CS 589 Final Project Report

BY : Bavan Prashant (A20272337)

[Introduction](#)
[Model Based Testing](#)
[Testing Default Transitions](#)
[Testing Methods of Account Class (path Testing)](#)
[Test Suite](#)
[Test Suite Results](#)
[Conclutions](#)
[Modified Account Class](#)
[Deliverable](#)

Introduction

The goal of the project was to conduct Obejct Oriented State based testing on the given Account.java Class. For this purpose, a Brief Descrption, A detailed EFSM Model, and the source code were provided. The end result was expected to be as a executive Test driver, A Test suite describing the Test cases, that satisfy, 2-pair transition testing and path testing along with certain default test cases.

Model Based Testing

The Model based Testing process involved identifying the States and the Transitions involved in the system. This was done with the help of the EFSM model provided.The following are the States and the input and output transitions along with the test case numbers of the test suit that satisfy the transitions.For the Test cases and the results please refer the Test Suit and test results attached

Idle State 12 possible transitions
Input States T1, T5, T6, T7, T9, T10

Output States T2, T7

* - T2 Test 1

*- T7 Test 2

Check Pin State 10 possible transitions

Input States T2, T3

Output States T3,T4,T5,T6,T8

T2- T3 , T4 , T5	Test 1
T2- T6	**Not possible T2 (attempts==0) -> T6 (attempts==2)**
T2 -T8	Test 3
T3- T3	Test 1
T3 - T6	Test 1
T3 - T4 , T5, T8	Test 3

Ready State 36 possible transitions

Input States T4, T11, T12, T13, T17, T15

Output States T11,T12,T13,T16,T14,T10

T4 -T10	Test 1
T4 -*	Test 4
T11 -*	Test 5
T12 -*	Test 6
T13 -*	Test 7
T17 - T10	Test 4
T17 - T11	Test 5
T17 - T12	Test 6
T17 - T13	Test 7
T17 -*	Test 8
T15- *	Test 9

Locked State 9 possible transitions

Input States T16, T18, T20

Output States T17, T18, T19

T16- T17	Test 5
T16- T18	Test 4
T16 -T19	**Not possible Ready locked to overdrawn is not allowed by EFSM**
T18-T17	Test 4
T18-T18	Test 10
T18-T19	Test 10
T20 - T17	**Not possible Overdrawn locked to Ready is not allowed by EFSM**
T20 - T18	Test 10
T20 - T19	Test 10

Overdrawn State 25 possible transitions

Input States T14,T19, T22, T21, T8

Output States T9, T15,T20,T22, t21

T8-*	Test 11
T14 -T9	Test 11
T14 - T15	Test 7
T14 - T20	Test 10
T14 - T21	Test 12
T14 - T22	Test 12
T19 - *	Test 13
T22 - *	Test 14
T21 - *	Test 15

Testing Default Transitions

All transitions not mentioned in the EFSM are considered to be default transitions.
As an understanding of the problem statement, we can say that there are 9 functions and 5 states, making 9*5 = 45 possible transactions. But only 22 are mentioned. Along with the remaining 23 test cases, certain situations where w<0 for withdraw and invalid inputs make the list of default transitions a huge number of test cases that can always be growing in nature.
but to satisfy certain possibilities, for each State, the functions that are not explicitly mentioned are tested.
These test cases are from Test cases number from 16-21 in the Test Suite.
By no means are these complete, but they are found to be the most basic of the default transitions that are covered in this test suit.

From the **Idle State** Test 16 covers

~~account~~
~~login~~
1) logout
2) pin
3) deposit
4) withdraw
5) balance
6) lock
7) unlock

From the **Check Pin State** Test 17 covers

~~account~~
1) login
2) logout
~~pin~~
3) deposit
4) withdraw

5) balance
6) lock
7) unlock

From the **Ready State** Test 18 covers

~~account~~
1) login
~~logout~~
2) pin
~~deposit~~
~~withdraw~~
~~balance~~
~~lock~~
3) unlock

From the **Locked State** Test 19 covers

~~account~~
1)login
2)logout
3)pin
4)deposit
5)withdraw
~~balance~~
6)lock
~~unlock~~

From the **Overdrawn State** Test 20 covers

~~account~~
1) login
~~logout~~
2) pin
~~deposit~~
3) withdraw
~~balance~~
~~lock~~
4) unlock

Testing Methods of Account Class (path Testing)

Path testing involved identifying all the possible paths in each function.For this every function in the Account class had to be under scrutiny line by line. From the process, a list of all the paths were listed as found bellow. Then in combination of the test suit, the corresponding test cases were filled in and new required test cases were authored to handle paths that were not already

taken care of.

	Account		**TEST NUMBER**
1	bal< 0		Test 21
2	! (bal < 0)		Test 22

	Deposit		**TEST NUMBER**
1	lg != 2	NOT	Test 16
2	locked != 0	NOT	Test 19
3	bal < min_bal	d > 0	Test 15
4	bal < min_bal	! (d > 0)	Test 23
5	!(bal < min_bal)	d < 0	Test 4
6	!(bal < min_bal)	! (d < 0)	Test 24

	Withdraw			**TEST NUMBER**
1	lg != 2	NOT		Test 16
2	locked ! = 0	NOT		Test 25
3	(bal >w) && (w>0)	(bal < min_bal)		Test 20
4	(bal >w) && (w>0)	!(bal < min_bal)	(bal < min_bal)	Test 1
5	(bal >w) && (w>0)	!(bal < min_bal)	!(bal < min_bal)	Test 4
6	!(bal > w)			Test 20
7	(bal > w) && (w<0)			Test 27

	Balance		**TEST NUMBER**
1	lg != 2	NOT	Test 16

	Lock		**TEST NUMBER**
1	lg != 2	NOT	Test 16
2	x != pin	NOT	Test 26
3	(locked == 0)		Test 4
4	!(locked == 0)	**(lg!=2)&&(locked!=0) Contradiction NOT POSSIBLE**	

	UNLOCK		**TEST NUMBER**
1	lg != 2	NOT	Test 16
2	(locked != 0)	x==pin_num	Test 4
3	(locked != 0)	!(x==pin_num)	Test 26
4	!(locked == 0)		Test 19

	Login		**TEST NUMBER**
1	lg != 0	NOT	Test 16
2	account_num ==x		Test 1
3	!(account_num==x)		Test 1

	LogOut		**TEST NUMBER**

1	lg==0	NOT	Test 1
2	(locked ==1)		Test 19
3	!(locked ==1)		Test 1

	PIN		TEST NUMBER
1	lg!=1	NOT	Test 16
2	(x==pin_num)	NOT	Test 16
3	!(x==pin_num)	k>=num	Test 1
4	!(x==pin_num)	!(k>=num)	Test 3

Test Suite

The Test Suit was created in accordance to pass a given test Suite checker.
The test case numbers used in the test suite was used in all other places in the project and the report
The following sections was the authored Test Suite and the file is attached for reference.

Test#1: account 1000 222 111 logout login 111 logout login 111 pin 111 pin 123 pin 234 login 111 pin 222 logout login 111 pin 222 withdraw 990 logout login 111 logout login 123 login 111 logout
Test#2: account 1000 222 111 login 123 login 111 logout login 123 login 111 pin 111 pin 123 pin 234 login 123 login 111 pin 222 logout login 123 login 111 pin 222 withdraw 990 logout login 123 login 111 logout login 123 login 123 login 111 logout
Test#3: account 1000 222 111 login 111 pin 222 withdraw 990 logout login 111 pin 222 logout login 111 pin 223 logout login 111 pin 223 pin 222 deposit 990 logout login 111 pin 223 pin 222 logout
Test#4: account 1000 222 111 login 111 pin 222 withdraw 10 logout login 111 pin 222 deposit 10 logout login 111 pin 222 balance logout login 111 pin 222 lock 222 balance unlock 222 logout login 111 pin 222 withdraw 990 logout
Test#5: account 1000 222 111 login 111 pin 222 withdraw 1 withdraw 1 deposit 2 withdraw 1 balance withdraw 1 lock 222 unlock 222 withdraw 1 withdraw 990 deposit 990 withdraw 1 logout
Test#6: account 1000 222 111 login 111 pin 222 deposit 1 withdraw 1 deposit 1 deposit 1 balance deposit 1 lock 222 unlock 222 deposit 1 withdraw 990 deposit 990 deposit 100 logout
Test#7: account 1000 222 111 login 111 pin 222 balance withdraw 100 balance deposit 100 balance balance lock 222 unlock 222 balance withdraw 990 deposit 990 balance logout
Test#8: account 1000 222 111 login 111 pin 222 lock 222 unlock 222 lock 222 unlock 222 withdraw 990 deposit 990 lock 222 unlock 222 logout

Test#9: account 1000 222 111 login 111 pin 222 withdraw 990 deposit 990 withdraw 1 withdraw 990 deposit 990 deposit 1 withdraw 990 deposit 990 balance withdraw 990 deposit 990 lock 222 unlock 222 withdraw 990 deposit 990 logout
Test#10: account 1000 222 111 login 111 pin 222 lock 222 balance balance unlock 222 withdraw 990 lock 222 balance unlock 222 lock 222 unlock 222
Test#11: account 10 222 111 login 111 pin 222 logout login 111 pin 222 deposit 1000 withdraw 1000 logout login 111 pin 222 lock 222 unlock 222 logout login 111 pin 222 balance logout login 111 pin 222 deposit 20 logout
Test#12: account 1000 222 111 login 111 pin 222 withdraw 990 deposit 10 deposit 990 withdraw 990 balance deposit 10 balance logout
Test#13: account 1000 222 111 login 111 pin 222 withdraw 990 lock 222 unlock 222 logout login 111 pin 222 lock 222 unlock 222 deposit 990 withdraw 990 lock 222 unlock 222 balance lock 222 unlock 222 deposit 20 logout
Test#14: account 10 222 111 login 111 pin 222 balance logout login 111 pin 222 balance deposit 990 withdraw 990 balance lock 222 unlock 222 balance balance deposit 20 logout
Test#15: account 10 222 111 login 111 pin 222 deposit 1 logout login 111 pin 222 deposit 1 deposit 990 withdraw 990 deposit 1 lock 222 unlock 222 deposit 1 balance deposit 1 deposit 1 logout
Test#16: account 1000 222 111 pin 222 withdraw 1 deposit 2 balance logout lock 222 unlock 222 login 111 login 111 logout pin 222 login 111 pin 222
Test#17: account 1000 222 111 login 111 balance withdraw 1 deposit 2 balance lock 222 lock 223 unlock 222 unlock 223 logout login 111 pin 223 logout
Test#18: account 111 222 111 login 111 pin 222 unlock 222 pin 222 login 111 logout
Test#19: account 1000 222 111 login 111 pin 222 unlock 222 lock 222 lock 222 balance withdraw 100 deposit 1000 pin 222 login 111 logout unlock 222 logout
Test#20: account 10 222 111 login 111 pin 222 withdraw 100 withdraw 1 deposit 12 withdraw 1 unlock 222 pin 222 logout
Test#21: account -20 222 111
Test#22: account 20 222 111
Test#23: account 10 222 111 login 111 pin 222 deposit -200
Test#24: account 1000 222 111 login 111 pin 222 deposit -220
Test#25: account 1000 222 111 login 111 pin 222 lock 222 withdraw 2 unlock 222 logout
Test#26: account 1000 222 111 login 111 pin 222 lock 223 lock 222 unlock 223 unlock 222
Test#27: account 1000 222 111 login 111 pin 222 withdraw -2

Test Suite Results

The test cases mentioned in the test Suite were tested using the test Driver class written and attached as a deliverable. The results were noted in separate files.It would be easier to reference them in the respective files.
A sample of the Result for Test case #1 is as follows.

TEST 1.txt file.

```
run:
/>/> 0 to quit testing /</<
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :1
Enter balance :1000
Enter PIN :222
Enter account id :111
Successfully initialized account.

The current  values in the account class are  bal :1000 locked :0 LG :0 account_num :111
penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :9

The current  values in the account class are  bal :1000 locked :0 LG :0 account_num :111
penalty :1 min_bal :100 K :0 Num :3
The Logout function executed with a result of -1

The current  values in the account class are  bal :1000 locked :0 LG :0 account_num :111
penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
```

7.lock needs int
8.unlock needs int
9.logout
Function Choice :2
Enter the login id :111

The current values in the account class are bal :1000 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Login function executed with a result of 0

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :9

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Logout function executed with a result of 0

The current values in the account class are bal :1000 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :2
Enter the login id :111

The current values in the account class are bal :1000 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Login function executed with a result of 0

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :3
Enter a pin :111

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Pin function executed with a result of -1

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :1 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :3
Enter a pin :123

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :1 Num :3
The Pin function executed with a result of -1

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :2 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int

6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :3
Enter a pin :234

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :2 Num :3
The Pin function executed with a result of -1

The current values in the account class are bal :1000 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :3 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :2
Enter the login id :111

The current values in the account class are bal :1000 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :3 Num :3
The Login function executed with a result of 0

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :3
Enter a pin :222

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Pin function executed with a result of 0

The current values in the account class are bal :1000 locked :0 LG :2 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :9

The current values in the account class are bal :1000 locked :0 LG :2 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Logout function executed with a result of 0

The current values in the account class are bal :1000 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :2
Enter the login id :111

The current values in the account class are bal :1000 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Login function executed with a result of 0

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int

4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :3
Enter a pin :222

The current values in the account class are bal :1000 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Pin function executed with a result of 0

The current values in the account class are bal :1000 locked :0 LG :2 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :6
Enter a withdraw ammount :990

The current values in the account class are bal :1000 locked :0 LG :2 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Withdraw function executed with a result of 0

The current values in the account class are bal :9 locked :0 LG :2 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :9

The current values in the account class are bal :9 locked :0 LG :2 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Logout function executed with a result of 0

The current values in the account class are bal :9 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :2
Enter the login id :111

The current values in the account class are bal :9 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Login function executed with a result of 0

The current values in the account class are bal :9 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :9

The current values in the account class are bal :9 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Logout function executed with a result of 0

The current values in the account class are bal :9 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int

4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :2
Enter the login id :123

The current values in the account class are bal :9 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Login function executed with a result of -1

The current values in the account class are bal :9 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :2
Enter the login id :111

The current values in the account class are bal :9 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Login function executed with a result of 0

The current values in the account class are bal :9 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :9

The current values in the account class are bal :9 locked :0 LG :1 account_num :111 penalty :1 min_bal :100 K :0 Num :3
The Logout function executed with a result of 0

The current values in the account class are bal :9 locked :0 LG :0 account_num :111 penalty :1 min_bal :100 K :0 Num :3
1.account needs int int int
2.login needs int
3.pin needs int
4.balance
5.deposit needs int
6.withdraw needs int
7.lock needs int
8.unlock needs int
9.logout
Function Choice :0
Please enter a valid Function choise
Closing the test driver
BUILD SUCCESSFUL (total time: 1 minute 44 seconds)

There are a total of 27 Test cases.All of them are filed for reference.
Some points noted based on the testing process
1)Test 1 : T3 and T6 return -1
2)Most tested Default test cases returned -1.
3)The system was run for only one account class and a second initiated account was not tested for. This, although depends on the implementation of the Account class, leaves out few ideas to test the class for a tougher specification requiring certain implementation rules.

Conclutions

The testing was complete successfully for the given conditions.The Business logic given in the EFSM was satisfied by the given code.The Test results are filled along for further analysis.
Testing was done almost completely manually, but automatic testing and analysis can make the process faster.
The test driver, was written to be run manually and an implementation that can combine inputs and run either one test case or a bunch of test cases together can be an improvement.
The entire idea of the Model based testing and Path testing was a process involving concentration and iterations of checking the process.Though the process is a time consuming lengthy one, it would be best done to check if a program satisfies business requirements.
Model Based and State based testing , is not the best processes for a UI rich , Web application of sorts, but will be best used for the back end system processes and mission critical procedures involving transactions etc.

Automation of the Driver, by design would have been a better idea, instead of designing a manual menu based Driver.
Along with a driver, if the output of the test results are going to be from a machine, spending some time to build a result analyser could also be of great use.
A full API, that can run test cases, and analyse its output can be an idea for further projects.

Modified Account Class

/////Start of Source Code ////

```
package accounttestdriver;

/**
 *
 * @author br_prashant
 */
public class Account {

    public Account(int x, int y, int z) {
        bal = x;
        if (bal < 0) {
            bal = 0;
        }
        pin_num = y;
        locked = 0;
        lg = 0;
        account_num = z;
        penalty = 1;
        min_bal = 100;
        k = 0;
        num = 3;
    }

    public int deposit(int d) {
        if (lg != 2) {
            return -1;
        }
        if (locked != 0) {
            return -1;
        };
        if (bal < min_bal) {
            if (d > 0) {
                bal = bal + d - penalty;
```

```
            return 0;
        }
    } else {
        if (d > 0) {
            bal = bal + d;
            return 0;
        }
    }
    return -1;
}

public int withdraw(int w) {
    if (lg != 2) {
        return -1;
    }
    if (locked != 0) {
        return -1;
    };
    if (bal > w) {
        if (w > 0) {
            if (bal < min_bal) {
                return -1;
            } else {
                bal = bal - w;
            };
            if (bal < min_bal) {
                bal = bal - 1;
            }
            return 0;
        }
    }
    return -1;
}

public int balance() {
    if (lg != 2) {
        return -1;
    }
    return bal;
}

public int lock(int x) {
    if (lg != 2) {
        return -1;
```

```
    }
    if (x != pin_num) {
        return -1;
    }
    if (locked == 0) {
        locked = 1;
        return 0;
    } else {
        return -1;
    }
}

public int unlock(int x) {
    if (lg != 2) {
        return -1;
    }
    if (locked != 0) {
        if (x == pin_num) {
            locked = 0;
            return 0;
        }
    }
    return -1;
}

public int login(int x) {
    if (lg != 0) {
        return -1;
    }
    if (account_num == x) {
        lg = 1;
        k = 0;
        return 0;
    }
    return -1;
}

public int logout() {
    if (lg == 0) {
        return -1;
    }
    if (locked == 1) {
        return -1;
    }
```

```
    lg = 0;
    return 0;
}

public int pin(int x) {
    if (lg != 1) {
        return -1;
    }
    if (x == pin_num) {
        lg = 2;
        return 0;
    } else {
        k++;
    }
    if (k >= num) {
        lg = 0;
    }
    return -1;
}
private int bal;
private int locked;
private int pin_num;
private int lg;
private int account_num;
private int penalty;
private int min_bal;
private int k;
private int num;

//getter for balance
public int getBal(){
    return bal;
}
// getter for locked status flag
public int getLocked(){
    return locked;
}
//getter for log in status flag
public int getLg(){
    return lg;
}
//getter for account number
public int getAccNum(){
    return account_num;
```

```
    }
    //getter for Penalty value
    public int getPenalty(){
        return penalty;
    }
    //getter for Min balance value
    public int getMinBal(){
        return min_bal;
    }
    //getter for K count holder
    public int getK(){
        return k;
    }
    //getter for account pin holder.
    public int getNum(){
        return num;
    }
}
/// END of Source Code  ///
```

AccountTestDriver Class

```
/// Start of Source Code ///
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package accounttestdriver;

/**
 *
 * @author br_prashant
 */
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class AccountTestDriver {

    /**
     * @param args the command line arguments
     */
    private static Account accObj;
    public static void main(String[] args) {

        // TODO code application logic here
```

```
    AccountTestDriver driver=new AccountTestDriver();

    System.out.println("/>/> 0 to quit testing /</<");
    int run=1;

    while  (run!=0){
        //display function options
        driver.DisplayOptions();
        //get input option
        run=driver.getInput("Function Choice");
        if (run > 0 && run <=9) {
            driver.processOption(run);
        }
        else{
            System.out.println("Please enter a valid Function choise");
        }
    }

    System.out.println("Closing the test driver");

}

public Integer getInput(String message) {
    try {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        System.out.print(message + " :");
        String input = in.readLine();

        if (input.isEmpty()) {
            getInput(message);
        }
        return new Integer(input);
    } catch (NumberFormatException nfe) {
        System.out.println("Please enter a valid integer.");
    } catch (Exception e) {
        System.out.println("Error reading input: " + e.getMessage());

    }
    return 0;
}

public int DisplayOptions(){
    System.out.println("1.account needs int int int");
    System.out.println("2.login needs int");
```

```
    System.out.println("3.pin needs int");
    System.out.println("4.balance");
    System.out.println("5.deposit needs int");
    System.out.println("6.withdraw needs int");
    System.out.println("7.lock needs int");
    System.out.println("8.unlock needs int");
    System.out.println("9.logout");
    return 0;
}

    public int processOption(int option) {
    if (accObj != null) {
        switch (option) {
            case 1:
                System.out.println("Account class already created");
                break;
            case 2:
                login();
                break;
            case 3:
                pin();
                break;
            case 4:
                balance();
                break;
            case 5:
                deposit();
                break;
            case 6:
                withdraw();
                break;
            case 7:
                lock();
                break;
            case 8:
                unlock();
                break;
            case 9:
                logout();
                break;
            case 0:
                break;
            default:
                break;
```

```
            }
        } else {
            if (option == 1) {
                initAccount();
            } else {
                System.out.println("Initialize account first");
            }
        }
        return 0;
    }

    private void initAccount() {
        int balance = getInput("Enter balance");
        int pin = getInput("Enter PIN");
        int id = getInput("Enter account id");

        accObj = new Account(balance, pin, id);
        System.out.println("Successfully initialized account.");
        printStatus();
    }
    private void account(){
        printStatus();
    }
    private void login(){
        int login=getInput("Enter the login id");
        printStatus();
        if(accObj!=null)
            System.out.println("The Login function executed with a result of " + accObj.login(login)) ;
        printStatus();
    }
    private void pin(){
        int pin=getInput("Enter a pin");
        printStatus();
        if(accObj!=null)
            System.out.println("The Pin function executed with a result of " + accObj.pin(pin)) ;
        printStatus();
    }
    private void deposit(){
        int deposit=getInput("Enter a deposit value");
        printStatus();
        if(accObj!=null)
            System.out.println("The Deposit function executed with a result of " +
accObj.deposit(deposit));
```

```
        printStatus();
    }
    private void withdraw(){
        int withdraw=getInput("Enter a withdraw ammount");
        printStatus();
        if(accObj!=null)
            System.out.println("The Withdraw function executed with a result of " +
accObj.withdraw(withdraw));
        printStatus();
    }
    private void balance(){
        printStatus();
        if(accObj!=null)
            System.out.println("The balance function executed with a result of " + accObj.balance());
        printStatus();
    }
    private void lock(){
        int lock=getInput("Enter the lock pin");
        printStatus();
        if(accObj!=null)
            System.out.println("The Lock function executed with a result of " + accObj.lock(lock));
        printStatus();

    }
    private void unlock(){
        int unlock=getInput("Enter the unlock pin");
        printStatus();
        if(accObj!=null)
            System.out.println("The Unlock function with a result of " + accObj.unlock(unlock));
        printStatus();
    }
    private void logout(){
        printStatus();
        if(accObj!=null)
            System.out.println("The Logout function executed with a result of " + accObj.logout());
        printStatus();
    }

    private void printStatus(){
        System.out.print("\nThe current  values in the account class are ");
        System.out.print(" bal :" + accObj.getBal()  );
        System.out.print(" locked :" + accObj.getLocked()  );
        System.out.print(" LG :" + accObj.getLg()  );
        System.out.print(" account_num :" + accObj.getAccNum());
```

```
        System.out.print(" penalty :" + accObj.getPenalty()  );
        System.out.print(" min_bal :" + accObj.getMinBal()  );
        System.out.print(" K :" + accObj.getK()  );
        System.out.print(" Num :" + accObj.getNum() + "\n"  );
    }
}
```

/// END of Source Code ///

Deliverable

The delivarables are in the folder this report is to be found.
The README file to be found in the folder is also attached here for cross reference.

>>README.txt <<<
This is the final project deliverable of the CS 589 Course of Fall 2012.
The files to be found in the folder include
0) Given Files including the Problem Statement etc.
1) Account Test Driver Netbeans project Folder
2) AccountTestDriver jar file to execute the application
3) Test Suit - ts.txt
4) Manual Test Results Folder
5) >>>>Final Report<<<<<
6) >>>>Final REPORT Supporting Files<<<<<

To run the Test Driver, copy the jar file to a location in your local machine with Java 1.6 installed for execution.
use the command
java -jar AccountTestDriver.jar
Navigate the program to run the test cases.

For sample test cases that include all the required test cases as described in the problem Statement, use the The Test Suit tx.txt file for reference.

To View the source code of the test Driver class and modified Account class,
Navigate to
~/AccountTestDriver/src/accounttestdriver/
You will find 1) Account.java and 2) AccountTestDriver.java

To View the manually run test results
Navigate to
~/TestResults/

Test *.txt is the result for each test corresponding in the ts.txt Test Suit.
>>END of README.txt<<<

THE END