CS-589

SOFTWARE TESTING AND ANALYSIS PROJECT

Object-Oriented and State-Based Testing Of Accounting System

SHAH PREYANG MUKESH

A20385823

Introduction

1. Model Based Transition-pair Testing

State	Incoming Transition	Outgoing Transition	Transition Pair
Idle	T1,T7,T9,T10,T5,T6	T2,T7	(T1,T2),(T1,T7) (T7,T2),(T7,T7) (T9,T2),(T9,T7) (T10,T2),(T10,T7) (T5,T2),(T5,T7) (T6,T2),(T6,T7)
Check pin	T2,T3	T3,T8,T16,T5 ,T6	(T2,T3),(T2,T8),(T2,T16),(T2,T5),(T2,T6), (T3,T3),(T3,T8),(T3,T16),(T3,T5),(T3,T6)
Ready	T16,T11,T12, T3,T17,T18	T10,T14,T4,T13,T12,T11	(T16,T10),(T16,T14),(T16,T4),(T16,T13),(T16,T12),(T16,T11) (T11,T10),(T11,T14),(T11,T4),(T11,T13),(T11,T12),(T11,T11) (T12,T10),(T12,T14),(T12,T4),(T12,T13),(T12,T12),(T12,T11)

			(T13,T10),(T13,T14), (T13,T4), (T13,T13), (T13,T12),(T13,T11) (T17,T10),(T17,T14), (T17,T4), (T17,T13), (T17,T12),(T17,T11), (T18,T10),(T18,T14), (T18,T4), (T18,T13),(T18,T12), (T18,T11)
Locked	T15,T20,T4	T15,T19,T17	(T15,T15),(T15,T19), (T15,T17) (T20,T15),(T20,T19), (T20,T17) (T4,T15),(T4,T19), (T4,T17)
Overdrawn	T8,T21,T22,T19, T14	T21,T22,T20,T18, T9	(T8,T21),(T8,T22), (T8,T20), (T8,T18),(T8,T9) (T21,T21),(T21,T22), (T21,T20), (T21,T18),(T21,T9) (T22,T21),(T22,T22), (T22,T20), (T22,T18),(T22,T9) (T19,T21),(T19,T22), (T19,T20), (T19,T18),(T19,T9) (T14,T21),(T14,T22), (T14,T20), (T14,T18),(T14,T9)

Test Cases:

TEST NO#	TEST CASE :	Transition Covered:	Transition Pair Covered
1	open 222 123 100 login 99 login 98 login 100 pin 122 pin 123 balance deposit 100 deposit 98 balance deposit 140 balance balance withdraw 100 balance logout login 99	T1,T7,T2,T3,T8,T22, T21, T18,T13,T14, T9	(T1,T7), (T7,T7), (T7,T2), (T2,T3), (T3,T8), (T8,T22), (T22,T21), (T21,T21), (T21,T22), (T22,T18), , (T18,T13), (T13,T13), (T13,T14), (T14,T22), (T22,T9), (T9,T7),
2	open 222 123 100 login 100 pin 12 logout login 1000 login 100 pin 123 logout login 100 pin 123 deposit 100 logout login 100 pin 123 deposit 100 deposit 200 logout login 1000 login 100 pin 123 withdraw 10 withdraw 10 balance withdraw 5 deposit 15 deposit 10 balance withdraw 100 logout	T1,T2,T3,T5,T7,T8,T9, T21,T18,T10,T16,T11,T13, T12,T14	(T1,T2),(T2,T3), (T3,T5), (T5,T7), (T7,T2), (T2,T8), (T8,T9), (T9,T2), (T21,T18), (T10,T7), (T2,T16),(T16,T11), (T11,T13),(T11,T11), (T13,T11),(T11,T12), (T12,T12),(T12,T13), (T13,T14) (T18,T10), (T21,T9),(T14,T9)
3	open 1222 123 100 login 100 pin 111 pin 99 pin 112 login 111 login 100 logout login 100 pin 123 balance logout login 100 pin 123 withdraw 10 logout login 100 pin 123 logout login 100 pin 123 deposit 10 withdraw 10 deposit 10 logout	T1,T2,T3,T6,T7,T10,T16, T11,T12,T11	(T1,T2) ,(T2,T3), (T3,T3), (T3,T6), (T6,T7),(T7,T10), (T10,T2),(T2,T16), (T16,T11),(T11,T10), (T16,T10), (T16,T12), (T12,T11), (T11,T12), (T12,T10)
4	open 1000 123 100 login 100 pin 111 pin 98 pin 112 login 100 pin 1234 pin 123 withdraw 600 deposit	T1,T2,T3,T6,T16,T14, T21,T22,T9	(T1,T2),(T2,T3), (T3,T3),(T3,T6), (T6,T2),(T3,T16), (T16,T14), (T21,T22), (T22,T22), (T14,T21), (T22,T9)

	20 balance balance logout		
5	open 100 123 100 login 100 pin 123 deposit 500 withdraw 100 withdraw 10 deposit 40 deposit 40 withdraw 10 logout	T1,T2,T8,T18,T11,T14, T12,T9,T10	(T1,T2),(T2,T8), (T8,T18), (T18,T11), (T11,T14), (T14,T18), (T18,T12), (T12,T11), (T11,T10)
6	open 1000 123 100 login 100 pin 123 lock 15 balance balance unlock 15 withdraw 10 lock 15 unlock 15 deposit 10 lock 15 unlock 15 balance lock 15 unlock 15 lock 15 unlock 15 withdraw 510 deposit 50 lock 15 unlock 15 logout	T1,T2,T16,T4,T15,T17, T11,T12,T13,T14,T18 ,T10	(T16,T4), (T11,T4), (T12,T4), (T13,T4), (T17,T10),(T17,T14), (T17,T4),(T17,T13), (T17,T12),(T17,T11), (T18,T4), (T15,T15),(T15,T17), (T4,T15),(T4,T17)
7	open 200 123 100 login 100 pin 123 lock 15 balance unlock 15 lock 15 unlock 15 balance lock 15 unlock 15 deposit 120 lock 15 unlock 15 deposit 400 withdraw 500 lock 15 unlock 15 logout	T1,T2,T8,T20,T15,T19, T22,T21,T18,T14,T9	(T15,T19), (T20,T15),(T20,T19), (T8,T20), (T21,T20), (T22,T20), (T19,T21),(T19,T22), (T19,T20),(T19,T18), (T19,T9),(T14,T20)
8	open 1000 100 200 login 200 pin 100 balance deposit 100 withdraw 800 logout	T1,T2,T16,T13,T12,T14,T9	(T1,T2),(T2,T16), (T16,T13),(T13,T12), (T12,T14),(T14,T9)

Reason for non-executable Transition Pair:

(T2, T6) :-

- (x! =pn) &&(attempts==2) if this condition satisfies then we can do the T6 transition which cannot be true after transition 2 because in transition T2 we set value of attempt==0.
- So, This transition pair cannot executable.

(T20, T17): -

- Here T20 transition lock the account in locked state from Overdrawn state. While T17 transition is unlocking the account from locked state to Ready state which is not possible because account can be unlocked if its locked from the same state.
- Also, when account is locked state you can not do any other transaction, so the balance condition contradict if we call T20 and T17 transition.

(T4, T19):-

- Here T4 transition lock the account in locked state from Reade state. While T17 transition is unlocking the account from locked state to Overdrawn state which is not possible because account can be unlocked if its locked from the same state.
- Also, when account is locked state you cannot do any other transaction, so the balance condition contradict if we call T4 and T19 transition.

2.Ghost Transition Testing

Test No#	State	Ghost Operation	Test case
9	Start State	Login(int x), Logout(), Pin(int x), Deposit(int d), Withdrawn(int w), Balance(), Lock(int x), Unlock(int x), Open (int -x, int -y, int -z)	login 500 logout withdraw 10 deposit 10 lock 2 unlock 2 balance open -1 232 122 open -1 232 -122 open -1 -232 122 open -1 -232 -122 open 1 232 -122 open 1 -232 -122 open 1 -232 122
10	Idle State	Logout(), Pin(int x), Deposit(int d), Withdrawn(int w), Balance(), Lock(int x), Unlock(int x), Open (-int x, int y, int z)	open 400 123 100 logout open -7 8 9 pin 123 deposit 2 withdraw 1 lock 4 unlock 7 balance login 100 logout open -1 2 -3 open -1 -2 3 open -1 -2 -3
11	Check Pin	Login(int x), Deposit(int d), Withdrawn(int w), Balance(), Lock(int x), Unlock(int x), Open (int x, int y, int z)	open 1000 123 100 deposit 79 withdraw 8 lock 6 unlock 8 balance open 8 9 20 login 5 login 100 logout open 1 2 -3 open 1 -2 3 open -1 -2 -3
12	Ready	Login(int x), Pin(int x), Lock(int x), Unlock(int x), Open(int x,int y,int z)	open 1111 123 100 login 100 pin 123 pin 111 open 1 2 3 login 8 unlock 7 lock 123 logout
13	Overdrawn	Login(int x), pin(int x), Withdrawn(int w), Lock(int x), Unlock(int x),	open 400 123 100 login 100 pin 123 open 8 9 89 login 8 pin 0 withdraw 4 lock 123 unlock 3

		Open(int x,int y,int z)	logout
14	Locked State	Login(int x), Logout(), Deposit(int d), Withdrawn(int w), pin(int x), Lock(int x), Unlock(int x), Open(int x,int y,int z)	open 535 123 100 login 100 pin 123 lock 5 open 1 2 4 login 15 logout lock 125 unlock 7 withdraw 8 deposit 7 pin 111 unlock 5 logout

3.Multiple Condition Testing

Test Case Used for Multiple condition testing

Test Case #15: open 1000 200 100 login 100 pin 200 deposit 0 logout
Test Case #16: open 450 200 100 login 100 pin 200 deposit 0 logout
Test Case #17: open 1000 200 300 login 300 pin 200 withdraw 0 withdraw 2000 logout
Test Case #18: open 500 200 300 login 300 pin 200 withdraw 480 withdraw 0 logout

open () Method:

No.	x>0	x4== (-1)	y>0	z>0	Test Case No#
1	T	T	T	T	T#1
2	T	T	T	F	T#8
3	T	T	F	T	T#8
4	T	T	F	F	T#8
5	T	F	T	T	T#10
6	T	F	T	F	T#10
7	T	F	F	T	T#10
8	T	F	F	F	T#10
9	F	T	T	T	T#8
10	F	T	T	F	T#8
11	F	T	F	T	T#8

12	F	T	F	F	T#8
13	F	F	T	T	T#9
14	F	F	T	F	T#9
15	F	F	F	T	T#9
16	F	F	F	F	T#9

pin () Method:

No.	x4 != (-1)	Test Case No#
1	T	T#8
2	F	T#6

No.	x == x3	Test Case No#
1	T	T#6
2	F	T#1

No.	x9 >= x0	Test Case No#
1	T	T#3
2	F	T#1

login () Method:

No.	x4 != 0	Test Case No#
1	T	T#8
2	F	T#10

No.	x5 == x	Test Case No#
1	T	T#2
2	F	T#10

logout () Method:

No.	x4 == 0	x2 == 1	Test Case No#
1	T	T	Non-Executable
2	T	F	T#9
3	F	T	T#13
4	F	F	T#8

balance () Method:

No.	x4 != 2	Test Case No#
1	T	T#9
2	F	T#4

lock () Method:

No.	x4 != 2	Test Case No#
1	T	T#9
2	F	T#6

No.	x == x3	Test Case No#
1	T	T#11
2	F	T#6

No.	x2 == 0	Test Case No#
1	T	T#6
2	F	T#13

unlock () Method:

No.	x4 != 2	Test Case No#
1	T	T#9
2	F	T#11

No.	x2 == 1	x == x8	Test Case No#
1	T	T	T#6
2	T	F	T#13
3	F	T	T#13
4	F	F	T#13

deposit () Method:

No.	x4 != 2	Test Case No#
1	T	T#9
2	F	T#5

No.	x2 == 1	Test Case No#
1	T	T#13
2	F	T#5

No.	x1 + d < x7	d>0	Test Case No#
1	T	T	T#2
2	T	F	T#15
3	F	T	T#3
4	F	F	T#14

No.	d > 0	Test Case No#
1	T	T#3
2	F	T#14

withdraw () Method:

No.	x4 != 2	Test Case No#
1	T	T#9
2	F	T#5

No.	x2 == 1	Test Case No#
1	T	T#13
2	F	T#5

No.	x1 > w	w>0	Test Case No#
1	T	T	T#4
2	T	F	T#16
3	F	T	T#16
4	F	F	T#17

No.	x1 < x7	Test Case No#
1	T	T#5
2	F	T#4

No.	x1 < x7	Test Case No#
1	T	T#4
2	F	T#5

Reason for non-executable condition:

- In logout() method x4 ==0 or x2 ==1. Here we have 2 condition that cannot be true together because x4==0 indicates that current state is login where x2==1 indicates that current state is locked.
- So, there is no way we can satisfy the condition that are in login state and locked state together. So, this branch is not executable.

4. Test Suite and Its Execution Results

Test Suite

Test#1: open 222 123 100 login 99 login 98 login 100 pin 122 pin 123 balance deposit 100 deposit 98 balance deposit 140 balance balance withdraw 100 balance logout login 99
Test#2: open 222 123 100 login 100 pin 12 logout login 1000 login 100 pin 123 logout login 100 pin 123 deposit 100 logout login 100 pin 123 deposit 100 deposit 200 logout login 1000 login 100 pin 123 withdraw 10 withdraw 10 balance withdraw 5 deposit 15 deposit 10 balance withdraw 100 logout
Test#3: open 1222 123 100 login 100 pin 111 pin 99 pin 112 login 111 login 100 logout login 100 pin 123 balance logout login 100 pin 123 withdraw 10 logout login 100 pin 123 logout login 100 pin 123 deposit 10 withdraw 10 deposit 10 logout
Test#4: open 1000 123 100 login 100 pin 111 pin 98 pin 112 login 100 pin 1234 pin 123 withdraw 600 deposit 20 balance balance logout
Test#5: open 100 123 100 login 100 pin 123 deposit 500 withdraw 100 withdraw 10 deposit 40 deposit 40 withdraw 10 logout
Test#6: open 1000 123 100 login 100 pin 123 lock 15 balance balance unlock 15 withdraw 10 lock 15 unlock 15 deposit 10 lock 15 unlock 15 balance lock 15 unlock 15 lock 15 unlock 15 withdraw 510 deposit 50 lock 15 unlock 15 logout
Test#7: open 200 123 100 login 100 pin 123 lock 15 balance unlock 15 lock 15 unlock 15 balance lock 15 unlock 15 deposit 120 lock 15 unlock 15 deposit 400 withdraw 500 lock 15 unlock 15 logout
Test#8: open 1000 100 200 login 200 pin 100 balance deposit 100 withdraw 800 logout
Test#9: login 500 logout withdraw 10 deposit 10 lock 2 unlock 2 balance open -1 232 122 open -1 232 -122 open -1 -232 122 open -1 -232 -122 open 1 232 -122 open 1 -232 -122 open 1 -232 122

Test#10: open 400 123 100 logout open -7 8 9 pin 123 deposit 2 withdraw 1 lock 4 unlock 7 balance login 100 logout open -1 2 -3 open -1 -2 3 open -1 -2 -3

Test#11: open 1000 123 100 deposit 79 withdraw 8 lock 6 unlock 8 balance open 8 9 20 login 5 login 100 logout open 1 2 -3 open 1 -2 3 open -1 -2 -3

Test#12: open 1111 123 100 login 100 pin 123 pin 111 open 1 2 3 login 8 unlock 7 lock 123 logout

Test#13: open 400 123 100 login 100 pin 123 open 8 9 89 login 8 pin 0 withdraw 4 lock 123 unlock 3 logout

Test#14: open 535 123 100 login 100 pin 123 lock 5 open 1 2 4 login 15 logout lock 125 unlock 7 withdraw 8 deposit 7 pin 111 unlock 5 logout

Test#15: open 1000 200 100 login 100 pin 200 deposit 0 logout

Test#16: open 450 200 100 login 100 pin 200 deposit 0 logout

Test#17: open 1000 200 300 login 300 pin 200 withdraw 0 withdraw 2000 logout

Test#18: open 500 200 300 login 300 pin 200 withdraw 480 withdraw 0 logout

$$

Execution results

Test Case #	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	Actual Result	Expected Result
1	3	400	0	123	0	100	20	500	0	1	PASS	PASS
2	3	462	0	123	0	100	20	500	0	0	PASS	PASS
3	3	1222	0	123	0	100	20	500	0	0	PASS	PASS
4	3	380	0	123	0	100	20	500	0	1	PASS	PASS
5	3	540	0	123	0	100	20	500	0	0	PASS	PASS
6	3	520	0	123	0	100	20	500	15	0	PASS	PASS
7	3	180	0	123	0	100	20	500	15	0	PASS	PASS
8	3	280	0	100	0	200	20	500	0	0	PASS	PASS
9	3	0	0	0	0	0	20	500	0	0	PASS	PASS
10	3	400	0	123	0	100	20	500	0	0	PASS	PASS
11	3	1000	0	123	0	100	20	500	0	0	PASS	PASS
12	3	1111	1	123	2	100	20	500	577	0	PASS	PASS
13	3	400	0	123	0	100	20	500	0	0	PASS	PASS
14	3	535	0	123	0	100	20	500	5	0	PASS	PASS
15	3	1000	0	200	0	100	20	500	0	0	PASS	PASS
16	3	450	0	200	0	100	20	500	0	0	PASS	PASS
17	3	1000	0	200	0	300	20	500	0	0	PASS	PASS
18	3	0	0	200	0	300	20	500	0	0	PASS	PASS

5. Conclusion

- In this project the given Account class has been successfully tested using Model Based Testing, default (ghost) Transition testing and Multiple Condition Testing.
- The project provided great guidance on various ways of implementing the testing strategies between Model Based Testing and Class Based Testing to acquire accurate results.
- The comparison between the two model can be more accurate and Less Time Consuming if the Automated Testing Oriented Methods are implemented to view the states like TestDriver Which is implemented here.
- All state with one condition in it worked as branch testing in multiple testing techniques.
- With the help of Automated testing scripts, we can generate the results which can be later used for report outcomes.
- I also found some defects as if Program is in the start state and try to call logout operation(ghost operation) it will reach to idle state which is not possible as we can see in the EFSM diagram.
- Finally, this project has been a good learning experience for me. I learn many things regarding testing techniques and its results execution techniques.

6.Source Code of both Class

- **TestDriver.java:**

```
import java.util.Scanner;

public class TestDriver {

      @SuppressWarnings("resource")
      public static void main(String[] args) {
            // TODO Auto-generated method stub

      account ac=new account();
    System.out.println("CS-589 Final Project:");
      System.out.println("Driver for Account class.");
       Scanner sc = new Scanner(System.in);
     String s="c";


       while(s.equals("c")){
                  System.out.println("Description of the Account
class:\n0. open (int b, int p,int id) - the account class is activated
which represents the balance by b, represents the pin by p and
represents the login id by id \n 1.login(x)- try to log in for enter in
the system\n"
+"2. logout() - logout from the system\n"
+"3. int pin(int x) -  provides pin # (parameter x)\n"
+"4. int deposit (int d) -  deposits amount d to the account\n"
+"5. int withdraw (int w)-  withdraws amount w from the account\n"
+"6. int balance () -  returns the value of the account balance\n"
+"7. int lock (int x) -  locks an account where x is the lock #\n"
+"8. int unlock (int x) - unlock the account where x is the unlock
id");
                  System.out.println();
                  System.out.println();
                  System.out.println("Testing Oriented Methods:");
                  System.out.println();
                  System.out.println("9.Show_balance()");
                  System.out.println("10.Show_all_values()");
                  System.out.println("11.Show State values");
                  System.out.println();
                  System.out.println("Enter your choice:");

            String i = sc.next();
            int r=0;

     switch(i)
```

```
{
case "0":
   {
              Scanner sc1 = new Scanner(System.in);
              System.out.println("You Pressed 0");
              System.out.println("You called open method");
            System.out.println("Enter balance-b:");
              int b = sc1.nextInt();
              System.out.println("Enter pin-p:");
              int p = sc1.nextInt();
              System.out.println("Enter id for login -id:");
              int id = sc1.nextInt();
              r=ac.open(b, p,id);
            System.out.println("Result-"+r);
             break;
     }
case "1":
  {
              Scanner sc1 = new Scanner(System.in);
              System.out.println("You Pressed 1");
             System.out.println("So you called login
             method");
                   System.out.println("Enter login
                   credential - x:");
              int x = sc1.nextInt();
              r=ac.login(x);
              System.out.println("Result-"+r);
             break;
  }

case "2":
   {
       System.out.println("You Pressed 2");
       System.out.println("So you called logout method");
       r=ac.logout();
         System.out.println("Result-"+r);
         break;
   }
case "3":
       {
              Scanner sc1 = new Scanner(System.in);
              System.out.println("You Pressed 3");
            System.out.println("So you called pin method");
            System.out.println("Enter pin p#:");
              int pn = sc1.nextInt();
              r=ac.pin(pn);
            System.out.println("Result-"+r);
            break;
       }
case "4":
  {
          Scanner sc1 = new Scanner(System.in);
          System.out.println("You Pressed 4");
System.out.println("So you called deposit method");
          System.out.println("Enter The amount you want to
          deposit d:");
          int d = sc1.nextInt();
```

```
                    r=ac.deposit(d);
                    System.out.println("Result-"+r);
                    break;
                  }
         case "5":
                 {

                       Scanner sc1 = new Scanner(System.in);
                       System.out.println("You Pressed 5");
                             System.out.println("So you called
                       withdraw method");
                       System.out.println("Enter the amount you want
                       to withdraw:");
                       int w = sc1.nextInt();
                       r=ac.withdraw(w);
                   System.out.println("Result-"+r);
                   break;
                   }
         case "6":{
                       System.out.println("You Pressed 6");
                       System.out.println("So you called balance
                       method");
                       r=ac.balance();
                       System.out.println("your balance is: "+r);
                       break;
                 }
         case "7":{
                       Scanner sc1 = new Scanner(System.in);
                       System.out.println("You Pressed 7");
                       System.out.println("So you called lock
                       method");
                       System.out.println("your account is locked with
                       id:");
                       int id = sc1.nextInt();
                       r=ac.lock(id);
                       System.out.println("Result-"+r);
                       break;
               }
         case "8":
              {
                       Scanner sc1 = new Scanner(System.in);
                       System.out.println("You Pressed 8");
                       System.out.println("So you called unlock
                       method");
                       System.out.println("you want to unlock account
                       with id:");
                       int id1 = sc1.nextInt();
                       r=ac.unlock(id1);
                       System.out.println("Result-"+r);
                       break;
              }
         case "9":
         {
                       System.out.println("You Pressed 9");
                       System.out.println("The balance is : ");
                       r=ac.show_balance();
                       System.out.println("Result-"+r);
```

```
                        break;
                }
                case "10" :
                {
                        System.out.println("You Pressed 10");
                        System.out.println("You can see all the values
                        of below:");
                        ac.show_all_values();
                        break;
                }
                case "11" :
                {
                        System.out.println("You Pressed 11");
                        ac.Show_State();
                        //System.out.println("Result-"+r);
                        break;
                }

                default:
                    {
                        System.out.println("wrong choice:");
                    }

                 }
                     System.out.println("Enter ''c'' if you want to
                     continue:");
                     s=sc.next();
             }



        System.out.println("Exit from Test Driver--------->");
        }

    }
```

- **Account.java:**

```
//**********************************************
//************* CLASS ACCOUNT ******************
//**********************************************
public class account
{
        private int x0;//max attempt of pin
        private int x1;//balance
        private int x2;//locked state flag
        private int x3;//pin
        private int x4;//state -1,0,1,2,3,4(start,idle,check pin,...)
        private int x5;//id
        private int x6;//extra charge if balance go over 500
        private int x7;//balance limit
        private int x8;//
        private int x9;//attempts of pin

public final int show_balance()
{
		System.out.println("Show balance");
		return x1;

	} //testing oriented method

	public final void show_all_values()
	{

		System.out.println("x0:max number of attempt for pin---" + x0);
		System.out.println("x1:balance---" + x1);
		System.out.println("x2:locked state flag---" + x2);
		System.out.println("x3:pin---" + x3);
		System.out.println("x4:Current State---" + x4);
		System.out.println("x5:id---" + x5);
		System.out.println("x6:Charge in $ if balance is below 500---" + 
	x6);
		System.out.println("x7:balance limit---" + x7);
		System.out.println("x8:---" + x8);
		System.out.println("x9:total attempts you made for pin---" + x9);

	}
	public final void Show_State()
	{


		if(x4==0)
		{
			System.out.println("You are in idle state.");
		}
		if(x4==1)
		{
			System.out.println("You are in Check pin  state.");
		}
		if(x4==2 && x1 < x7 && x2==0)
		{
			System.out.println("You are in OverDrawn state.");
		}
		if(x4==2 && x9>=x0)
		{
			System.out.println("You are in idle  state.");
		}
		if(x4==2 && x1>x7 && x2==0 )
		{
			System.out.println("You are in Ready  state.");
		}
		if( x4==2 && x1>x7 && x2==1)
		{
			System.out.println("You are in locked state.");
		}

		if( x4==2 && x1<x7 && x2==1)
		{
			System.out.println("You are in locked state.");
		}
	}

	public account()
	{
		x2 = 0;//locked state flag
		x4 = -1;//state
		x6 = 20;//extra charge over 500 balance
		x7 = 500;//balance limit
		x9 = 0;//attempt of pin
		x0 = 3;//max attempt of pin
	}
	public final int open(int x, int y, int z)
	{


		if ((x > 0) && (x4 == -1) && (y > 0) && (z > 0))
		{
			x1 = x;
			x3 = y;
			x5 = z;
			x4 = 0;
			System.out.println("Set value to open function if all input 
			is correct");
			return 0;
		};
		System.out.println("This operation not affect the transition as 
		it is ghost transition.!!!");
		return -1;
	}
	public final int pin(int x)
	{

		if (x4 != 1)
		{
			System.out.println("This operation not affect the 
			transition as it is ghost transition.!!!");
			return -1;
		}
		if (x == x3)
		{
			x4 = 2;
			System.out.println("You entered right pin.Please 
			continue!!!");
			return 0;
		}
		else
		{
			x9++;
			System.out.println("you enterd wrong pin please try 
			again!");
		}
		if (x9 >= x0)
		{
			System.out.println("too many attempts of pin. Please try to 
			login again!!!");
			x4 = 0;
		}
```

```
            return -1;
        }
public final int logout()
{

        if ((x4 == 0) || (x2 == 1))
        {
            System.out.println("This operation not affect the transition as
            it is ghost transition.!!!");
            return -1;
        }
        x4 = 0;
        System.out.println("log out Successfully!");
        return 0;
}
public final int login(int x)
{

        if (x4 != 0)
        {
            System.out.println("This operation not affect the transition as
            it is ghost transition.!!!");
            return -1;
        }
        if (x5 == x)
        {
            x4 = 1;
            x9 = 0;
            System.out.println("login credential is correct! you are  logged
            in to system.");
            return 0;
        }
        System.out.println("Wrong credential...Please Try Again!!! ");
        return -1;
}
public final int balance()
{

        if (x4 != 2)
        {
            System.out.println("This operation not affect the transition as
            it is ghost transition.!!!");
            return -1;
        }
        System.out.println("your current balance is :");
        return x1;
}
public final int lock(int x)
{

        if (x4 != 2)
        {
            System.out.println("This operation not affect the transition as
            it is ghost transition.!!!");
            return -1;
        }
        if (x == x3)
```

```
		{
			System.out.println("System can not be locked because the
			credential you are trying is same   as pin");
			return -1;
		}
		if (x2 == 0)
		{
			x2 = 1;
			x8 = x;
			System.out.println("You are in locked state.You need to perform
			unlock operation with same credentials.");
			return 0;
		}
		else
		{
			System.out.println("NO operation performed from here!!!");
			return -1;
		}
	}
	public final int unlock(int x)
	{

		if (x4 != 2)
		{
			System.out.println("This operation not affect the transition as
			it is ghost transition.!!!");
			return -1;
		}
		if ((x2 == 1) && (x == x8))
		{
			x2 = 0;
			System.out.println("Your system is unlocked!!");
			return 0;
		}
		else
		{
			System.out.println("No operation performed from here!!");
			return -1;
		}
	}
	public final int deposit(int d)
	{

		if (x4 != 2)
		{
			System.out.println("This operation not affect the transition as
			it is ghost transition.!!!");
			return -1;
		}
		if (x2 == 1)
		{
			System.out.println("You cannot deposit money when your account is
			locked. ");
			return -1;
		};
		if ((x1 + d < x7) && (d>0))
		{
```

```
            x1 = x1 + d - x6;
            System.out.println("Your balance is below 500 so it charged $20
            extra and your new money is deposited!");
            return 0;
        }
        else
        {
            if (d > 0)
            {
                x1 = x1 + d;
                System.out.println("Money deposited!!");
                return 0;
            }
        }
        System.out.println("No operation done!!!");
 return -1;
}
public final int withdraw(int w)
{

if (x4 != 2)
{
        System.out.println("This operation not affect the transition as it is
        ghost transition.!!!");
        return -1;
}
if (x2 == 1)
{
        System.out.println("You can not withdraw money without entering
        pin!!!");
        return -1;
};
if ((x1 > w) && (w > 0))
{
        if (x1 < x7)
        {
            System.out.println("balance is less then balance limit");
            return -1;
        }
        else
        {
            x1 = x1 - w;
            System.out.println("Money Withdraw from account---" + x1);
        };
        if (x1 < x7)
        {
            x1 = x1 - x6;
            System.out.println("Money Withdraw from account with $20 extra
            fees as balance is below 500"+ x1);
        }
         return 0;
        }
        System.out.println("NO operation Done!!!");
        return -1;
        }
}
```