CS 589 - Software Testing Analysis (Fall 2017)

# Project Report

[Accounting System]

Divya Patel

dpatel107@hawk.iit.edu
A20386344

## Deliverables

1. Model-based testing of the Account class
2. Testing default(ghost) transitions of the Account class
3. Multiple-condition testing
4. A Test Suite and the results of its execution
5. Conclusions
6. Source Code

## 1. Model-based testing of the Account class

Firstly, identify all the transition pairs for each state in the EFSM.

**State:**

1. **Idle:**

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

   Outgoing transitions: T2, T7

   Transition Pair: (T1, T2) , (T1, T7), (T5, T2), (T5, T7), (T6, T2), (T6, T7), (T7, T2), (T7, T7), (T9, T2), (T9, T7), (T10, T2), (T10, T7)

2. **check pin:**

   Incoming transitions: T2, T3

   Outgoing transitions: T3, T6, T8, T16, T5

   Transition Pair: (T2, T3) , (T3, T3), (T2, T6), (T3, T6), (T2, T8), (T3, T8), (T2, T16), (T3, T16), (T2, T5), (T3, T5)

3. **Ready:**

   Incoming transitions: T11, T12, T13, T17, T18, T16

   Outgoing transitions: T11, T12, T13, T4, T14, T10

   Transition Pair: (T11, T11) , (T11, T12), (T11, T13), (T11, T4), (T11, T14), (T11, T10),
   (T12, T11) , (T12, T12), (T12, T13), (T12, T4), (T12, T14), (T12, T10),
   (T13, T11) , (T13, T12), (T13, T13), (T13, T4), (T13, T14), (T13, T10),
   (T16, T11) , (T16, T12), (T16, T13), (T16, T4), (T16, T14), (T16, T10),
   (T18, T11) , (T18, T12), (T18, T13), (T18, T4), (T18, T14), (T18, T10)

4. **Locked:**

   Incoming transitions: T4, T15, T20

   Outgoing transitions: T15, T17, T19

   Transition Pair: (T4, T15), (T4, T17), (T4, T19), (T15, T15), (T15, T17), (T15, T19), (T20, T15), (T20, T17), (T20, T19),

5. **overdrawn:**

   Incoming transitions: T8, T14, T19, T21, T22

   Outgoing transitions: T9, T18, T20, T21, T22

   Transition Pair: (T8, T9), (T8, T18), (T8, T20), (T8, T21), (T8, T22),
   (T14, T9), (T14, T18), (T14, T20), (T14, T21), (T14, T22),
   (T19, T9), (T19, T18), (T19, T20), (T19, T21), (T19, T22),
   (T21, T9), (T21, T18), (T21, T20), (T21, T21), (T21, T22),
   (T22, T9), (T22, T18), (T22, T20), (T22, T21), (T22, T22)

**Coverage of Transition Pairs:**

| **Test#** | **Transition Pair Covered** |
|---|---|
| Test#1 | (T1, T7), (T7, T2), (T2, T16), (T16, T10), (T10, T2), (T2, T3), (T3, T3), (T3, T6), (T6, T2), (T2,T5) |
| Test#2 | (T1, T2), (T2, T8), (T8, T9), (T9, T2), (T2, T8), (T8, T21), (T21, T18), (T18, T12), (T12, T13), (T13, T13), (T13, T11), (T11, T13), (T13, T12), (T12, T12), (T12, T4), (T4, T15), (T15, T15), (T15, T17), (T17, T14), (T14, T20), (T20, T19), (T19, T22), (T22, T22), (T22, T21), (T21, T21), (T21, T9) |
| Test#3 | (T1, T2), (T2, T3), (T3, T16), (T16, T11), (T11, T11), (T11, T12), (T12, T11), (T11, T14), (T14, T18), (T18, T13), (T13, T4), (T4, T17), (T17, T11), (T11, T4), (T4, T17), (T17, T12), (T12, T10) |
| Test#4 | (T1, T2), (T2, T3), (T3, T8), (T8, T18), (T18, T11), (T11, T10), (T10, T7), (T7, T7), (T7, T2), (T2, T5), (T5, T2) , (T2, T3) , (T3, T5) |
| Test#5 | (T1, T2), (T2, T16), (T16, T12), (T12, T14), (T14, T21), (T21, T20), (T20, T15), (T15, T19), (T19, T21), (T21, T22), (T22, T18), (T18, T14), (T14, T22), (T22, T9), (T9, T7) |

| | |
|---|---|
| Test#6 | (T1, T2), (T2, T3), (T3, T3), (T3, T6), (T6, T7), (T7, T2), (T2, T5), (T5, T7), (T7, T2), (T2, T5), (T5, T7), (T7, T2), (T2, T16), (T16, T4), (T4, T17), (T17, T13), (T13, T14), (T14, T9) |
| Test#7 | (T1, T2), (T2, T16), (T16, T13), (T13, T10), (T10, T2), (T2, T16), (T16, T14), (T14, T20), (T20, T19), (T19, T18), (T18, T4), (T4, T17), (T17, T4), (T4, T17), (T17, T10) |
| Test#8 | (T1, T2), (T2, T16), (T16, T14), (T14, T18), (T18, T10), (T10, T2), (T2, T16), (T16, T14), (T14, T20), (T20, T19), (T19, T20), (T20, T19), (T19, T9) |
| Test#9 | (T1, T2), (T2, T8), (T8, T20), (T20, T19), (T19, T22), (T22, T20), (T20, T19), (T19, T9), (T9, T2), (T2, T8), (T8, T22) |

**Non-executable transition pairs: (T2, T6), (T4, T19), (T20, T17)**

## 2. Testing default(ghost) transitions of account class

Firstly, identify all the default/ghost transitions for each state in the EFSM.

| State | Default Transition |
|---|---|
| start | open(x,y,z)[x<=0ǁy<=0ǁz<=0], login(x), pin(pn), deposit(d), withdraw(w), balance(), lock(x), unlock(x), logout() |
| idle | open(x,y,z), logout(), pin(pn), deposit(d), withdraw(w), balance(), lock(x), unlock(x) |
| Check pin | open(x,y,z), login(x), deposit(d), withdraw(w), balance(), lock(x), unlock(x) |
| ready | open(x,y,z), login(x), pin(pn), lock(x)[x==pn], unlock(x) |
| locked | open(x,y,z), login(x), logout(), pin(pn), deposit(d), withdraw(w), lock(x1), unlock(x2)[x1!=x2] |
| overdrawn | open(x,y,z), login(x), pin(pn), withdraw(w), lock(x)[x==pn], unlock(x) |

| Test# | Default/ghost Transition Covered |
|---|---|
| Test#21 | Default transition of “start” state |
| Test#10 | Default transition of “idle” |
| Test#11 | Default transition of “check pin” |
| Test#12 | Default transition of “ready” |
| Test#13 | Default transition of “locked” |
| Test#14 | Default transition of “overdrawn” |

## 3. Multiple-condition Testing

Firstly, identify all the multiple conditions for each method in the source code.

**open() :**

1. **Condition : (x>0) && (x4 == -1) && (y>0) && (z>0)**

| **x > 0** | **X4 == -1** | **y>0** | **z>0** | **Covered in Test#** |
|---|---|---|---|---|
| T | T | T | T | Test#1 |
| T | T | T | F | Test#16 |
| T | T | F | T | Test#17 |
| T | T | F | F | Test#18 |
| T | F | T | T | Test#10 |
| T | F | T | F | Test#16 |
| T | F | F | T | Test#17 |
| T | F | F | F | Test#18 |
| F | T | T | T | Test#19 |
| F | T | T | F | Test#20 |
| F | T | F | T | Test#21 |
| F | T | F | F | Test#22 |
| F | F | T | T | Test#19 |
| F | F | T | F | Test#20 |
| F | F | F | T | Test#21 |
| F | F | F | F | Test#22 |

**login(x) :**

2. **Condition : (x4 != 0)**

| **X4!=0** | **Covered in Test #** |
|---|---|
| T | Test#11 |
| F | Test#2 |

3. **Condition: (x5 == x)**

| **X5==X** | **Covered in Test #** |
|---|---|
| T | Test#2 |
| F | Test#1 |

**pin(x) :**

4. **Condition : (x4 != 1)**

| **X4!=1** | **Covered in Test #** |
|---|---|
| T | Test#10 |
| F | Test#1 |

5. **Condition: (x9 >= x0)**

| **X9>=X0** | **Covered in Test #** |
|---|---|
| T | Test#1 |
| F | Test#2 |

**logout() :**

6. **Condition : (x4 == 0) || (x2 == 1)**

| **X4 == 0** | **X2 == 1** | **Covered in Test #** |
|---|---|---|
| T | T | No test case #1 |
| T | F | Test#10 |
| F | T | Test#13 |
| F | F | Test#1 |

**balance() :**

7. **Condition : (x4 != 2)**

| **X4!=2** | **Covered in Test #** |
|---|---|
| T | Test#10 |
| F | Test#2 |

**deposit(d) :**

8. **Condition : (x4 != 2)**

| **X4!=2** | **Covered in Test #** |
|---|---|
| T | Test#11 |
| F | Test#2 |

**9. Condition : (x2 == 1)**

| X2==1 | Covered in Test # |
|---|---|
| T | Test#13 |
| F | Test#2 |

**10. Condition : ((x1 + d < x7) && (d>0))**

| X1 + D < X7 | D > 0 | Covered in Test # |
|---|---|---|
| T | T | Test#3 |
| T | F | Test#27 |
| F | T | Test#5 |
| F | F | Test#23 |

**11. Condition : d>0**

| D > 0 | Covered in Test # |
|---|---|
| T | Test#5 |
| F | Test#23 |

**withdraw(w) :**

**12. Condition : (x4 != 2)**

| X4!=2 | Covered in Test # |
|---|---|
| T | Test#11 |
| F | Test#3 |

**13. Condition : (x2 == 1)**

| X2==1 | Covered in Test # |
|---|---|
| T | Test#13 |
| F | Test#3 |

**14. Condition: (x1 > w) && (w > 0)**

| X1 > W | W > 0 | Covered in Test # |
|---|---|---|
| T | T | Test#3 |
| T | F | Test#26 |
| F | T | Test#15 |
| F | F | Test#28 |

**15. Condition : (x1 < x7)**

| X1<X7 | Covered in Test # |
|---|---|
| T | Test#14 |
| F | Test#4 |

**16. Condition : (x1 < x7)**

| X1<x7 | Covered in Test # |
|---|---|
| T | Test#3 |
| F | Test#3 |

**lock(x) :**

**17. Condition : (x4 != 2)**

| X4!=2 | Covered in Test # |
|---|---|
| T | Test#11 |
| F | Test#2 |

**18. Condition : (x == x3)**

| X==X3 | Covered in Test # |
|---|---|
| T | Test#12 |
| F | Test#2 |

**19. Condition : (x2 == 0)**

| X2==0 | Covered in Test # |
|---|---|
| T | Test#9 |
| F | Test#13 |

**unlock(x) :**

**20.Condition : (x4 != 2)**

| X4!=2 | Covered in Test # |
|---|---|
| T | Test#10 |
| F | Test#2 |

**21. Condition : (x2 == 1) && (x == x8)**

| X2==1 | X==X8 | Covered in Test # |
|---|---|---|
| T | T | Test#2 |
| T | F | Test#13 |
| F | T | Test#25 |
| F | F | Test#26 |

**Reason for Non-executable branches :**

| No Test Case # | Condition : Combination | Reason |
|---|---|---|
| 1 | (x4==0) \|\| (x2==1) : TT | When x2==1 is true then x4 should be 2 and if x4==0 then x2 should be 0. So, both conditions can not be true at a time. |

## 4. A Test Suite and the results of its execution

**TS.txt :**

Test#1: open 500 111 100 login 000 login 100 pin 111 logout login 100 pin 11 pin 12 pin 12 login 100 logout

Test#2: open 400 111 100 login 100 pin 111 logout login 100 pin 111 deposit 50 deposit 70 deposit 100 balance balance withdraw 10 balance deposit 100 deposit 10 lock 11 balance balance unlock 11 withdraw 300 lock 11 unlock 11 balance balance deposit 60 deposit 100 logout

Test#3: open 600 111 100 login 100 pin 2 pin 111 withdraw 10 withdraw 40 deposit 50 withdraw 50 withdraw 100 deposit 100 balance lock 2 unlock 2 withdraw 20 lock 2 unlock 2 deposit 70 logout

Test#4: open 400 111 100 login 100 pin 200 pin 111 deposit 200 withdraw 50 logout login 2 login 1 login 100 logout login 100 pin 1 logout

Test#5: open 500 111 100 login 100 pin 111 deposit 100 withdraw 200 deposit 20 lock 1 balance unlock 1 deposit 50 balance deposit 200 withdraw 400 balance logout login 1

Test#6: open 500 111 100 login 100 pin 2 pin 2 pin 2 login 1 login 100 logout login 1 login 100 pin 111 lock 3 unlock 3 balance withdraw 100 logout

Test#7: open 500 111 100 login 100 pin 111 balance logout login 100 pin 111 withdraw 100 lock 2 unlock 2 deposit 200 lock 2 unlock 2 lock 3 unlock 3 logout

Test#8: open 500 111 100 login 100 pin 111 withdraw 100 deposit 200 logout login 100 pin 111 withdraw 200 lock 3 unlock 3 lock 3 unlock 3 logout

Test#9: open 400 111 100 login 100 pin 111 lock 2 unlock 2 balance lock 2 unlock 2 logout login 100 pin 111 balance

Test#10: open 500 111 100 open 500 111 100 logout pin 111 deposit 100 withdraw 10 balance lock 2 unlock 2

Test#11: open 500 111 100 login 100 open 500 111 100 login 111 deposit 100 withdraw 100 balance lock 100 unlock 100

Test#12: open 500 111 100 login 100 pin 111 open 500 111 100 login 100 pin 111 lock 111 unlock 111

Test#13: open 500 111 100 login 100 pin 111 lock 5 open 500 111 100 login 100 logout pin 111 deposit 100 withdraw 100 lock 5 unlock 3

Test#14: open 500 111 100 login 100 pin 111 withdraw 200 open 500 111 100 login 100 pin 111 withdraw 100 lock 111 unlock 5

Test#15: open 500 111 100 login 100 pin 111 withdraw 600 logout

Test#16: open 500 111 0 login 0 open 500 111 00

Test#17: open 500 0 100 open 500 0 100

Test#18: open 500 0 -1 login -1 open 500 0 -1

Test#19: open 0 111 100 open 0 111 100

Test#20: open 0 111 -1 open 0 111 -1

Test#21: open 0 -2 100 open 0 111 0 open 500 0 0 open 0 111 100 open 500 0 100 open 500 111 0 login 100 open 0 -2 100 login 100 pin -2 deposit 100 withdraw 20 balance lock 2 unlock 2 logout

Test#22: open 0 0 0 logout open 0 0 0

Test#23: open 500 111 100 login 100 pin 111 deposit 0

Test#24: open 500 111 100 login 100 pin 111 lock 2 unlock 2 unlock 2

Test#25: open 500 111 100 login 100 pin 111 lock 2 unlock 2 unlock 5

Test#26: open 400 111 100 login 100 pin 111 logout login 100 pin 111 deposit 50 deposit 70 deposit 100 balance balance withdraw 0 logout

Test#27: open 400 111 100 login 100 pin 111 logout login 100 pin 111 deposit 50 deposit 70 deposit 100 balance balance withdraw 10 balance deposit 100 deposit 10 logout

Test#28: open 500 111 100 login 100 pin 111 withdraw 480 withdraw 0 logout

$$

**Results of Execution:**

| Test # | x0 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | x9 | state | Result |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 500 | 0 | 111 | 0 | 100 | 20 | 500 | 0 | 0 | idle | Pass |
| 2 | 3 | 520 | 0 | 111 | 0 | 100 | 20 | 500 | 11 | 0 | idle | Pass |
| 3 | 3 | 580 | 0 | 111 | 0 | 100 | 20 | 500 | 2 | 1 | idle | Pass |
| 4 | 3 | 550 | 0 | 111 | 0 | 100 | 20 | 500 | 0 | 1 | idle | Pass |
| 5 | 3 | 190 | 0 | 111 | 0 | 100 | 20 | 500 | 1 | 0 | idle | Pass |
| 6 | 3 | 380 | 0 | 111 | 0 | 100 | 20 | 500 | 3 | 0 | idle | Pass |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 3 | 580 | 0 | 111 | 0 | 100 | 20 | 500 | 3 | 0 | idle | Pass |
| 8 | 3 | 360 | 0 | 111 | 0 | 100 | 20 | 500 | 3 | 0 | idle | Pass |
| 9 | 3 | 400 | 0 | 111 | 0 | 100 | 20 | 500 | 2 | 0 | idle | Pass |
| 10 | 3 | 500 | 0 | 111 | 0 | 100 | 20 | 500 | 0 | 0 | idle | Pass |
| 11 | 3 | 500 | 0 | 111 | 1 | 100 | 20 | 500 | 0 | 0 | Check pin | Pass |
| 12 | 3 | 500 | 0 | 111 | 2 | 100 | 20 | 500 | 0 | 0 | ready | Pass |
| 13 | 3 | 500 | 1 | 111 | 2 | 100 | 20 | 500 | 5 | 0 | locked | Pass |
| 14 | 3 | 280 | 0 | 111 | 2 | 100 | 20 | 500 | 0 | 0 | overdrawn | Pass |
| 15 | 3 | 500 | 0 | 111 | 0 | 100 | 20 | 500 | 0 | 0 | idle | Pass |
| 16 | 3 | 0 | 0 | 0 | -1 | 0 | 20 | 500 | 0 | 0 | start | Pass |
| 17 | 3 | 0 | 0 | 0 | -1 | 0 | 20 | 500 | 0 | 0 | start | Pass |
| 18 | 3 | 0 | 0 | 0 | -1 | 0 | 20 | 500 | 0 | 0 | start | Pass |
| 19 | 3 | 0 | 0 | 0 | -1 | 0 | 20 | 500 | 0 | 0 | start | Pass |
| 20 | 3 | 0 | 0 | 0 | -1 | 0 | 20 | 500 | 0 | 0 | start | Pass |
| 21 | 3 | 0 | 0 | 0 | 0 | 0 | 20 | 500 | 0 | 0 | idle | Pass |
| 22 | 3 | 0 | 0 | 0 | -1 | 0 | 20 | 500 | 0 | 0 | start | Pass |
| 23 | 3 | 500 | 0 | 111 | 2 | 100 | 20 | 500 | 0 | 0 | ready | Pass |
| 24 | 3 | 500 | 0 | 111 | 2 | 100 | 20 | 500 | 2 | 0 | ready | Pass |
| 25 | 3 | 500 | 0 | 111 | 2 | 100 | 20 | 500 | 2 | 0 | ready | Pass |
| 26 | 3 | 600 | 0 | 111 | 0 | 100 | 20 | 500 | 0 | 0 | idle | Pass |
| 27 | 3 | 700 | 0 | 111 | 0 | 100 | 20 | 500 | 0 | 0 | idle | Pass |
| 28 | 3 | 0 | 0 | 111 | 0 | 100 | 20 | 500 | 0 | 0 | idle | Pass |

- Here, in test case 21 the final state should be "start" state. But, there is no condition in logout method for the "start" state that it should be return -1 at "start" state. So, it is going to the "idle" state instead of being there only.

## 5. Conclusion

By doing this project, I got a lot of knowledge for object oriented testing. I understood that one of the most costly and tedious process in the software development cycle is testing.

It is tough to obtain the state of each method in class testing. Thus, it will be helpful to have automated testing oriented methods.

## 6. Source Code

### Account.java:

```
public class Account
{
        private int x0;
        private int x1;
        private int x2;
        private int x3;
        private int x4;
        private int x5;
        private int x6;
        private int x7;
        private int x8;
        private int x9;


        //constructor of the Account class
        public Account()
        {
                x2 = 0;
                x4 = -1;
                x6 = 20;
                x7 = 500;
                x9 = 0;
                x0 = 3;
        }

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("\nAccount has been opened successfully....\n");
			return 0;
		};
		System.out.println("\nSorry,something went wrong !!!\n");
		return -1;
	}
	public final int pin(int x)
	{
		if (x4 != 1)
		{
			System.out.println("\nSorry,you can not perform this operation at this state 
!!!\n");
			return -1;
		}
		if (x == x3)
		{
			System.out.println("\nCorrect pin!!\n");
			x4 = 2;
			return 0;
		}
		else
		{
			System.out.println("\nWrong pin! Please try again...\n");
			x9++;
		}
		if (x9 >= x0)
		{
			System.out.println("\nSorry, too many attempts.!!! Login again to move 
forward.\n");
			x4 = 0;
		}
		System.out.println("\nSorry,something went wrong !!!\n");
		return -1;
	}
	public final int logout()
	{
		if ((x4 == 0) || (x2 == 1))

		{
			System.out.println("Sorry, you can not perform this operation at this 
state!!!");
			return -1;
```

```
                }
                System.out.println("\nlogged out successfully!!!!\n");
                x4 = 0;
                return 0;
        }
        public final int login(int x)
        {
                if (x4 != 0)
                {
                        System.out.println("\nSorry,you can not perform this operation at this
state!!!\n");
                        return -1;
                }
                if (x5 == x)
                {
                        x4 = 1;
                        x9 = 0;
                        System.out.println("\nsuccessfully logged in!!\n");
                        return 0;
                }
                System.out.println("\nWrong Credentials!!\n");
                return -1;
        }
        public final int balance()
        {
                if (x4 != 2)
                {
                        System.out.println("\nSorry,you can not perform this operation at this
state!!!\n");
                        return -1;
                }
                System.out.println("your balance is" + x1);
                return x1;
        }
        public final int lock(int x)
        {
                if (x4 != 2)
                {
                        System.out.println("\nSorry,you can not perform this operation at this
state!!!\n");
                        return -1;
                }
                if (x == x3)
                {


                        return -1;
                }
```

```
		if (x2 == 0)
		{
			x2 = 1;
			x8 = x;
			System.out.println("\naccount is locked.....:(\n");
			return 0;
		}
		else
		{
			System.out.println("\nSorry,something went wrong !!!\n");
			return -1;
		}
	}
	public final int unlock(int x)
	{
		if (x4 != 2)
		{
			System.out.println("\nSorry,you can not perform this operation at this 
state!!!\n");
			return -1;
		}
		if ((x2 == 1) && (x == x8))
		{
			x2 = 0;
			System.out.println("\naccount unlocked.....:)\n");
		return 0;
		}
		else
		{
			System.out.println("\nSorry,something went wrong !!!\n");
			return -1;
		}
	}
	public final int deposit(int d)
	{
		if (x4 != 2)
		{
			System.out.println("\nSorry,you can not perform this operation at this 
state!!!\n");
			return -1;
		}
		if (x2 == 1)
		{
			System.out.println("\nYou have to unlock your account first...\n");
			return -1;
		};
		if ((x1 + d < x7) && (d>0))
		{
			x1 = x1 + d - x6;
			System.out.println("\namount has been deposited successfully\n");
			return 0;
		}
		else
		{
			if (d > 0)
			{
			x1 = x1 + d;
			System.out.println("\namount has been deposited successfully\n");

			return 0;
			}
		}
		System.out.println("\nSorry,something went wrong !!!\n");
		return -1;
	}
	public final int withdraw(int w)
	{
		if (x4 != 2)
		{
			System.out.println("\nSorry,you can not perform this operation at this
state\n");
			return -1;
		}
		if (x2 == 1)
		{
			System.out.println("\nYou have to unlock your account first...\n");
			return -1;
		}
		if ((x1 > w) && (w > 0))
		{
			if (x1 < x7)
			{
				return -1;
			}
			else
			{
				x1 = x1 - w;
				System.out.println("\namount has been withdrawn
successfully...\n");

			};
			if (x1 < x7)
			{
				x1 = x1 - x6;
```

```
                    }
                return 0;
            }
            return -1;
    }

    //testing purpose method
    public void Show_all() {


            System.out.println("\nx0= "+ x0);
            System.out.println("x1= "+ x1);
            System.out.println("x2= "+ x2);
            System.out.println("x3= "+ x3);
            System.out.println("x4= "+ x4);
            System.out.println("x5= "+ x5);
            System.out.println("x6= "+ x6);
            System.out.println("x7= "+ x7);
            System.out.println("x8= "+ x8);
            System.out.println("x9= "+ x9 + "\n");

            if(x4==0) {
                    System.out.println("\nCurrent State is \"idle\"\n");
            }
            else if(x4==1) {
                    System.out.println("\nCurrent State is \"check pin\"\n");
            }
            else if(x4==2) {
                    if(x2==1) {
                            System.out.println("\nCurrent State is \"locked\"\n");
                    }
                    else if(x1<x7) {
                            System.out.println("\nCurrent State is \"overdrawn\"\n");
                    }
                    else {
                            System.out.println("\nCurrent State is \"ready\"\n");
                    }
            }else {
                    System.out.println("\nCurrent State is \"starting state\"\n");
            }

            System.out.println("\nCurrent Balance: $"+x1 + "\n");


    }
```

```
//testing purpose method
        public void Show_state() {
                if(x4==0) {
                        System.out.println("\nCurrent State is \"idle\"\n");
                }
                else if(x4==1) {
                        System.out.println("\nCurrent State is \"check pin\"\n");
                }
                else if(x4==2) {
                        if(x2==1) {
                                System.out.println("\nCurrent State is \"locked\"\n");
                        }
                        else if(x1<x7) {
                                System.out.println("\nCurrent State is \"overdrawn\"\n");
                        }
                        else {
                                System.out.println("\nCurrent State is \"ready\"\n");
                        }
                }else {
                        System.out.println("\nCurrent State is \"starting state\"\n");
                }

        }

        //testing purpose method
        public void Show_balance() {
                System.out.println("\nCurrent Balance: $"+x1 + "\n");

        }


}
```

**TestDriver.java**:

```
import java.util.Scanner;


public class TestDriver {
        public static void main(String[] args) {
                // TODO Auto-generated method stub


                Account ac=new Account();
                System.out.println("STA Project Demo:");
                System.out.println("Driver for Account class.");
                Scanner sc = new Scanner(System.in);




                String s="y";
                while(s.equals("y")){
                        System.out.println("Description of the Account class:\n1. open(int x, int y, int
z) - sets balance to the value of x, pin number to the value of y, and an account # to the value of
z\n"
                                        +"2. login() - allows to login to the account, where x is an
account # \n"
                                        +"3. logout() -  allows to logout from the account\n"
                                        +"4. pin(int x) -  provides pin # (parameter x)\n"
                                        +"5. deposit(int d) - deposits amount d to the account\n"
                                        +"6. withdraw(int w) -  withdraws amount w from the account
\n"
                                        +"7. balance() -  returns the value of the account balance \n"
                                        +"8. lock(int x) -  locks an account where x is the lock # \n"
                                        +"9. unlock(int x) -  unlocks an account when x equals to the
correct lock # \n");

                        System.out.println("Testing Oriented Methods:");

                        System.out.println("10.Show_all()");
                        System.out.println("11.Show_state()");
                        System.out.println("12.Show_balance()");

                        System.out.println("Enter ur choice:");
                        String i = sc.next();
                        int r=0;
                        switch(i)
                        {
```

```
case "1":{
        System.out.println("\nEnter balance amount :");
        int x = sc.nextInt();
        System.out.println("\nEnter pin :");
        int y = sc.nextInt();
        System.out.println("\nEnter account #:");
        int z = sc.nextInt();
        r=ac.open(x,y,z);
        System.out.println("Result-"+r);
        break;  }
case "2":{
        System.out.println("\nEnter account # to login :");
        int x = sc.nextInt();
        r=ac.login(x);
        System.out.println("Result-"+r);
        break;}



case "3":{
        r=ac.logout();
        System.out.println("Result-"+r);
        break;}
case "4":{
        System.out.println("\nEnter pin to go ahead :");
        int x = sc.nextInt();
        r=ac.pin(x);
        System.out.println("Result-"+r);
        break;}
case "5":{
        System.out.println("\nEnter amount you want to deposit :");
        int d = sc.nextInt();
        r=ac.deposit(d);
        System.out.println("Result-"+r);
        break;}
case "6":{
        System.out.println("\nEnter amount you want to withdraw  :");
        int w = sc.nextInt();
        r=ac.withdraw(w);
        System.out.println("Result-"+r);
        break;}
case "7":{
        r=ac.balance();
        System.out.println("Result-"+r);
        Break;}
```

```
                        case "8":{
                                System.out.println("Enter lock # :");
                                int x = sc.nextInt();
                                r=ac.lock(x);
                                System.out.println("Result-"+r);
                                break;}
                        case "9":{
                                System.out.println("Enter  correct lock # to unlock :");
                                int x = sc.nextInt();
                                r=ac.unlock(x);
                                System.out.println("Result-"+r);
                                break;}

                        case "10":{ac.Show_all();break;}

                        case "11":{ac.Show_state();break;}

                        case "12":{ac.Show_balance();break;}

                        default:{System.out.println("wrong choice:");}
                        System.out.println("Enter "y" if you want to continue:");
                        s=sc.next();
        }
  }
  System.out.println("<*********************Exit from Test Driver************************>");
 }
}
```