin cannot be executed from Idle state

Actual Outcome Access Denied! Is returned when trying to enter pin soon after opening.

Result: Pass

Test #46

Expected Outcome: Entering wrong pin should reduce the number of Attempts left

Actual Outcome Access Denied! Is returned when trying enter wrong pin and executing show_attempts will display 1

Result: Pass

Test #47

Expected Outcome: If wrong pin is entered twice the value of k should be 2

Actual Outcome Access Denied! Is returned when trying enter wrong pin and if wrong pin is enetered twice and executing show_attempts is displaying 0, which is correct(since num =2).

Result: Pass

Test #48

Expected Outcome: locking for the second time is not possible.

Actual Outcome Access Denied! Is returned when trying to lock for second time.

Result: Pass

6. Conclusion

Model Based Testing and Default Transition Testing are used to cover all valid and invalid transactions that can be traversed in a given EFSM model. In additional to these tests the Multiple Condition Testing all helps a tester to find some more interesting aspects which once passed makes a project more reliable and the outcome is more error free.

We can assure that the project is error free for most of the extent except the transition of logout state from Start state

7. Implementation:

```
//package sta;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;


/**
 *
 * @author Nickhil
 */

public class Sta
{


   public static void main(String[] args)  throws IOException
   {   String ch,bal1,pin1;
       int bal,no=0,pin=0,amt,result;
       account acc=new account();
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("\n\n\n  Nickhil Revu                  A20299434  \n");
                System.out.println("  ***** CS 589:Software Testing Analysis *****  \n");
                System.out.println("\n\n\nPress Enter to continue.....");
                ch=br.readLine();

                System.out.println("");
       System.out.println("----------------Menu---------------");
       System.out.println("1.Open");
       System.out.println("2.Login");
       System.out.println("3.Pin");
       System.out.println("4.balance");
       System.out.println("5.Lock");
       System.out.println("6.Unlock");
       System.out.println("7.Withdraw");
       System.out.println("8.Deposit");
       System.out.println("9.Logout");
       System.out.println("10.exit");
```

```
                System.out.println("\n -----Test Oriented Methods-----");
System.out.println("a.Show Balance");
System.out.println("b.Current State");
                System.out.println("c.Show Pin");
                System.out.println("d.Show account Number");
                System.out.println("e.Show Number of Attempts left");
                System.out.println("f.Is locked");
                System.out.println(“g.Current State);

System.out.println("\nselect choice");

ch=br.readLine();

while(ch.compareTo("10")!=0)
{
                try
                            {
   switch(ch)
   {

      case "1":
         System.out.println("Enter Opening Balance");
         bal1=br.readLine();
         bal=Integer.parseInt(bal1);
         System.out.println("Enter Pin");
         pin1=br.readLine();
         pin=Integer.parseInt(pin1);
         System.out.println("Enter Acoount Number");
         no=Integer.parseInt(br.readLine());
         result=acc.open(bal, pin, no);
                                        if(result==0)
                                          System.out.println("\nAccount Opened Successfull");
                                        else
                                                 System.out.println("\nAccess Denied!");
         break;

      case"2":
         System.out.println("Enter Account Number");
         no=Integer.parseInt(br.readLine());
         result=acc.login(no);
                                        if(result==0)
                                          System.out.println("\nLogin Successfull");
                                        else
                                           System.out.println("\nAccess Denied!");
```

```
                                              break;

                case"3":
                    System.out.println("Enter pin");
                    pin=Integer.parseInt(br.readLine());
                    result=acc.pin(pin);
                                              if(result==0)
                                                System.out.println("\nSuccessfull");
                                              else
                                                      System.out.println("\nAccess Denied!");
                    break;

                case "4":
                    result=acc.balance();
                                              if(result!=-1)
                                                 System.out.println("Current Balanace :"+result);
                                              else
                                                      System.out.println("\nAccess Denied!");
                    break;

                case "5":
                     System.out.println("Enter your Pin");
                     pin =Integer.parseInt(br.readLine());
                     result=acc.lock(pin);
                                               if(result==0)
                                                System.out.println("\nAccount Successfull Locked");
                                              else
                                                      System.out.println("\nAccess Denied!");
                     break;

                case "6":
                     System.out.println("Enter your Pin");
                     pin =Integer.parseInt(br.readLine());
                     result=acc.unlock(pin);
                                               if(result==0)
                                                System.out.println("\nAccount Successfull Unlocked");
                                              else
                                                      System.out.println("\nAccess Denied!");
                     break;

                case "7":
                     System.out.println("Enter Amount for Withdraw");
                     amt=Integer.parseInt(br.readLine());
                     result=acc.withdraw(amt);
```

```
                                          if(result==0)
                                            System.out.println("\nAmount Withdrawn Successfull");
                                         else
                                                  System.out.println("\nAccess Denied!");
                    break;

                case "8":
                    System.out.println("Enter Amount for Deposit");
                    amt=Integer.parseInt(br.readLine());
                    result=acc.deposit(amt);
                                          if(result==0)
                                            System.out.println("\nAmount Deposited Successfull");
                                         else
                                                  System.out.println("\nAccess Denied!");
                    break;

                case "9":
                    result=acc.logout();
                                          if(result==0)
                                            System.out.println("\nSuccessfully Logged out....See you
Again.");
                                         else
                                                  System.out.println("\nAccess Denied!");

                    break;

                case "10":
                    break;

                case "a":
                    System.out.println("Balance : "+acc.show_balance());
                    break;

                case "b":
                    System.out.println("Current State : "+acc.show_balance());
                    break;

                                 case "c":
                                          System.out.println("Pin : "+acc.show_pin());
                                          break;

                                 case "d":
                                          System.out.println("Account          Number          :
"+acc.show_account());
```

```
                                        break;

                                case "e":
                                        System.out.println("Attempts            Left            :
"+acc.show_attempts());
                                        break;

                                case "f":
                                        result=acc.is_locked();
                                        if (result ==0)
                                                System.out.println("Account is in not locked state");
                                        else
                                                System .out.println("Account is  locked");
                                        break;
                                case "g":
                                        System.out.println("Current State :”);
                                       Acc.show_state();
                                        break;



                        }

        }catch(IOException | NumberFormatException | ArrayIndexOutOfBoundsException e)
                                       {
                                       System.out.println("\nERROR: Invalid Entry\n\n"+e);
                                       }

            System.out.println("\n\nPress Enter to continue...");
            ch=br.readLine();
            System.out.println("----------------Menu---------------");
            System.out.println("1.Open");
            System.out.println("2.Login");
            System.out.println("3.Pin");
            System.out.println("4.balance");
            System.out.println("5.Lock");
            System.out.println("6.Unlock");
            System.out.println("7.Withdraw");
            System.out.println("8.Deposit");
            System.out.println("9.Logout");
            System.out.println("10.exit");
            System.out.println("\n -----Test Oriented Methods-----");
            System.out.println("a.Show Balance");
            System.out.println("b.Current State");
```

```
                    System.out.println("c.Show Pin");
                    System.out.println("d.Show account Number");
                    System.out.println("e.Show Number of Attempts left");
                    System.out.println("f.Is locked");
                    System.out.println(“g.Current State);
        System.out.println("\nselect choice");
        ch=br.readLine();

        }

    }

}

class account
{
public account()
{
            x2 = 0;
            x4 = -1;
            x6 = 10;
            x7 = 1000;
            k = 0;
            num = 2;
}
public int open(int x, int y, int z)
{
            if ((x > 0) && (x4 == -1))
            {
                        x1 = x;
                        x3 = y;
                        x5 = z;
                        x4 = 0;
                        return 0;
            }
            return -1;
}
public int deposit(int d)
{
            if (x4 != 2)
            {
                        return -1;
            }
            if (x2 != 0)
            {
                        return -1;
            }
            if ((x1 < x7) && (d>0))
            {
                        x1 = x1 + d - x6;
                        return 0;
            }
            else
            {
                        if (d > 0)
                        {
                        x1 = x1 + d;
                        return 0;
                        }
            }
 return -1;
}
public int withdraw(int w)
{
if (x4 != 2)
{
            return -1;
}
if (x2 != 0)
{
            return -1;
}
if ((x1 > w) && (w > 0))
{
            if (x1 < x7)
            {
                        return -1;
            }
            else
            {
                        x1 = x1 - w;
            };
            if (x1 < x7)
            {
                        x1 = x1 - x6;
            }
  return 0;
}
```

```
return -1;
}
public int balance()
{
        if (x4 != 2)
        {
                return -1;
        }
        return x1;
}
public int lock(int x)
{
        if (x4 != 2)
        {
                return -1;
        }
        if (x != x3)
        {
                return -1;
        }
        if (x2 == 0)
        {
                x2 = 1;
                return 0;
        }
        else
        {
                return -1;
        }
}
public int unlock(int x)
{
        if (x4 != 2)
        {
                return -1;
        }
        if ((x2 != 0) && (x == x3))
        {
                x2 = 0;
        return 0;
        }
        else
        {
                return -1;
        }
    }
    public int login(int x)
    {
        if (x4 != 0)
        {
            return -1;
        }
        if (x5 == x)
        {
            x4 = 1;
            k = 0;
            return 0;
        }
        return -1;
    }
    public int logout()
    {
        if ((x4 == 0) || (x2 == 1))
        {
            return -1;
        }
        x4 = 0;
        return 0;
    }
    public int pin(int x)
    {
        if (x4 != 1)
        {
            return -1;
        }
        if (x == x3)
        {
            x4 = 2;
            return 0;
        }
        else
        {
            k++;
        }
        if (k >= num)
        {
            x4 = 0;
        }
```

```
            return -1;
    }

    /*Testing Oriented Methods*/
    public int show_balance()
    {
            return x1;
    }
    public int show_pin()
    {
            return x3;
    }
    public int show_account()
    {
            return x5;
    }
    public int show_attempts()
    {
            return (2-k);
    }
    public int is_locked()
    {
            if (x2==0)
              return 0;
            else
               return -1;
    }
    public void show_state()
    {
       if((x2==1)&&(x4==2))
       {
          System.out.println("LOCKED STATE");
       }
       if((x4==0)&&(x2==0))
       {
          System.out.println("IDLE STATE");

       }
       if((x4==1)&&(x2==0)&&(k<num))
       {
          System.out.println("CHECKPIN STATE");
       }
       if((x4==2)&&(x1>999)&&(x2==0))
       {
          System.out.println("READY STATE");
       }
       if((x4==2)&&(x1<=999)&&(x2==0))
       {
          System.out.println("OVER DRAWN");
       }
       if(x4==-1)
       {
          System.out.println("START STATE");
       }

    }


    /*End of Testing Oriented Methods*/
            private int x1; //balance
            private int x2;
            private int x3; //pin #
            private int x4;
            private int x5; // account #
            private int x6; // penalty
            private int x7; // minimum balance
            private int k;
            private int num; //maximum # of attempts with incorrect pin
    }
```

Note: The line numbers mentioned in the Project Report are the line numbers of the given account class