MDA-EFSM Events:
Open()
Login()
IncorrectLogin()
IncorectPin(int max)
CorrectPinBelowMin()
CorrectPinAboveMin()
Deposit()
BelowMinBalance()
AboveMinBalance()
Logout()
Balance()
Withdraw()
WithdrawBelowMinBalance()
NoFunds()
Lock()
IncorrectLock()
Unlock()
IncorrectUnlock()
Suspend()
Activate()
Close()

MDA-EFSM Actions:
A1: StoreData() // stores pin from temporary data store to *pin* in data store
A2: IncorrectIdMsg() // displays incorrect ID message
A3: IncorrectPinMsg() // displays incorrect pin message
A4: TooManyAttemptsMsg() // display too many attempts message
A5: DisplayMenu() // display a menu with a list of transactions
A6: MakeDeposit() // makes deposit (increases balance by a value stored in temp. data store)
A7: DisplayBalance() // displays the current value of the balance
A8: PromptForPin() // prompts to enter pin
A9: MakeWithdraw() // makes withdraw (decreases balance by a value stored in temp. data store)
A10: Penalty() // applies penalty (decreases balance by the amount of penalty)
A11: IncorrectLock Msg() // displays incorrect lock msg
A12: IncorrectUnlock Msg() // displays incorrect unlock msg
A13: NoFundsMsg() // Displays no sufficient funds msg

Operations of the Input Processor
(ACCOUNT-1)

```
open (string p, string y, float a) {
// store p, y and a in temp data store
        ds->temp_p=p;
        ds->temp_y=y;
        ds->temp_a=a;
        m->Open();
}

pin (string x) {
        if (x==ds->pin) {
            if (d->balance > 500)
                m->CorrectPinAboveMin ();
            else m->CorrectPinBelowMin();
        }
        else m->IncorrectPin(3)
}


deposit (float d) {
        ds->temp_d=d;
        m->Deposit();
        if (ds->balance>500)
            m->AboveMinBalance();
        else m->BelowMinBalance();
}

withdraw (float w) {
        ds->temp_w=w;
        m->withdraw();
        if ((ds->balance>500)
             m->AboveMinBalance();
        else m->WithdrawBelowMinBalance();
}
```

```
balance() {m->Balance();}

login (string y) {
        if (y==ds->uid)
            m->Login();
        else m->IncorrectLogin();
}

logout() {m->Logout();}

lock (string x) {
        if (ds->pin==x) m->Lock();
        else m->IncorrectLock();
}

unlock (string x) {
        if (x==ds->pin) {
            m->Unlock();
            if (ds->balance > 500)
                m->AboveMinBalance ();
            else m->BelowMinBalance();
        }
        else m->IncorrectUnlock();
}
```

Notice:
m: is a pointer to the MDA-EFSM object
ds: is a pointer to the Data Store object which contains the following data items:

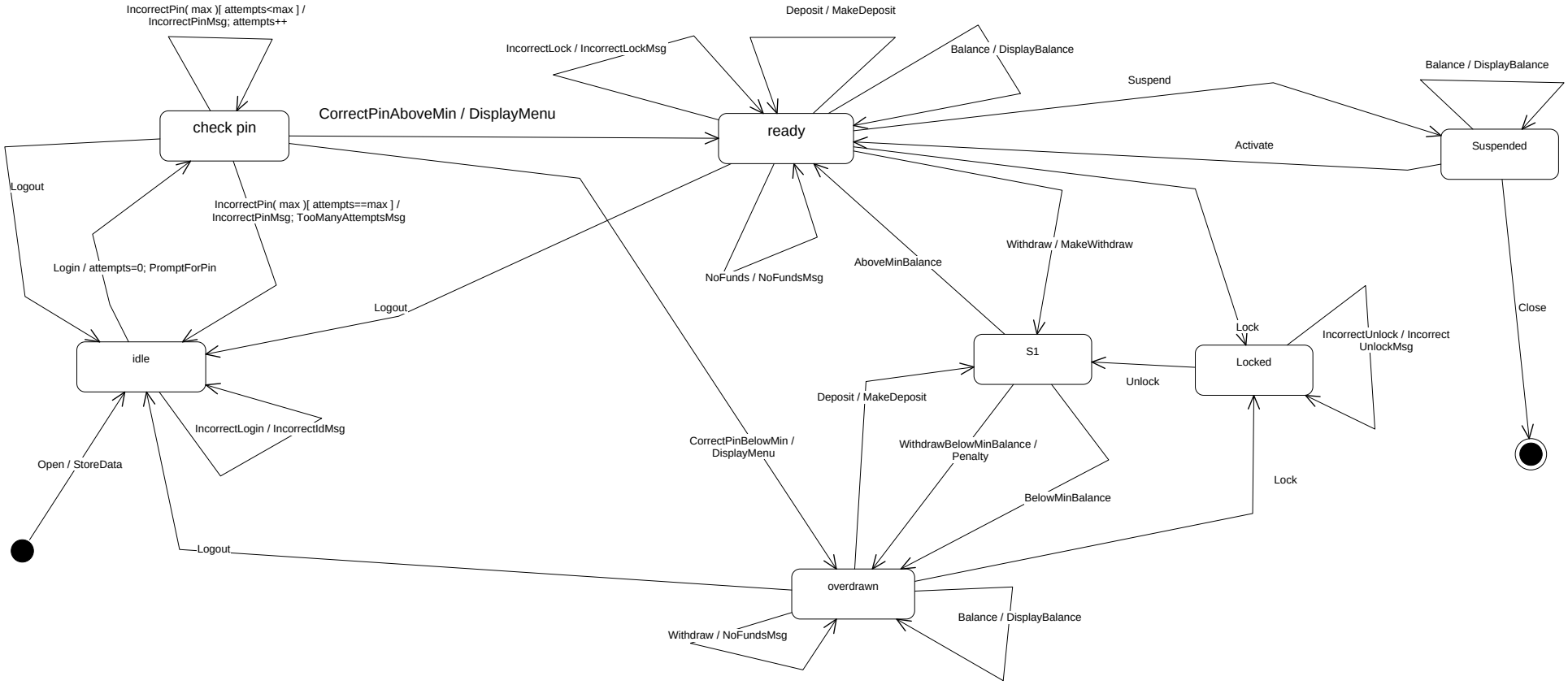
- *balance*: contains the current balance
- *pin*: contains the correct pin #
- *uid:* contains the correct user ID
- *temp_p, temp_y, temp_a, temp_d, temp_w* are used to store values of parameters

Operations of the Input Processor (ACCOUNT-2)

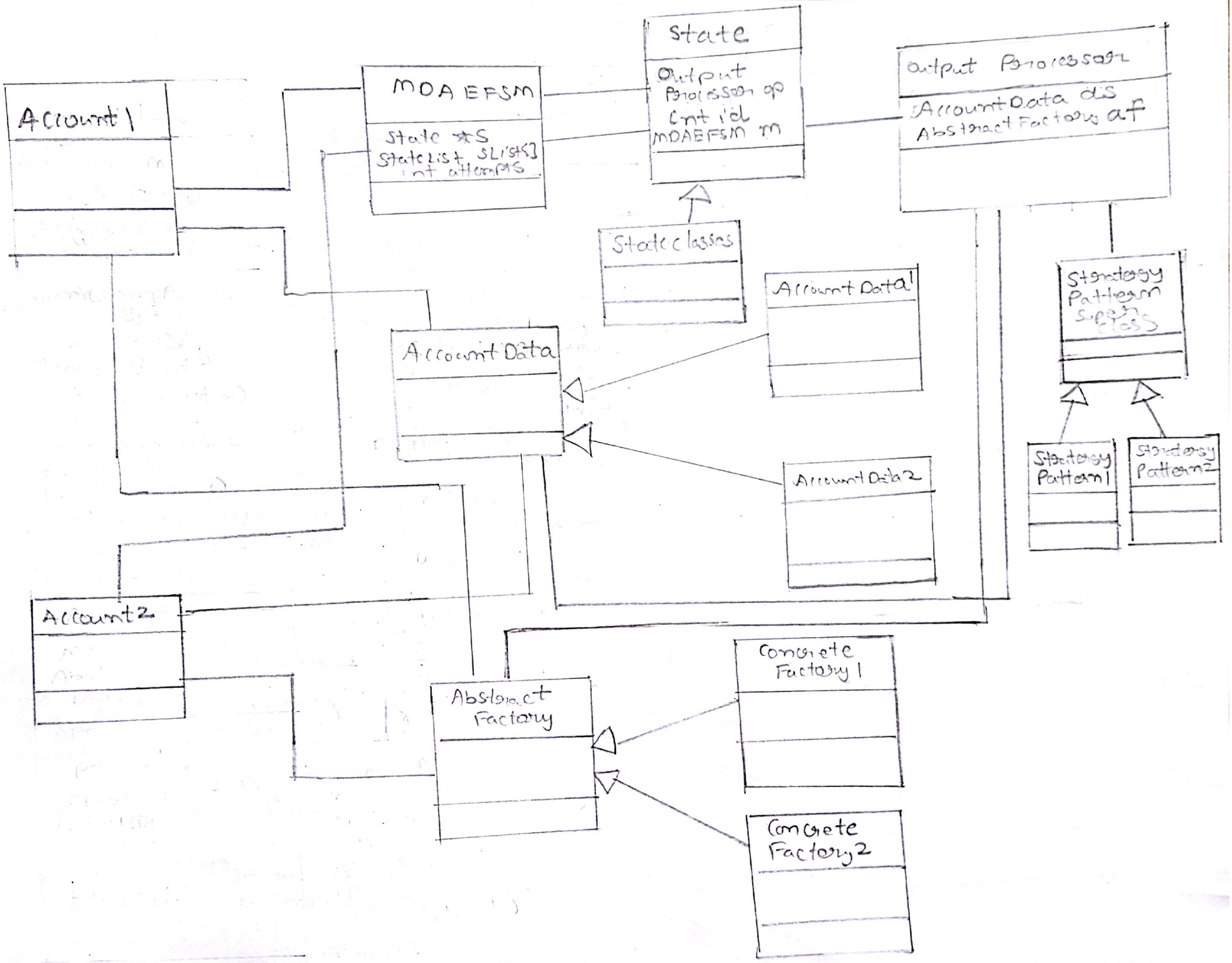
```
OPEN (int p, int y, int a) {
// store p, y and a in temp data store
        ds->temp_p=p;
        ds->temp_y=y;
        ds->temp_a=a;
        m->Open();
}

PIN (int x) {
        if (x==ds->pin)
              m->CorrectPinAboveMin ();
        else m->IncorrectPin(2)
}


DEPOSIT (int d) {
        ds->temp_d=d;
        m->Deposit();
}

WITHDRAW (int w) {
        ds->temp_w=w;
        if (ds->balance>0)
              m->Withdraw();
        else m->NoFunds();
}


BALANCE() {m->Balance();}

LOGIN (int y) {
        if (y==ds->uid)
              m->Login();
        else m->IncorrectLogin();
}

LOGOUT() {m->Logout();}

suspend () {
        m->Suspend();
}

activate () {
        m->Activate();
}

close () {
        m->Close();
}
```

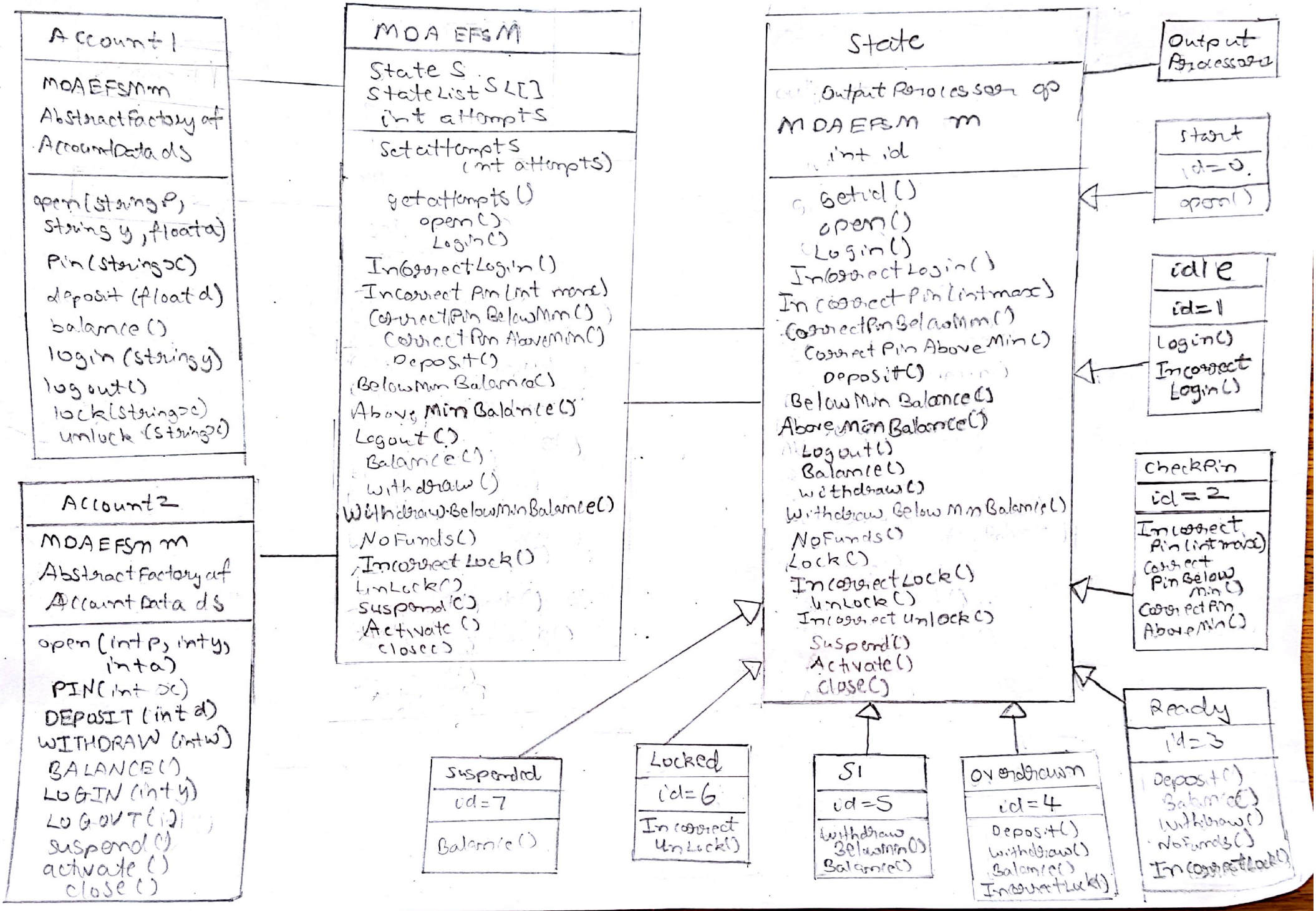
Notice:
m: is a pointer to the MDA-EFSM object
ds: is a pointer to the Data Store object which contains the following data items:

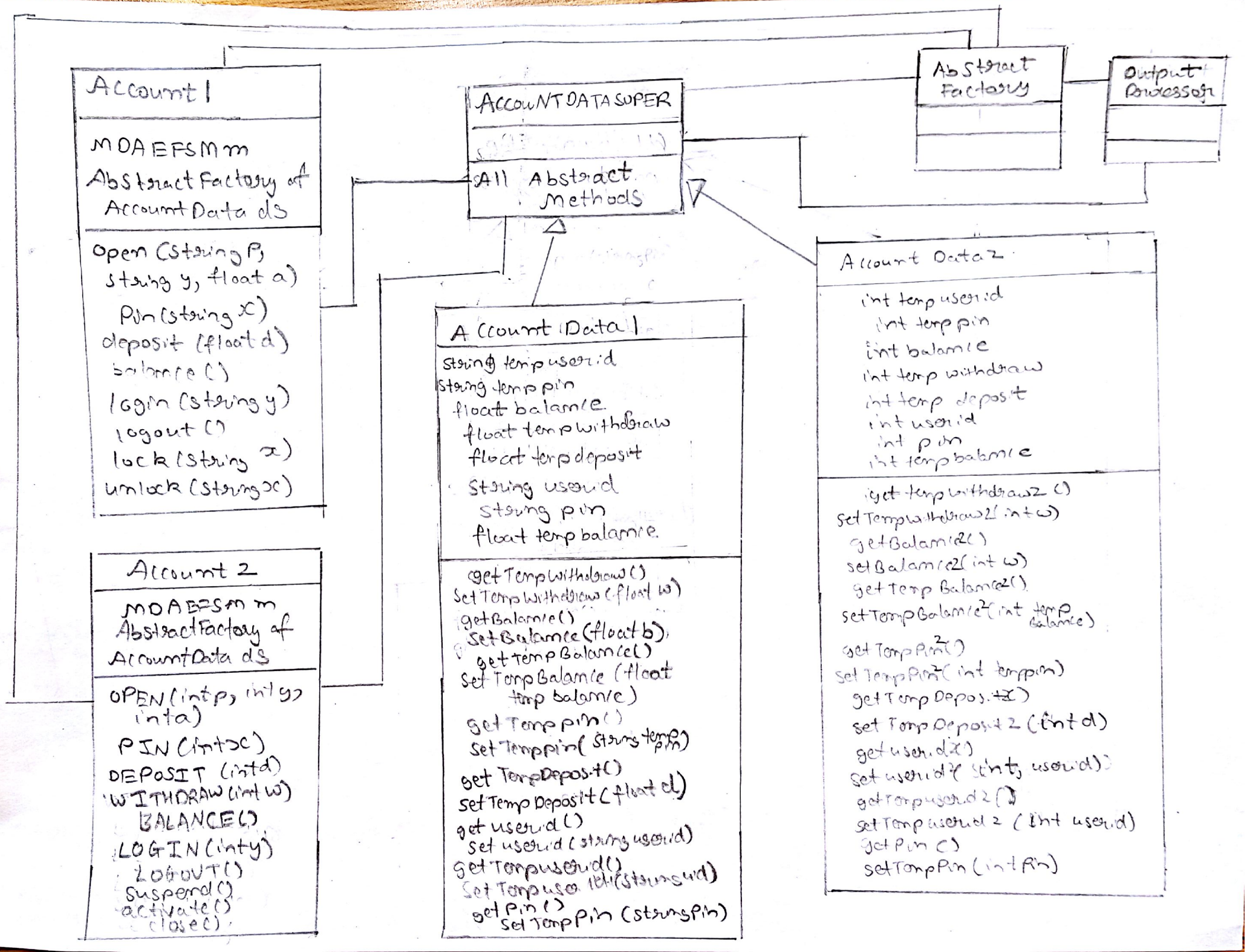
- *balance*: contains the current balance
- *pin*: contains the correct pin #
- *uid:* contains the correct user ID
- *temp_p, temp_y, temp_a, temp_d, temp_w* are used to store values of parameters

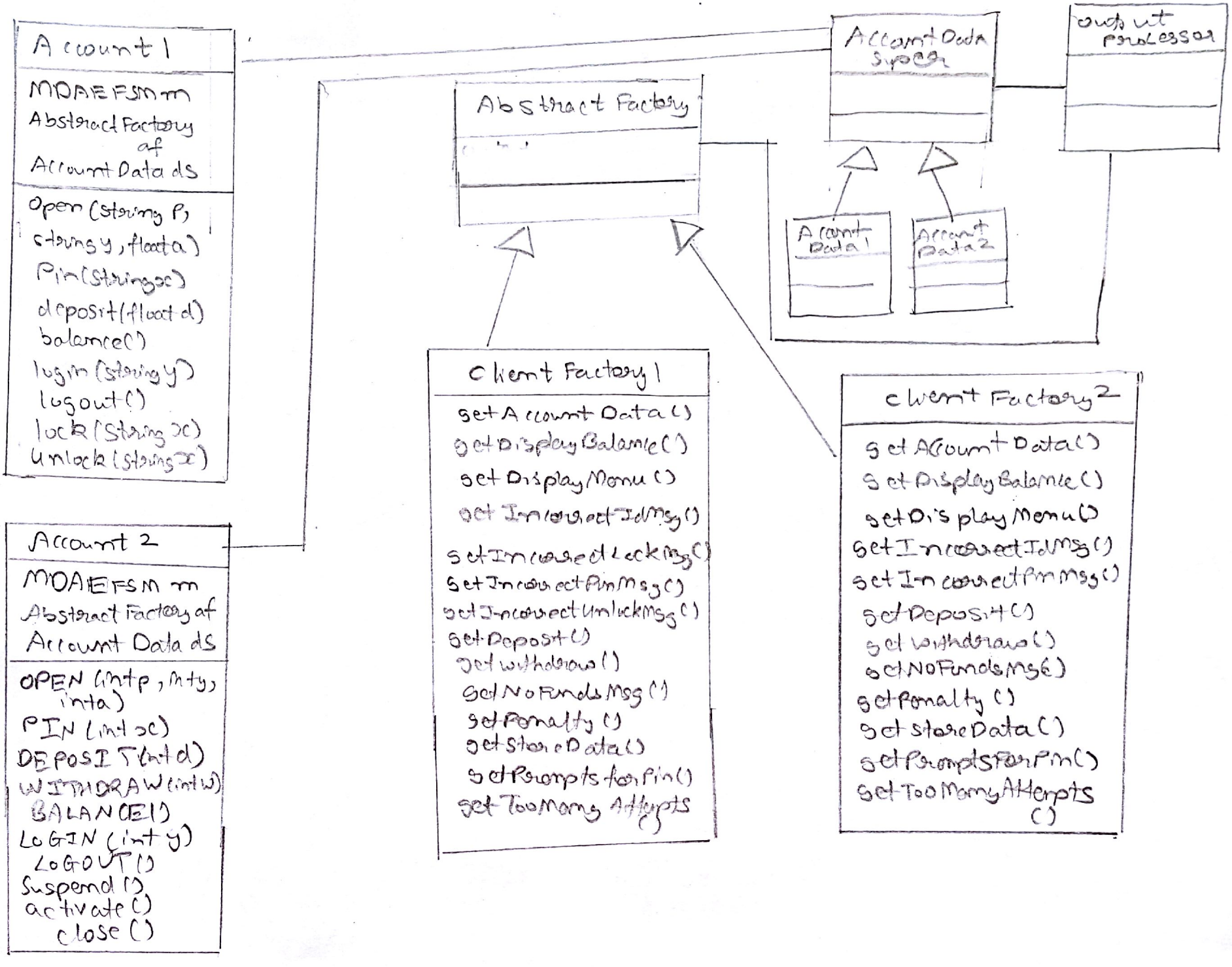

IncorrectPin(max)[attempts<max] /
IncorrectPinMsg; attempts++
Deposit / MakeDeposit
IncorrectLock / IncorrectLockMsg
Balance / DisplayBalance
Balance / DisplayBalance
Suspend
check pin
CorrectPinAboveMin / DisplayMenu
ready
Activate
Suspended
Logout
IncorrectPin(max)[attempts==max] /
IncorrectPinMsg; TooManyAttemptsMsg
Withdraw / MakeWithdraw
Login / attempts=0; PromptForPin
NoFunds / NoFundsMsg
AboveMinBalance
Close
Logout
Lock
IncorrectUnlock / Incorrect
UnlockMsg
idle
S1
Locked
Unlock
Deposit / MakeDeposit
IncorrectLogin / IncorrectIdMsg
CorrectPinBelowMin /
DisplayMenu
WithdrawBelowMinBalance /
Penalty
Open / StoreData
Lock
BelowMinBalance
Logout
overdrawn
Balance / DisplayBalance
Withdraw / NoFundsMsg

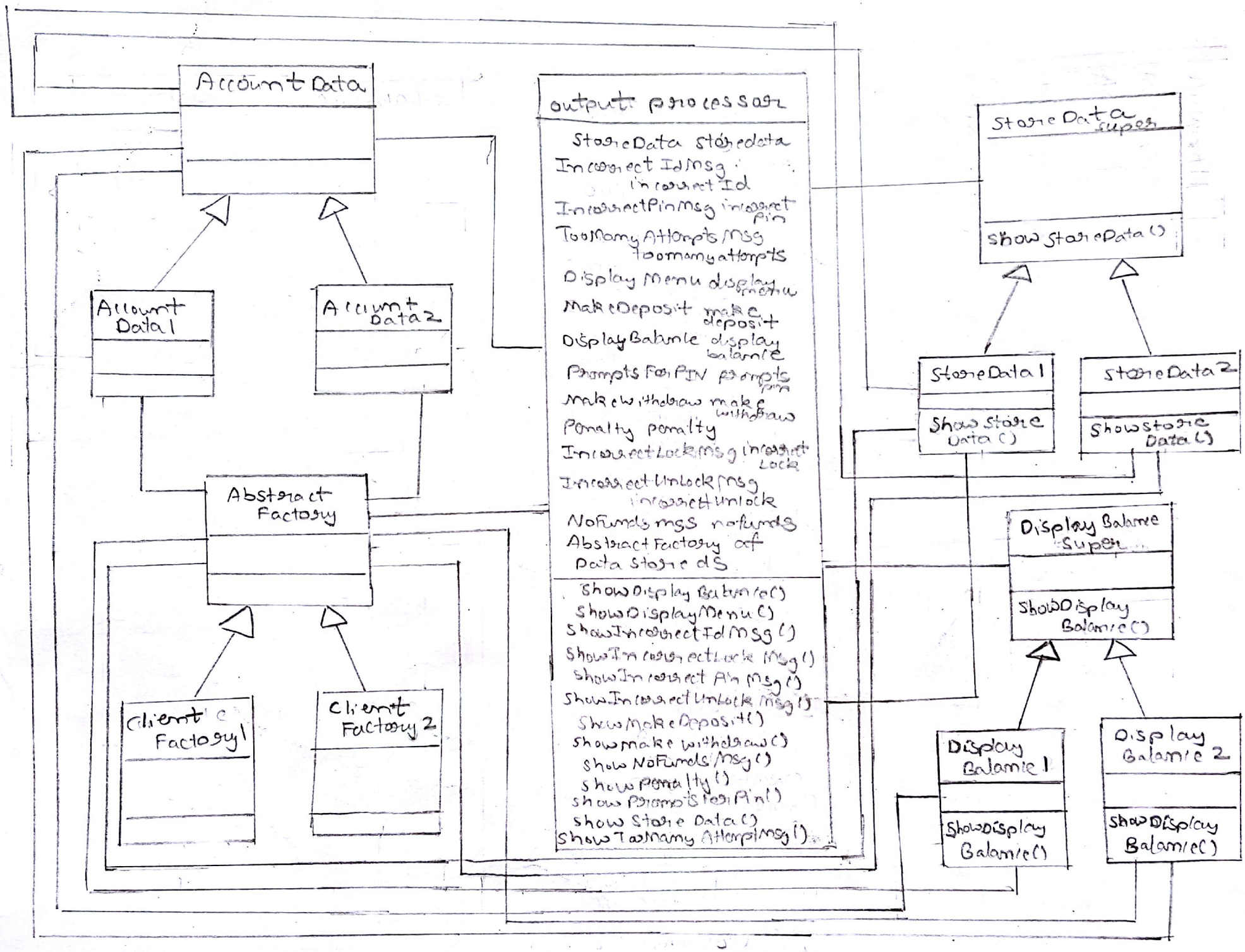
Class Diagram

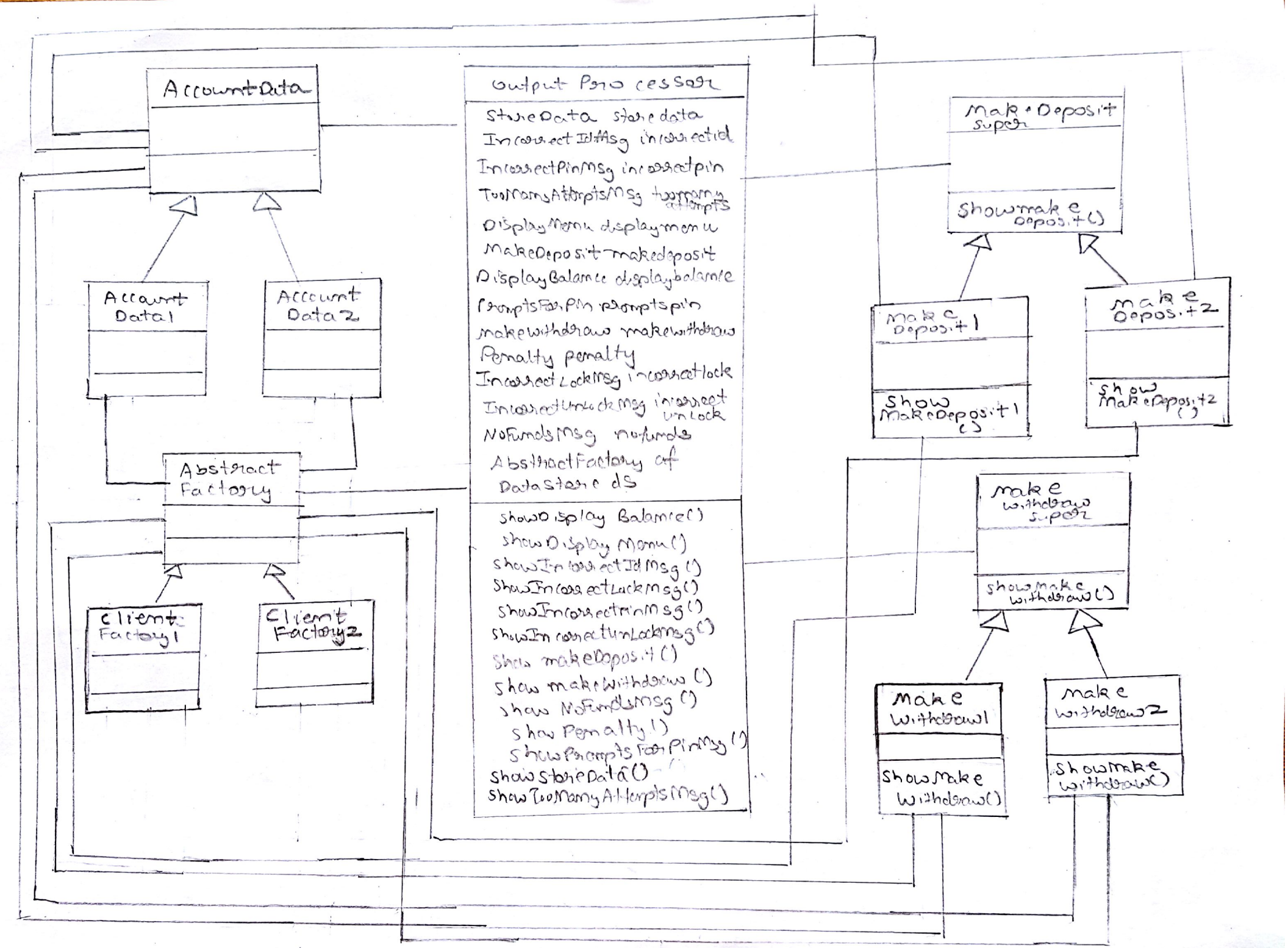


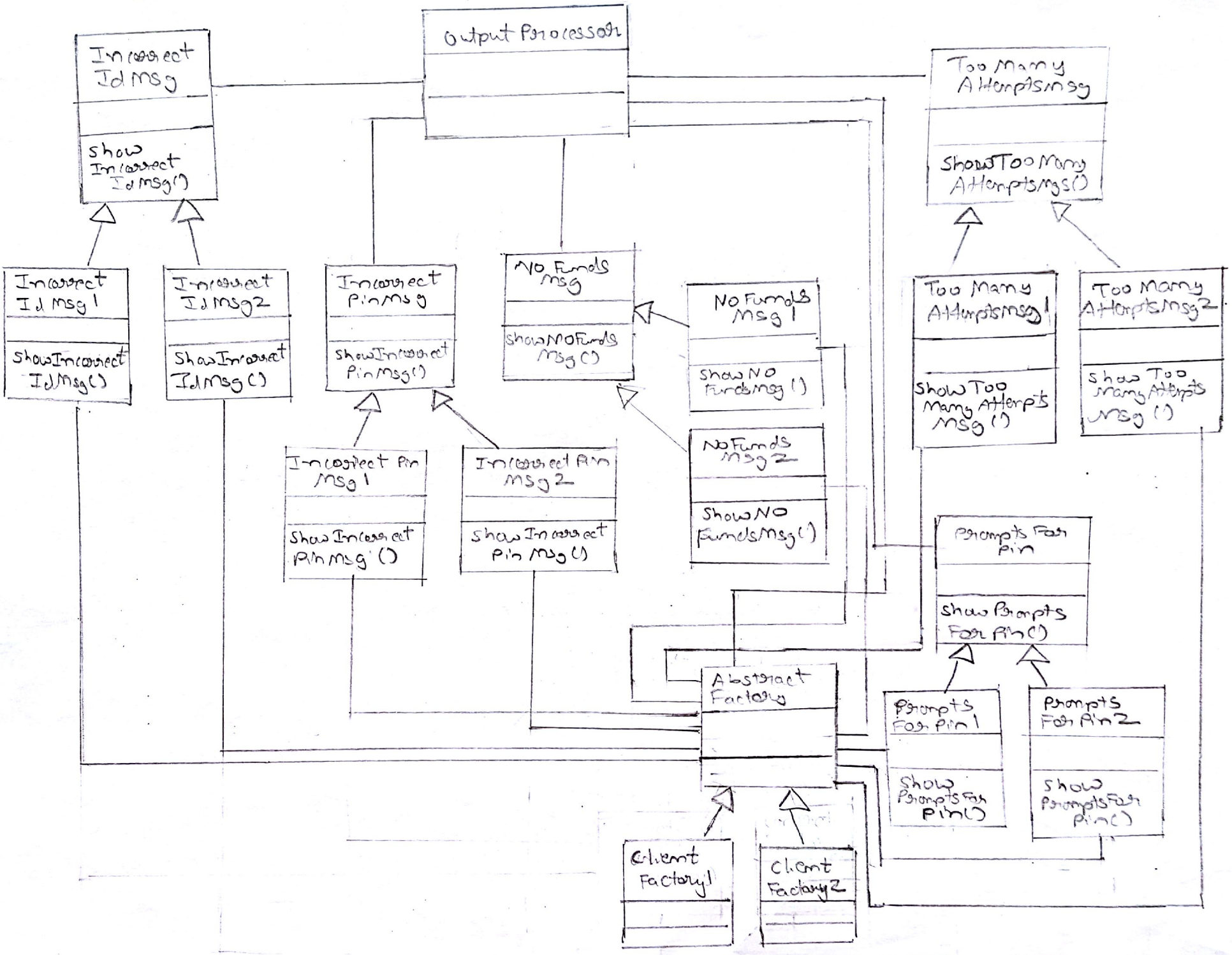


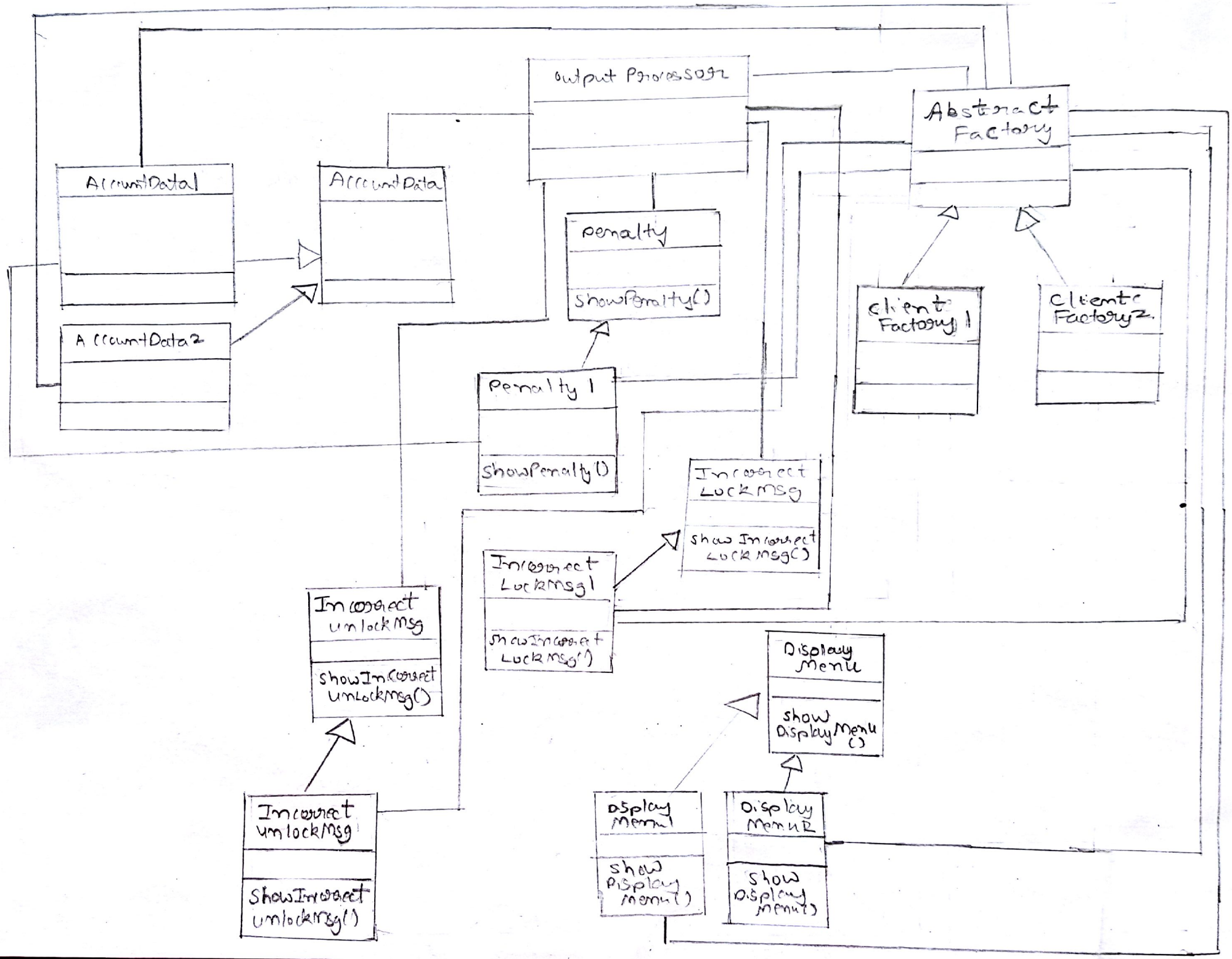








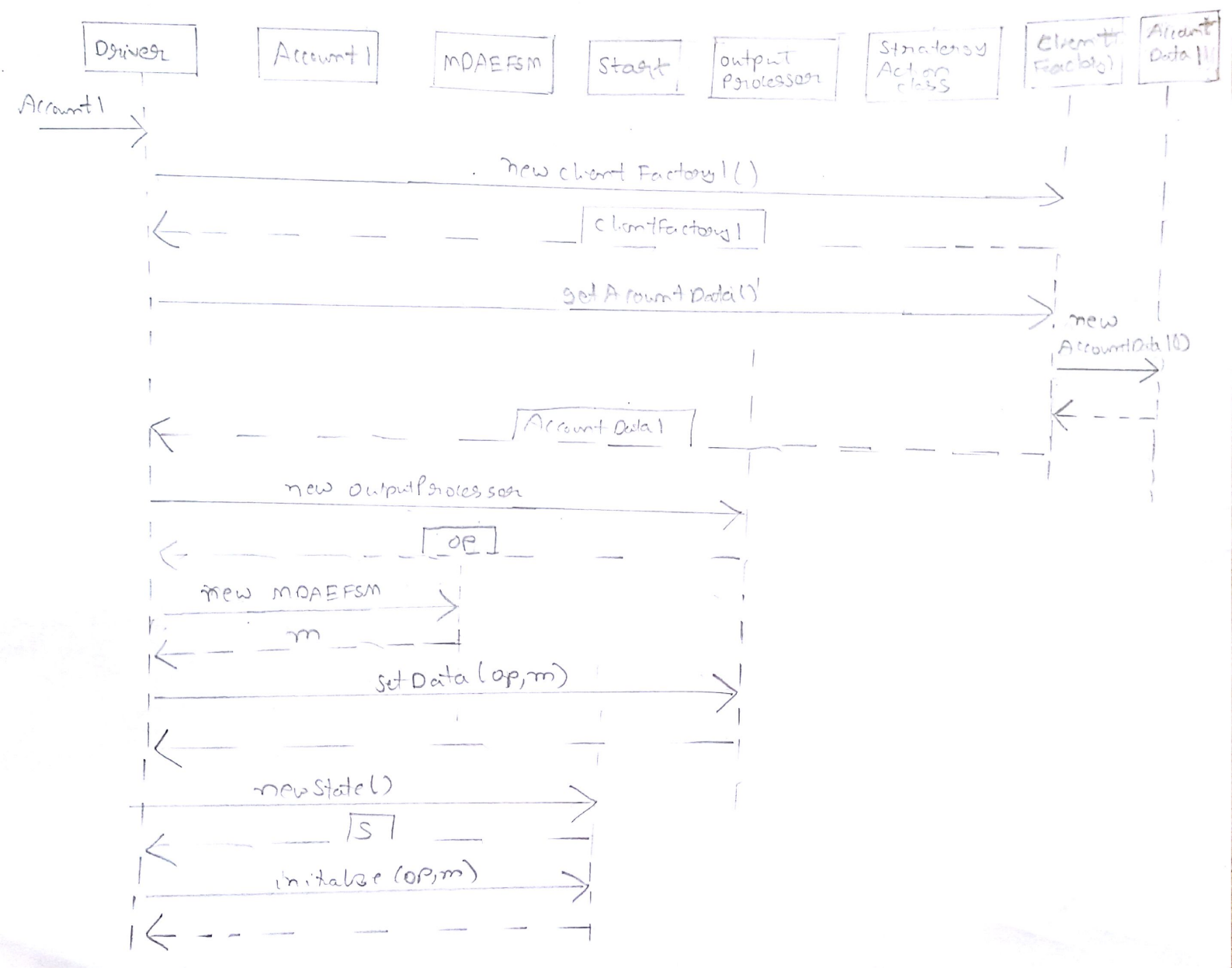


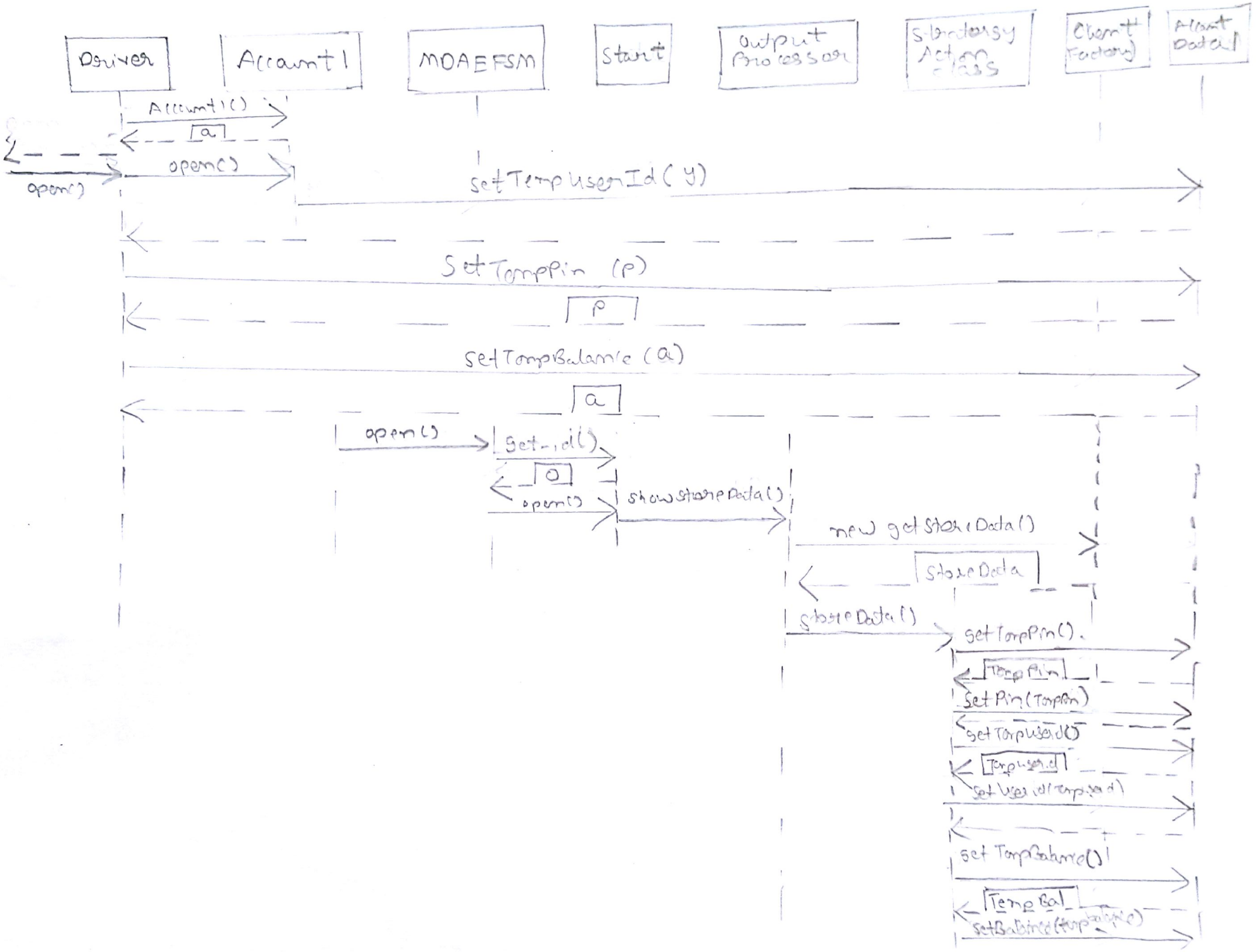


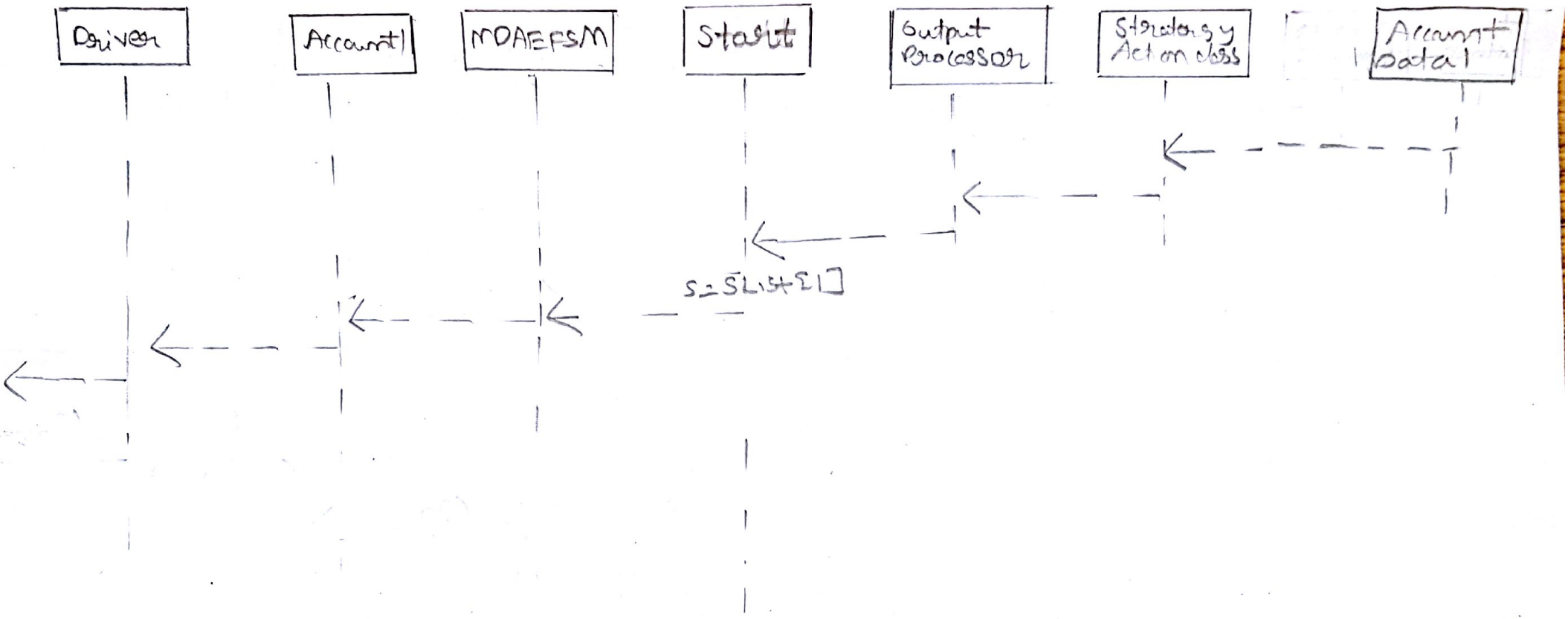
Sequence Diagram

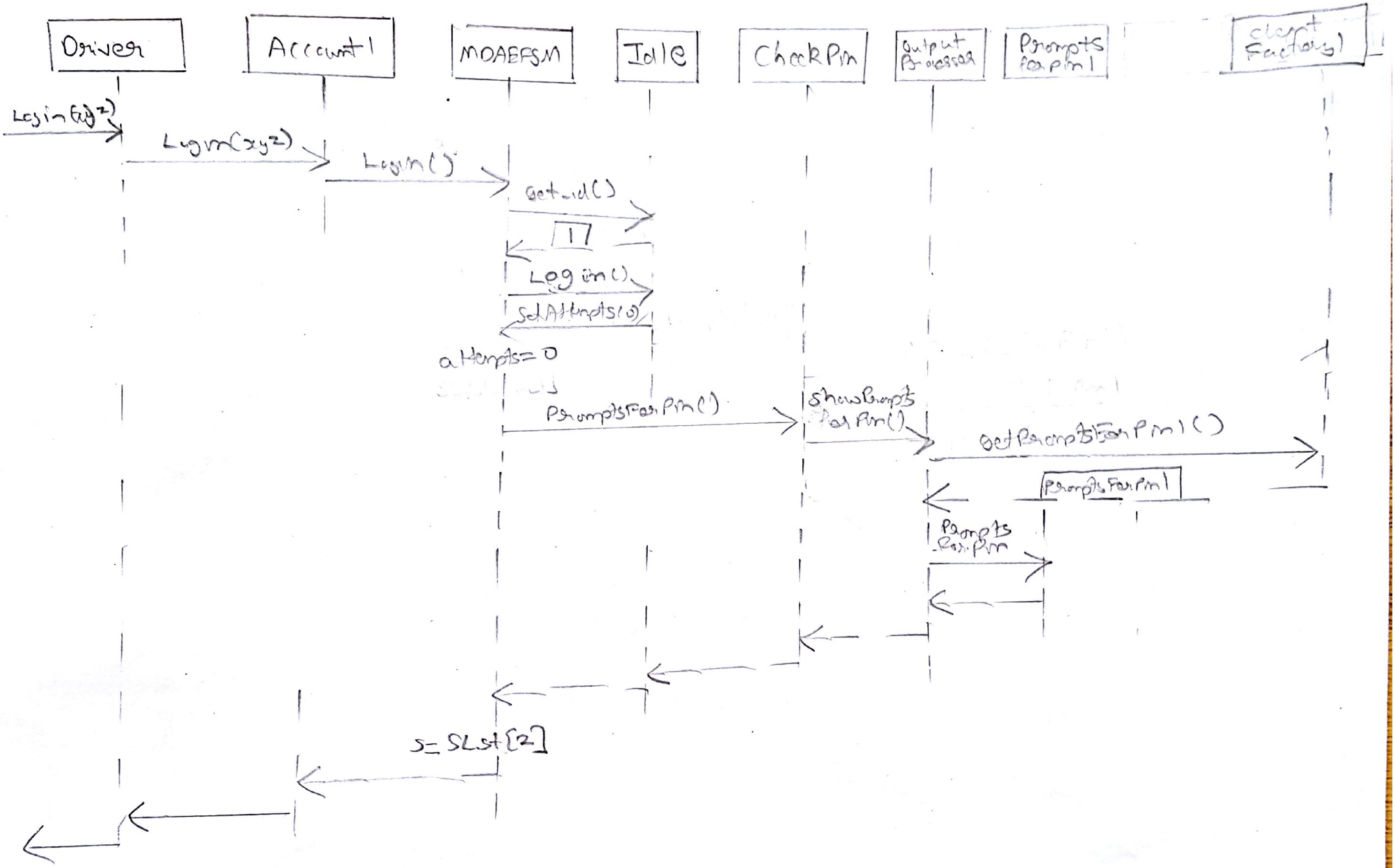
Sequence 1

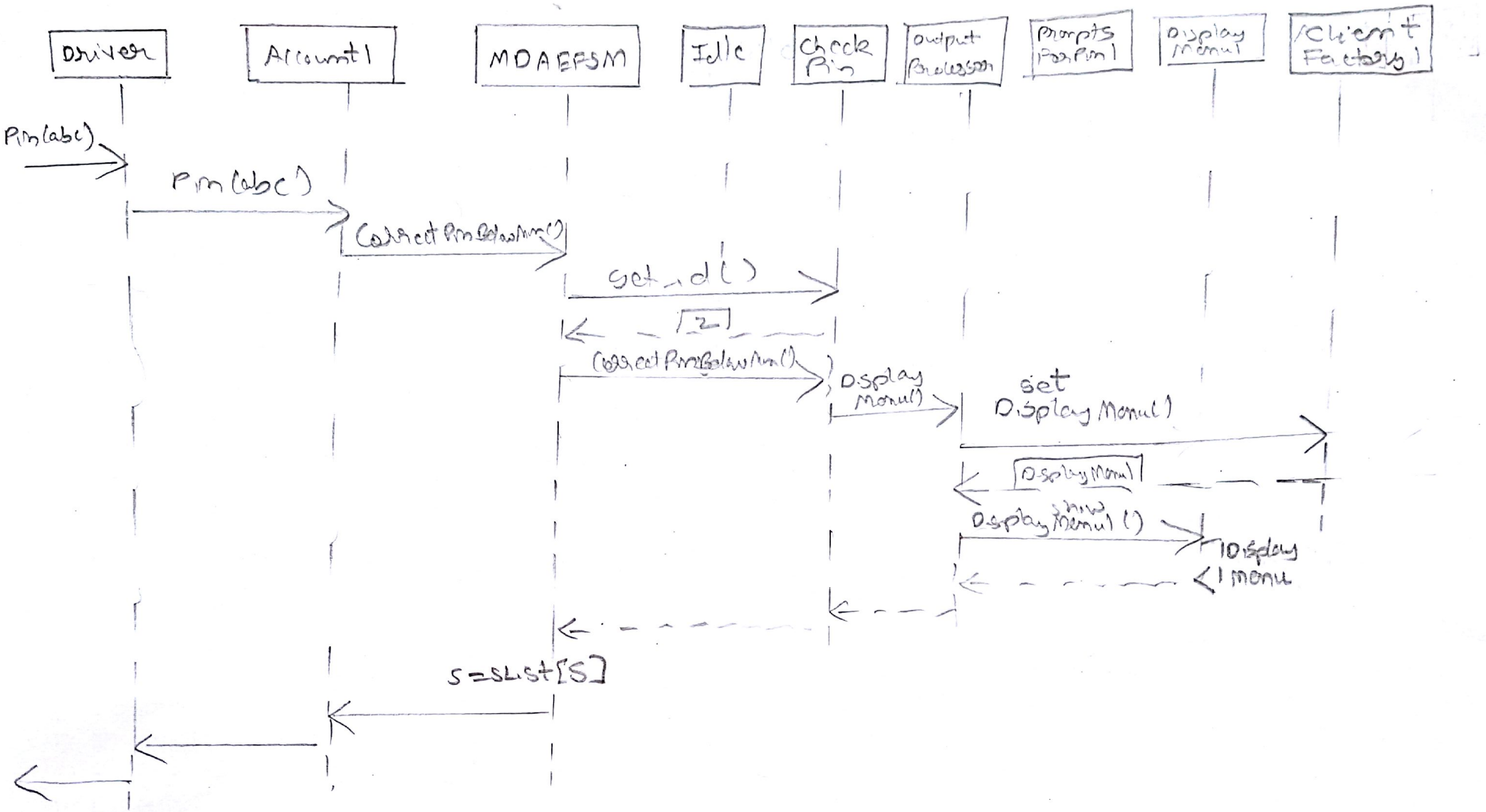
open(abc,xyz,100.5), login(xyz), pin(abc), deposit(400), balance(), logout()

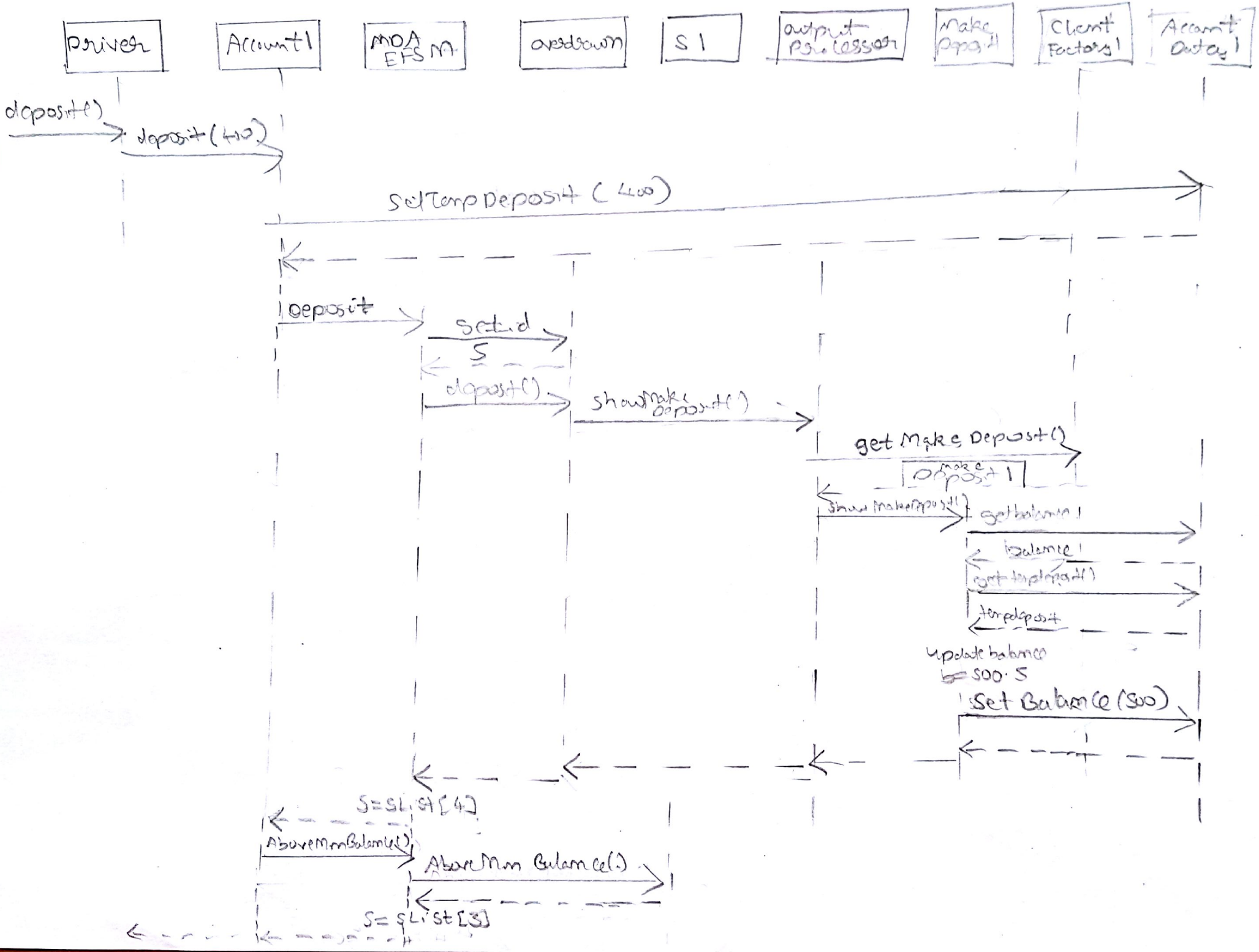


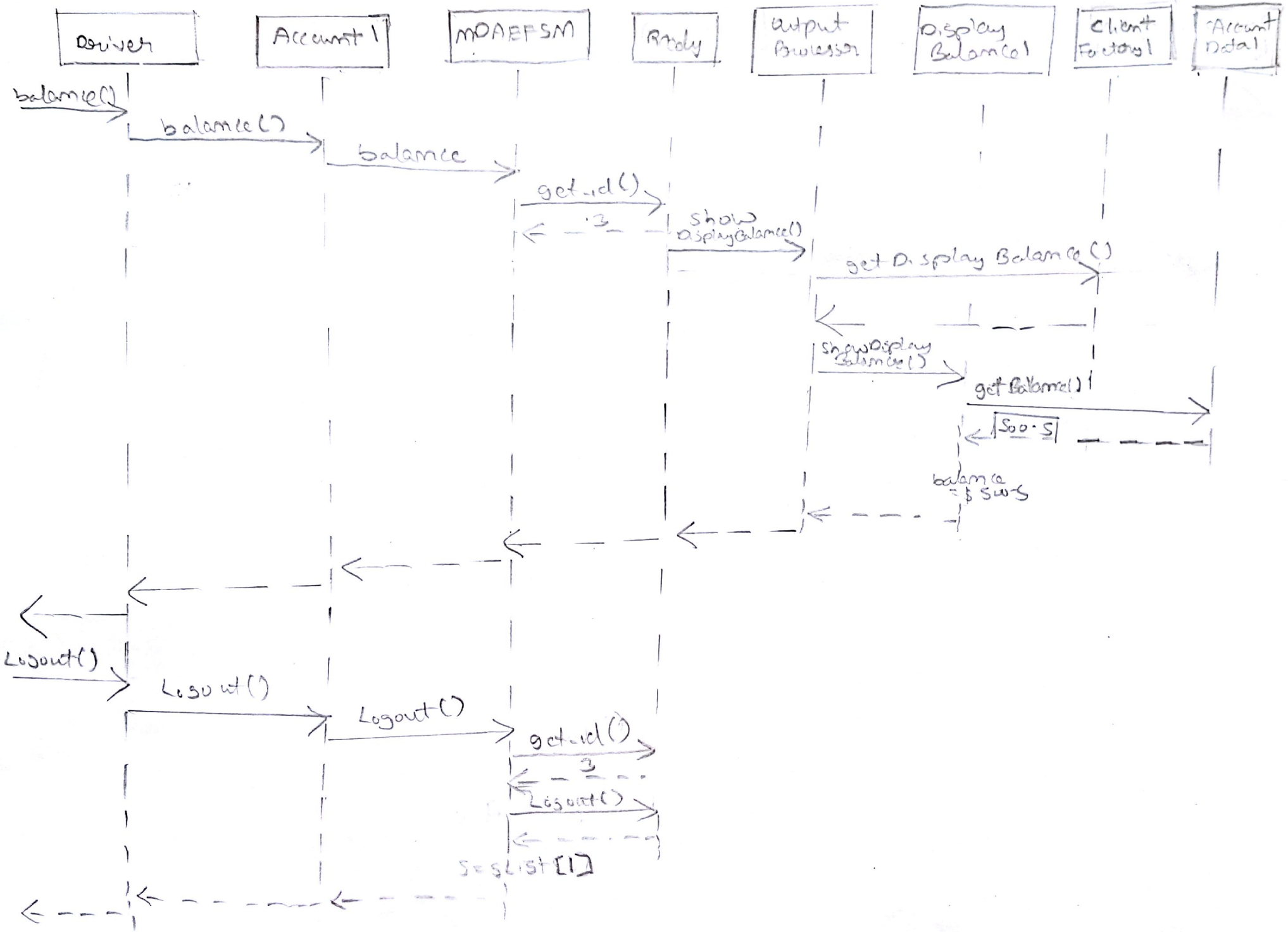






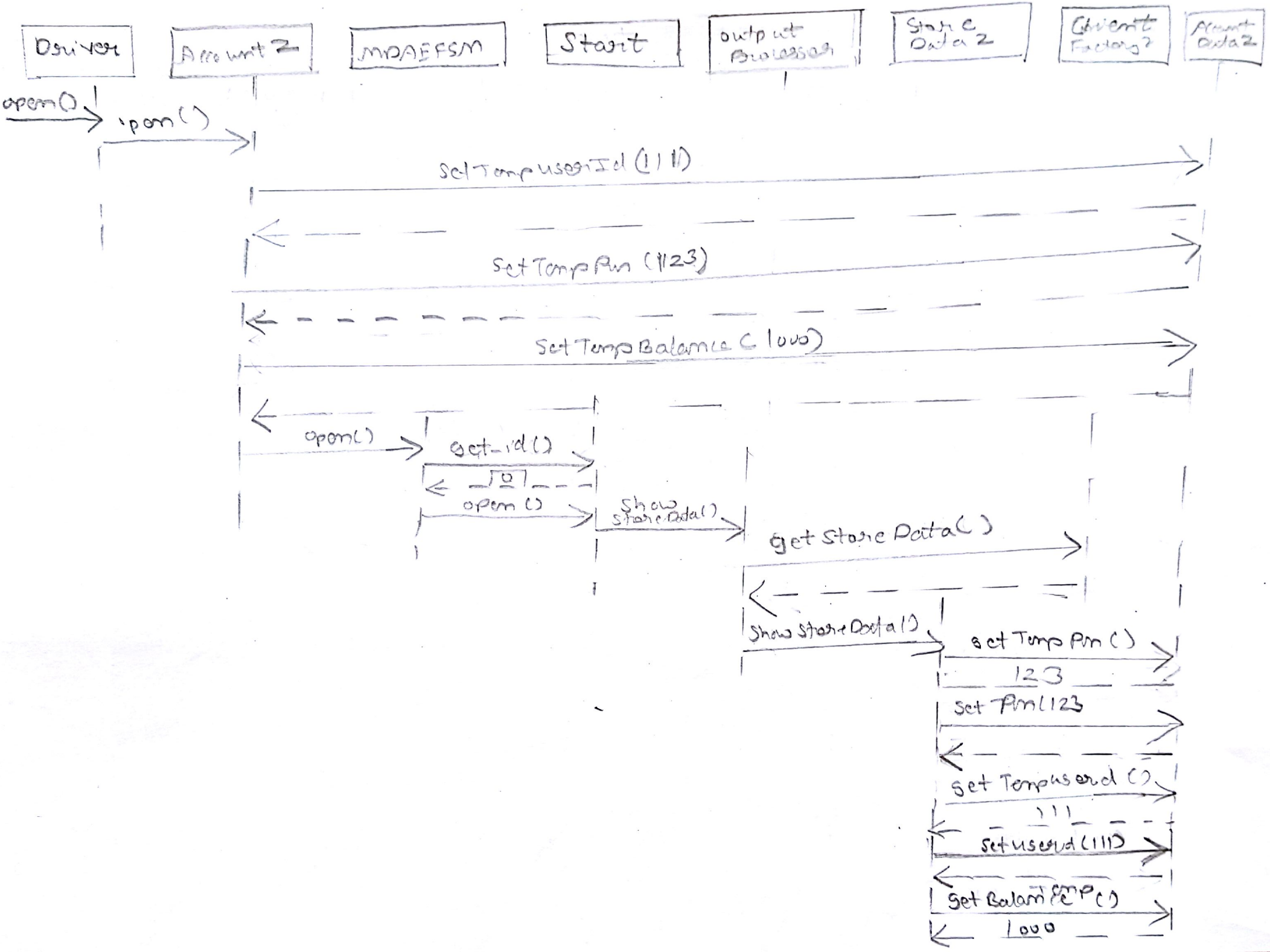


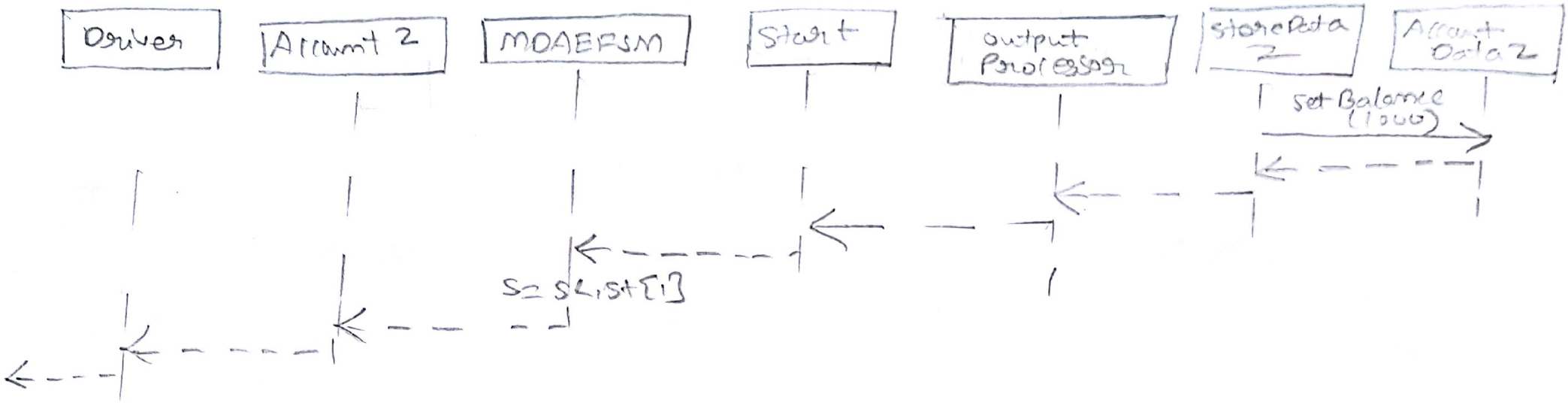


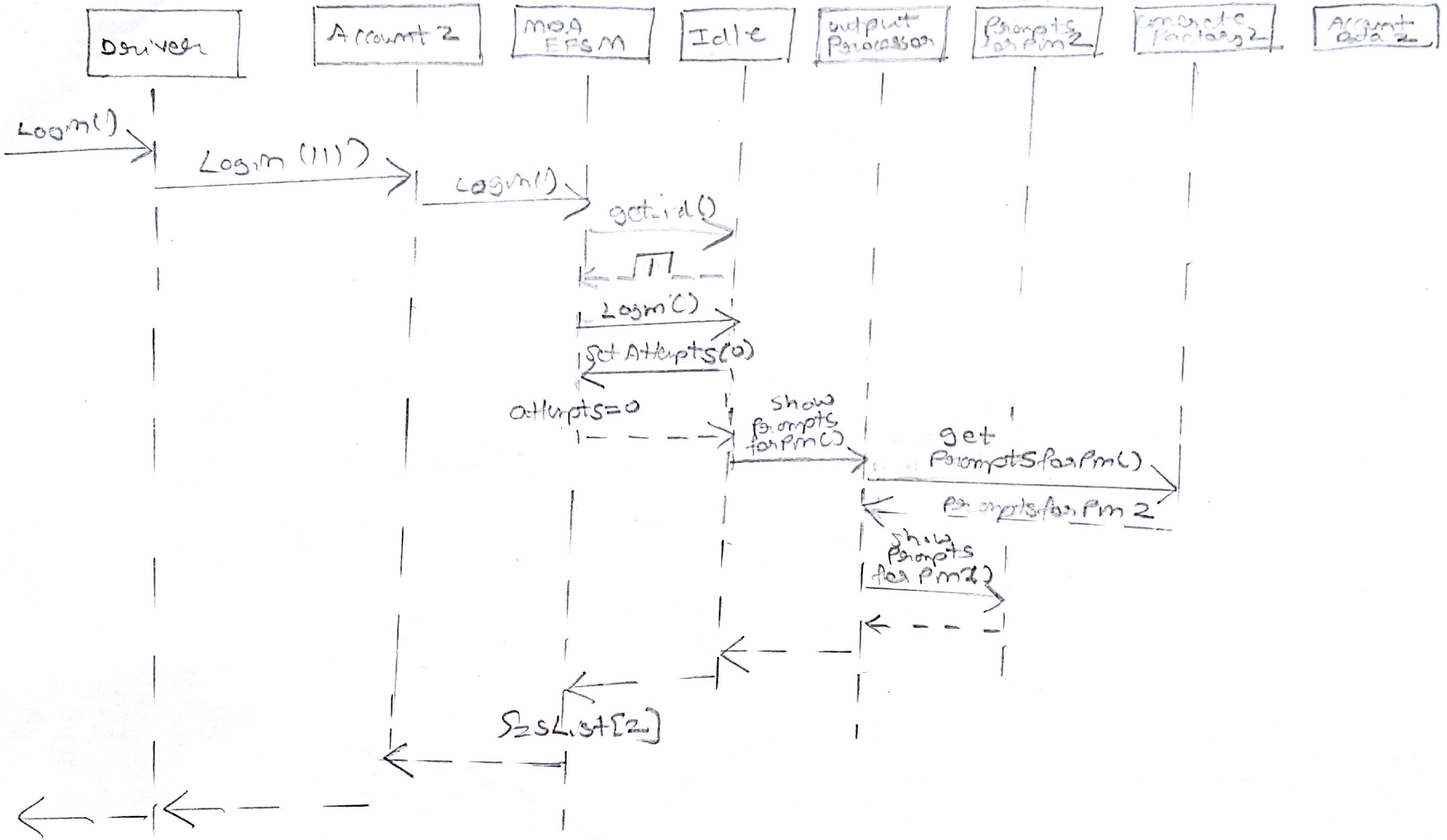


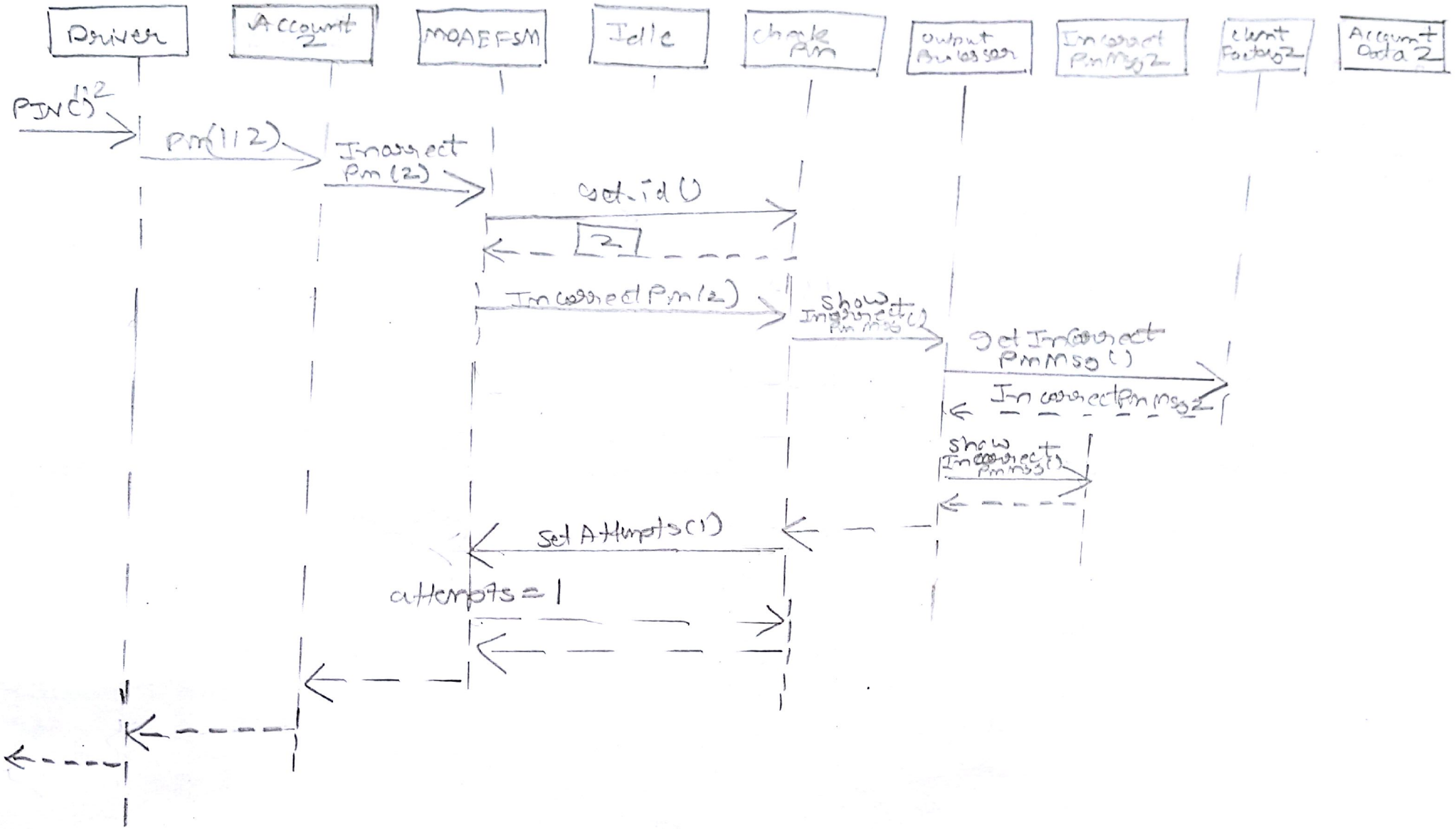
Sequence 2

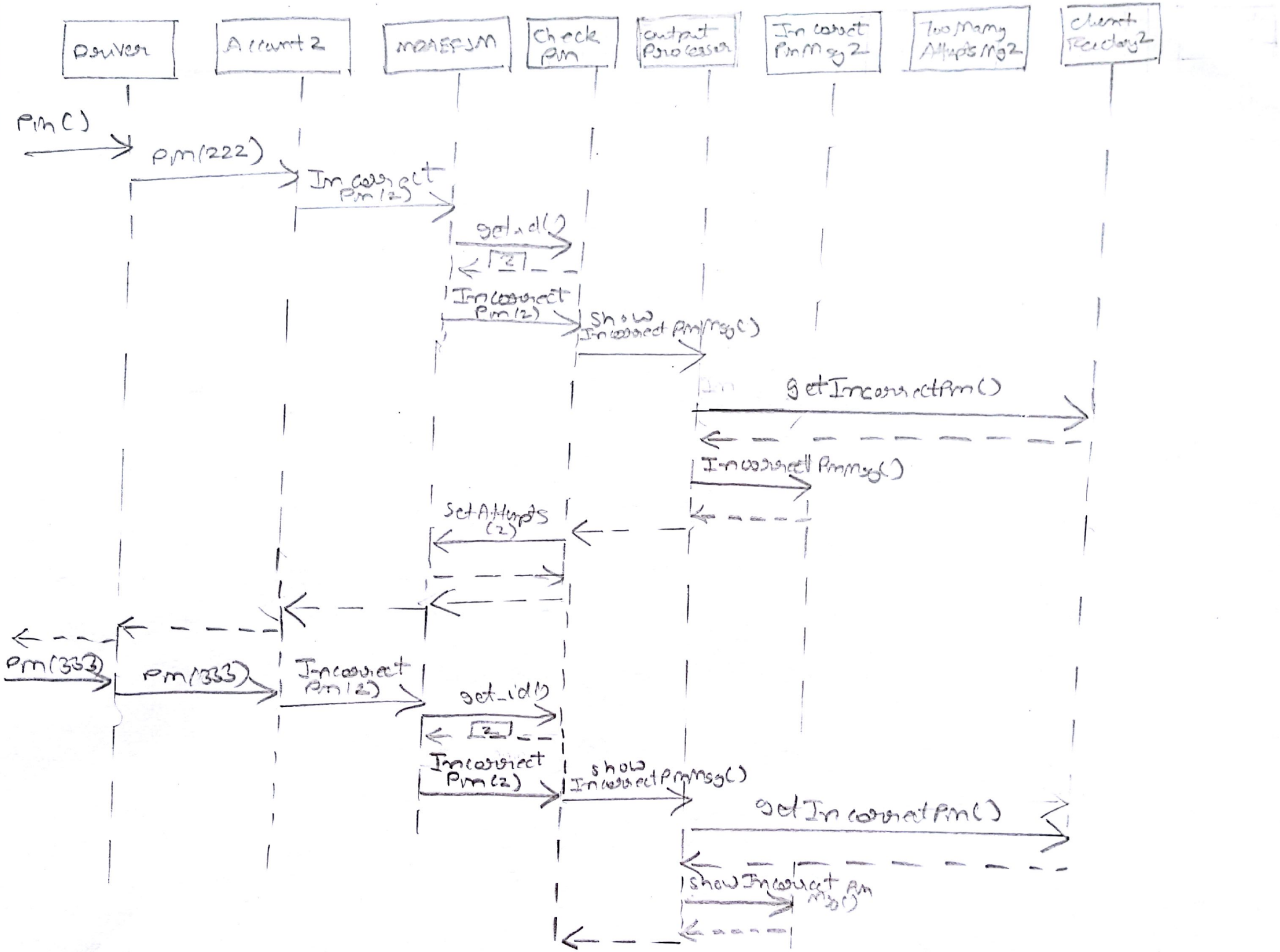
OPEN(123,111,1000), LOGIN(111), PIN(112), PIN(222), PIN(333)

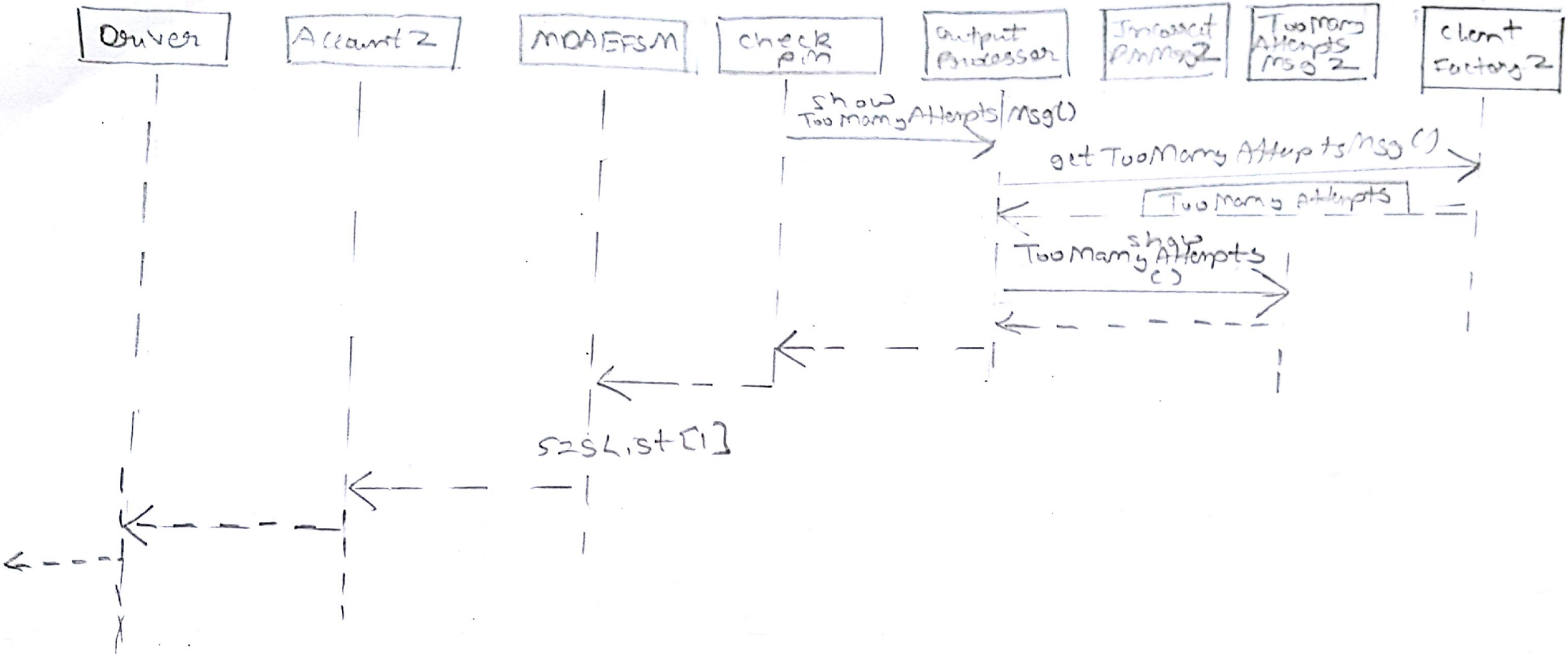












Responsiblities

Classes
InputProcessor

Account1
Responsiblities

```
private AccountDataSuper ds; //used to store the data store object to update the data
private AbstractFactory af; //used to store the abstractfactory object for accesing
the stratergy pattern
private MDAEFSM m; //mdaefsm pointer to access the efsm states and perform efsm
operation

public void open(String p, String y, float a)
//function for opening an account
This function store the pin userid and balance to temperory variable and call the
efsm open operation to open account


public void  pin (String x)
this operation accepts as input the user account pin and validates it with stored pin
in datatore if pin matches user is logged into system to perform operation based on
balance if balance is greater than min balance set in account user will goto to ready
state (correctpinabovemin) else it will goto overdrawn state (correctpinbelowmin)

public void deposit (float d)
this operaton allows user to deposit amount in the account
first amount will be stored to tempdeposit in data store which will be updated to
balance on performing actions  from state class
if current state is overdrawn  and after adding amount if balance is more than min
balance it will goto ready state else it will stay in  overdrawn state


public void  withdraw (float w)
this operaton allows user to withdraw amount from the account
first amount will be stored to tempwithdraw in data store which will be updated to
balance on performing actions from state class
if current state is overdrawn withdraw wont be performed and no funds msg will be
displayed if current state is ready and if withdraw amount is sufficient than min
balance withdraw is performed and after withdraw if balance drops than min state
changes to overdrawn

public void balance()
displays the balance in account

public void login()
it checks if accnt no entered by user is same as account no stored if it is same user
enters into system else it will throw error message saying incorrectlogin

public void lock (String x)
locks the account user id entered by the user

      public void  unlock (String x)
```

unlocks the account user id entered by the user

```
public void logout()
```
logout from the system and state changes to idle

Account2

Responsiblities

```
public class Account2
{
	private AccountDataSuper ds;
	private AbstractFactory af;
	private MDAEFSM m;
	public Account2(MDAEFSM m,AccountDataSuper ds)
	{
		this.m=m;
		this.ds=ds;
	}
```

```
public void OPEN (int p, int y, int a)
```
//function for opening an account
This function store the pin userid and balance to temperory variable and call the efsm open operation to open account

```
public void PIN (int x)
```
this operation accepts as input the user account pin and validates it with stored pin in datatore if pin matches user is logged into system to perform operation based on balance if balance is greater than min balance set in account user will goto to ready state (correctpinabovemin) else it will goto overdrawn state (correctpinbelowmin)

```
public void DEPOSIT (int d)
```
this operaton allows user to deposit amount in the account
first amount will be stored to tempdeposit in data store which will be updated to balance on performing actions from state class
if current state is overdrawn and after adding amount if balance is more than min balance it will goto ready state else it will stay in overdrawn state

```
public void WITHDRAW (int w)
```
this operaton allows user to withdraw amount from the account
first amount will be stored to tempwithdraw in data store which will be updated to balance on performing actions from state class
if current state is overdrawn withdraw wont be performed and no funds msg will be displayed if current state is ready and if withdraw amount is sufficient than min balance withdraw is performed and after withdraw if balance drops than min state changes to overdrawn

```
public void BALANCE()
```
displays the balance in account

```
public void LOGIN (int y)
it checks if accnt no entered by user is same as account no stored if it is same user
enters into system else it will throw error message saying incorrectlogin

public void LOGOUT()
logout from the system and state changes to idle

public void suspend ()
Account state changes to suspend state if function is invoked when accont is in ready
state

public void activate ()
event to  activate account back to ready state from suspended

public void close ()
```
close account from suspend state goes to new state 8

Class mdaefsm
State Slist[] // stores the list of states available in efsm initialize all the states and store in list to be assigned on state changes
State s //store the pointer to state currently active used to invoke any state operation
Int attempts//store the no of times user has tried to login into system this value increments whenever user enters a incorrect pin on pin operation if attempts value reaches the maximum attempts allowed by user to login it wil change the state to idle based on account
State s=slist[0]; //initialize the state with start

public void Open() // opens an account and store the basic account information like userid pin and balance
after data is stored perform the state transitions from start to idle

public void Login()
perform login operation and changes the state to checkpin

public void IncorrectLogin()
if userid does not match with registered userid incorrectlogin is called there wont be any state change

public void IncorectPin(int max)
checks the id if id is 2 and attempts less than max attempts it stays in checkpin state increases the attempts else if attempts is greater than max it moves to idle state

public void CorrectPinBelowMin()
checks if the id is 2 it changes state to overdrawn state is invoked when balance is above min in checkpin state

public void CorrectPinAboveMin()
checks if the id is 2 account state changes to ready state is invoked when balance is above min in checkpin state

public void BelowMinBalance()
checks if in s1 state it changes state to overdrawn

public void Deposit()
if state is in 3 or 5 ready or overdrawn it allows to do deposit in account calls state deposit function

public void AboveMinBalance()
if state is in s1 4 then it goes to ready state

public void Logout()
if current state is ready or overdrawn move to idle state

public void Balance()
check the balance in ready overdrawn or suspended state using state pointer to s.balance()

public void Withdraw()
if state is in ready or overdrawn call withdraw operation if in ready state it changes state to s1

public void WithdrawBelowMinBalance()
in s1 state if balance is below minimum then make a withdraw with penalty and changes state to overdrawn

public void NoFunds()
in ready state if balance is below minimum then display no enough funds

public void IncorrectLock()
in ready state if lock is incorrect display message incorrect lock

public void Unlock()
in locked state if lock pin is correct unlock account go to s1 state

public void IncorrectUnlock()

in locked state if lock pin is incorrect display incorrect unlock

public void Suspend()
if in ready state for account 2 goto suspend state

public void Activate()
if in suspend state for account 2 activate account again and goto ready state

public void Close()
close an account

public void setAttempts(**int** attempts)
for setting the attempts no of times user tries to login in check pin state
public int getAttempts()

for getting the no of times user tried to login in check pin state for performing transition

```
State class
//abstract class having all function of mdaefsm where actions has to be perfomed
Int id //store the current id which points to state of Account which it is currently
in

Get_id //returns the id to be accessed by mdaefsm to keep account of current state
and perform state transitions

Class Start
Id=0 //id for start state
Open()
Calls output procedure storedata to store data

Class idle
Id=1 //id for idle state
Login()
// perform the login operation and calls action prompts for pin to ask pin from user

IncorrectLogin()
//if user id does not match calls action incorrectlogin to display incorrect login
message to user

Class Checkpin
Id=2 //id for checkpin state

IncorrectPin (int max)

CorrectPinBelowMin
Calls outputprocesor show displaymenu action that Displays menu to the user

CorrectPinAboveMin
Calls outputprocesor show displaymenu action that Displays menu to the user



public class Ready

id=3 //id for ready state

      public void Deposit()

// call action showmakedeposit to deposit the amount


      public void Balance()
//call action showbalance to display the balance to user
```

```
Withdraw()
//call action makewithdraw to withdraw amount from account


IncorrectLock()

//call action showincorrectlock to display incorrectlock message

	public void NoFunds()

//call outputprocessor function nofunds that displays that account has no funds to
perform operation




Class s1
	public void WithdrawBelowMinBalance()
 call action penalty that charges penalty and decreases to balance


public class Overdrawn

id=5 //id for overdrawn state

	public void Deposit()

// call action showmakedeposit to deposit the amount

	public void Balance()

//call action showbalance to display the balance to user


Withdraw()
//call action makewithdraw to withdraw amount from account

	public void IncorrectLock()
//call action showincorrectlock to display incorrectlock message


public class Locked

id=6 //
	public void IncorrectUnlock()
call action incorrectunlock of outputprocessor

public class Suspended
id=7 //store the state id for suspended

	public void Balance()
//call action showbalance to display the balance to user

Public class stop
```

```
//id=8 id for storing stop state
```

Stratergy Pattern
Class Output Processor
Responsiblities

```
	private AccountDataSuper ds; //pointer to accountdata
	private AbstractFactory af;//pointer to abstractfactory
	private DisplayBalanceSuper displaybalance; //pointer to action displaybalance
	private DisplayMenuSuper displaymenu; //pointer to action displaymenu
	private IncorrectIdMsgSuper incorrectid; //pointer to action incorrectid
	private IncorrectPinMsgSuper incorrectpin; //pointer to action incorrectpin
	private IncorrectLockMsgSuper incorrectlock; //pointer to action incorrectlock
	private IncorrectUnlockMsgSuper incorrectunlock; //pointer to action
incorrectunlock
	private MakeDepositSuper makedeposit; //pointer to action makedeposit
	private MakeWithdrawSuper makewithdraw; //pointer to action makewithdraw
	private NoFundsMsgSuper nofunds; //pointer to action nofunds
	private PenaltySuper penalty; //pointer to action penalty
	private PromptsForPinSuper promptsforpin; //pointer to action promptsforpin
	private StoreDataSuper storedata; //pointer to action storedata
	private TooManyAttemptsMsgSuper toomanyattempts; //pointer to action
toomanyattemptsmsg

	public void showDisplaybalance()
//based on abstractfactorypointer call getdisplaybalance of clientfactory and then
invoke respective action to displaybalance of account

	public void showdisplaymenu()

//based on abstractfactorypointer point to clientfactory call getdisplaymenu to
create object of displaymenuaction from clientfactory and then invoke respective
action to displaymenu to user

	public void showIncorrectid()

//based on abstractfactorypointer call getincorrectid of clientfactory and then
invoke respective action to display incorrectid message to user

	public void showIncorrectpin()

//based on abstractfactorypointer call getIncorrectpin of clientfactory and then
invoke respective action to display incorrectpin message to user

	public void showIncorrectlock()

//based on abstractfactorypointer call getIncorrectlock of clientfactory and then
invoke respective action to display incorrectlock message to user
```

```
	public void showIncorrectunlock()

//based on abstractfactorypointer call getIncorrectunlock of clientfactory and then
invoke respective action to display incorrectunlock message to user


	public void showMakedeposit()

//based on abstractfactorypointer call getdeposit of clientfactory and then invoke
respective action to perform deposit on account

	public void showMakewithdraw()

//based on abstractfactorypointer call getwithdraw of clientfactory and then invoke
respective action to perform deposit on account

	public void showNofunds()

//based on abstractfactorypointer call getdisplaybalance of clientfactory and then
invoke respective action to displaybalance of account

	public void showPenalty()

//based on abstractfactorypointer call penalty of clientfactory and then invoke
respective action  showpenalty of account to perform penalty

	public void showPromptsforpin()

//based on abstractfactorypointer call getPromptsforpin of clientfactory and then
invoke respective action to show Promptsforpin message to user

	public void showStoredata()

//based on abstractfactorypointer call getstoredata of clientfactory and then invoke
respective action to store the data into accountdata

	public void showToomanyattempts()

//based on abstractfactorypointer call getToomanyattempts of clientfactory and then
invoke respective action to display showToomanyattempts of account

	public void setData(AbstractFactory af,AccountDataSuper ds)
```
//set abstractfactory and accountdata object respectively for account from driver passed

Class AbstractFactory
//all abstract methods for client factory 1 and 2
Class ClientFactory1

```
	public AccountDataSuper getAccountData()
// create object for accountdata1
	public DisplayBalanceSuper getDisplayBalance()
// create object for displaybalance
```

```
	public DisplayMenuSuper getDisplayMenu()
// create object for action displaymenu1

	public IncorrectIdMsgSuper getIncorrectIdMsg()
// create object for incorrectidmsg11

	public IncorrectLockMsgSuper getIncorrectLockMsg()
// create object for incorrectlockmsg1

	public IncorrectPinMsgSuper getIncorrectPinMsg()
// create object for incorrectpinmsg1

	public IncorrectUnlockMsgSuper getIncorrectUnlockMsg()
// create object for incorrectunlockmsg1

	public MakeDepositSuper getDeposit()
// create object for makedeposit1

	public MakeWithdrawSuper getWithdraw()
// create object for makewithdraw1

	public NoFundsMsgSuper getNoFundsMsg()
// create object for nofundsmsg1

	public PenaltySuper getPenalty()
// create object for penalty1

	public StoreDataSuper getStoreData()
// create object for storedata1

	public PromptsForPinSuper getPromptForPin()
// create object for promptsforpin1

	public TooManyAttemptsMsgSuper getTooManyAttempts()
// create object for toomanyattempts1
```

Class ClientFactory2

```
	public AccountDataSuper getAccountData()
// create object for accountdata2
	public DisplayBalanceSuper getDisplayBalance()
// create object for displaybalance2


	public DisplayMenuSuper getDisplayMenu()
// create object for action displaymenu2

	public IncorrectIdMsgSuper getIncorrectIdMsg()
// create object for incorrectidmsg2

	public IncorrectPinMsgSuper getIncorrectPinMsg()
// create object for incorrectpinmsg2
```

```
	public MakeDepositSuper getDeposit()
// create object for action makedeposit2

	public MakeWithdrawSuper getWithdraw()
// create object for makewithdraw2

	public NoFundsMsgSuper getNoFundsMsg()
// create object for nofundsmsg2

	public StoreDataSuper getStoreData()
// create object for storedata2

	public PromptsForPinSuper getPromptForPin()
// create object for promptsforpin2

	public TooManyAttemptsMsgSuper getTooManyAttempts()
// create object for toomanyattempts2
```

Public class AccountData1

```
float tempbalance; //store the tempbalance
	String tempuserid; //store tempuserid
	String pin;//store pin regeistered for user
	float balance; //store balance in account
	float tempdeposit; //store temperory deposit
	float tempwithdraw; //store temperory withdraw later to be updated to balance
	String temppin; //store temppin later to be set in permananent pin in actions
	String userid; //store user id

	public float getTempbalance() //get the tempbalance stored

	public void setTempbalance(float tempbalance) //set the temperory balance
string


	public String getTempuserid() //get temperory userid

	public void setTempuserid(String tempuserid) //set temperory userid

	public String getPin() //get the pin in account
	public void setPin(String pin) //set the pin in account

	public float getBalance() //get the balance of account

	public void setBalance(float balance) //set the balance in account

	public float getTempdeposit()  //get the temp deposit entered by user in
inputprocessor

	public void setTempdeposit(float tempdeposit) //set the temp deposit invoked
in inputprocessor of account
	public float getTempwithdraw() //get temp withdraw amount
	public void setTempwithdraw(float tempwithdraw) //set tempwithdraw amount
called from inputprocessor
```

```
	public String getTemppin() //get temperory pin
	public void setTemppin(String temppin) //set temperory pin
	public String getUserid() //get userid
	public void setUserid(String userid) //set userid invoked from action
```

Public class AccountData2

Public class AccountData1

```
int tempbalance; //store the tempbalance
	int tempuserid; //store tempuserid
	int pin;//store pin regeistered for user
	int balance; //store balance in account
	int tempdeposit; //store temperory deposit
	int tempwithdraw; //store temperory withdraw later to be updated to balance
	int  temppin; //store temppin later to be set in permananent pin in actions
	int userid; //store user id

	public int getTempbalance2() //get the tempbalance stored

	public void setTempbalance2(int tempbalance) //set the temperory balance
string


	public String getTempuserid2() //get temperory userid

	public void setTempuserid2(int tempuserid) //set temperory userid

	public int getPin2() //get the pin in account
	public void setPin2(int pin) //set the pin in account

	public int getBalance2() //get the balance of account

	public void setBalance2(int balance) //set the balance in account

	public int getTempdeposit2()  //get the temp deposit entered by user in
inputprocessor

	public void setTempdeposit2(int tempdeposit) //set the temp deposit invoked in
inputprocessor of account
	public int getTempwithdraw2() //get temp withdraw amount
	public void setTempwithdraw2(int tempwithdraw) //set tempwithdraw amount
called from inputprocessor
	public int getTemppin2() //get temperory pin
	public void setTemppin2(int temppin) //set temperory pin
	public int getUserid2() //get userid
	public void setUserid2(int userid) //set userid invoked from action
```

Actions

Public class DisplayBalance1

```
	public void showDisplaybalance(AccountDataSuper ds)
```

```
// display balance of account to invoke pass parameter data store object to get
balance from accountdata


Public class displaymenu1

Public void showdisplaymenu
//displays the menu for account


public class IncorrectIdMsg1

      public void showIncorrectIdMsg()]
//displays incorrect id msg to user

public  class IncorrectLockMsg1

      public void showIncorrectLockMsg()
//displays incorrect lock message to user

Public class IncorrectPinMsg1
//displays incorrectpin msg to user


Public class IncorrectUnlockMsg1
//displays incorrect unlock message to user

Public class MakeDeposit


      public void showMakeDeposit(AccountDataSuper ds)
//function to make deposit to account
It gets the tempdeposit from account data using pointer ds
And then adds that deposit to balance amount

Public class makewithdraw

      public void showMakeWithdraw(AccountDataSuper ds)
 //function to make withdraw in account
It gets the temp withdraw amount from accountdata and updates it to balance amount


Public class noFundsMsg1

      public void showNoFundsMsg()
It displays message no funds if balance is less than minimum



Public class penalty1
      public void showPenalty(AccountDataSuper ds) //it performs penalty on account
and reduces the balance by 20 and store balance in data store
```

```
public class PromptsForPin1


	public void showPromptsForPin()

//it prompts the user for pin

Public class store data

	public void showStoreData(AccountDataSuper ds)
//gets the temppin userid and balance
And stores in permamanent variable pin userid balance in accountdata


Public class TooManyAttemptsMsg1

//displays too many attemptes message to user
```

Source Code Responsible for State Pattern

```
package MDAEFSM;
//mdaefsm class that calls each meta events from states and perform the transitions based on state
public class MDAEFSM {

	State slist[]={new Start(this),new Idle(this),new CheckPin(this),new Ready(this),new S1(this),new
Overdrawn(this),new Locked(this),new Suspended(this),new Stop()};

	State s=slist[0];
	int attempts;

public void Open(){

	int id=s.get_id();
	System.out.println("State Id:"+s.get_id());
	if(id==0)
	{
		System.out.println("change of state from start to idle");
		s.open();
		s=slist[1];
		System.out.println("New state Id:"+s.get_id());
	}
	else
	{		System.out.println("Inavalid Input");
	}
}
public void Login(){
int id=s.get_id();
System.out.println("id is"+id);
System.out.println("inside mdaefsm Login");
```

```
		if(id==1)
				{
			System.out.println("change of state from idle to CheckPin");

		s.Login();
		s=slist[2];
		System.out.println("New state Id:"+s.get_id());

				}
		else
		{
			System.out.println("Invalid Input");
		}
}
public void IncorrectLogin(){
		int id=s.get_id();
		System.out.println("id is"+id);
		System.out.println("inside incorrect login");
		if(id==1)
			{
			System.out.println("Stays is  Idle state:");
			s.IncorrectLogin();
		//s=slist[1];
			}
		else
		{
			System.out.println("Invalid Input");
		}
}
public void IncorectPin(int max){
		int id=s.get_id();

		if(id==2){
		if(getAttempts()<max){
			System.out.println("Stays is  CheckPin state:");
		s.IncorectPin(max);
		s=slist[2];}
		else{
			System.out.println("Current State:Idle:");
			s.IncorectPin(max);
			s=slist[1];
		}
		}else
		{
			System.out.println("Invalid Input");
		}
}
public void CorrectPinBelowMin(){
```

```
		int id=s.get_id();
		System.out.println("inside correct pin below  min");
		System.out.println("State Id:"+s.get_id());
		if(id==2){
			System.out.println("idle below min");
		s.CorrectPinBelowMin();
		s=slist[5];
		}
		else{
			System.out.println("Invalid Input");
		}
}
public void CorrectPinAboveMin(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==2){
		System.out.println("go to Ready state");
		s.CorrectPinAboveMin();
		s=slist[3];
		}else
		{
			System.out.println("Invalid Input");
		}
}
public void Deposit(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==3)
		{
			System.out.println("ready state deposit");
		s.Deposit();
		}
		else if(id==5)
		{

			System.out.println("overdrawn state deposit");
		s.Deposit();
		s=slist[4];
		}
		else
			{
			System.out.println("Invalid Input");
		}
}
public void BelowMinBalance(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==4){
```

```
		System.out.println("Current State:s1");
		s.BelowMinBalance();
		s=slist[5];
		}
		else
		{
			System.out.println("Invalid Input");
		}
}
public void AboveMinBalance(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
if(id==4)
		{System.out.println("Current State:s1");
		s.AboveMinBalance();
		s=slist[3];
		}
else
		{System.out.println("Invalid Input");
}
}
public void Logout(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==3)
		{System.out.println("Current State:ready");
		//s.AboveMinBalance();
		s.Logout();

		s=slist[1];
		}
else if(id==5)
{System.out.println("Current State:overdrawn");
		//s.AboveMinBalance();
		s.Logout();

		s=slist[1];
}
else if(id==2)
{
		s.Logout();

		System.out.println("Current State:checkpin");
		s=slist[1];

}
else
		{System.out.println("Invalid Input");
```

```
}}
public void Balance(){
	int id=s.get_id();
	System.out.println("State Id:"+s.get_id());
	System.out.println("Display Balance::"+s.get_id());
	if(id==3){
	System.out.println("Current state:Ready");
	s.Balance();
	}
	if(id==5)
	{
		System.out.println("Current state:Overdrawn");
	s.Balance();
	}
	if(id==7)
		{System.out.println("Current state:Suspended");
	s.Balance();
	}
	else
	{
 System.out.println("Invalid Input");
}
}
public void Withdraw(){
	int id=s.get_id();
	System.out.println("State Id:"+s.get_id());
	if(id==3){
		System.out.println("Making a Withdraw:");
	s.Withdraw();
	s=slist[4];
	}
	if(id==5)
		{System.out.println("Making a Withdraw:");
	s.Withdraw();
	//s=slist[4];
		}
	else
	{	System.out.println("Invalid Input");
	}
}
public void WithdrawBelowMinBalance(){
	int id=s.get_id();
	System.out.println("State Id:"+s.get_id());
	if(id==4)
			{System.out.println("Withdraw with penality");
	s.WithdrawBelowMinBalance();
	s=slist[5];
			}
```

```
		else
		{
			System.out.println("Invalid Input");
		}
}
public void NoFunds(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==3){
			System.out.println("No funds:");
		s.NoFunds();
		}
		else
		{
			System.out.println("Invalid Input");
		}
}
public void Lock(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==3)
			{System.out.println("Locking the account:Change of state from ready to locked");
		s.Lock();
		s=slist[6];
			}
		if(id==5)
			{System.out.println("Locking the account:Change of state from overdrawn to locked");
		s.Lock();
		s=slist[6];
			}
		else
			{System.out.println("Invalid Input");
		}
}
public void IncorrectLock(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==3)
			{System.out.println("Incorrect Lock:ready state");
		s.IncorrectLock();
			}
		if(id==5)
			{System.out.println("Incorrect Lock:overdrawn state");
		s.IncorrectLock();
			}
		else
			{
			System.out.println("Invalid Input");
```

```
		}
}
public void Unlock(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==6){
			System.out.println("unLocking the account:Change of state from locked to s1");
		s.Unlock();
		s=slist[4];
		}
		else
			{System.out.println("Invalid Input");
		}
}
public void IncorrectUnlock(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==6)
		{
		s.IncorrectUnlock();
		System.out.println("Incorrect Unlock; in locked state");

		}

		else
			{System.out.println("Invalid Input");
		}
}
public void Suspend(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==3)
		{
		s.Suspend();
		System.out.println("Suspend the account:Change of state from ready to suspended");
		s=slist[7];
		}
		else
			{System.out.println("Invalid Input");
		}
}
public void Activate(){
		int id=s.get_id();
		System.out.println("State Id:"+s.get_id());
		if(id==7)
			{System.out.println("Activate a suspended account:");
		s.Activate();
		s=slist[3];
```

```
                }

        else
                {System.out.println("Invalid Input");
        }
}
public void Close(){
        int id=s.get_id();
if(id==7)
        {System.out.println("Closing the account:");
        s=slist[8];
        }
else
        {System.out.println("Invalid Input");
        }
}
public int getAttempts() {
        return attempts;
}
public void setAttempts(int attempts) {
        this.attempts = attempts;
}

}

package MDAEFSM;

import Actions.OutputProcessor;
import Factory.AbstractFactory;

//abstract state class that stores each state events
public class State {

        static OutputProcessor  op;
        static MDAEFSM m;
        int id;

public int get_id()
{
        return id;
}


public void initalize(OutputProcessor op,MDAEFSM m){
        this.op=op;
        this.m=m;
}
```

```
public void open(){
}
public void Login(){}
public void IncorrectLogin(){}
public void IncorectPin(int max){}
public void CorrectPinBelowMin(){}
public void CorrectPinAboveMin(){}
public void Deposit(){}
public void BelowMinBalance(){}
public void AboveMinBalance(){}
public void Logout(){}
public void Balance(){}
public void Withdraw(){}
public void WithdrawBelowMinBalance(){}
public void NoFunds(){}
public void Lock(){}
public void IncorrectLock(){}
public void Unlock(){}
public void IncorrectUnlock(){}
public void Suspend(){}
public void Activate(){}

}
package MDAEFSM;


//start state class to open an account
public class Start extends State{

	public Start(MDAEFSM mdaefsm) {
		id=0;
	}

	@Override
	public void open(){
		System.out.println("in start state");
		op.showStoredata();
	}
}
package MDAEFSM;

//idle state class to login into an account
public class Idle extends State{

	public Idle(MDAEFSM mdaefsm) {

		id=1;
		}
```

```
	@Override
	public void Login(){
		System.out.println("inside idle state login function");
		System.out.println("Log in sucessful");
		System.out.println("m is "+m);
		m.setAttempts(0);

		op.showPromptsforpin();
	}
	@Override
	public void IncorrectLogin(){

		System.out.println("inside idle state incorrectlogin function");
		System.out.println("output processor is "+op);
		op.showIncorrectid();
	}


}
package MDAEFSM;


//checkpin state class to check if user entered properpin and display menu based on balance if pin is
correct
public class CheckPin extends State {

	public CheckPin(MDAEFSM mdaefsm) {
	id=2;
	}

	@Override
	public void IncorectPin(int max){
		int attempts=m.getAttempts();
		if(attempts<max){

			attempts++;
			m.setAttempts(attempts);
			op.showIncorrectpin();
		}
		else if(attempts==max){
			op.showIncorrectpin();
			op.showToomanyattempts();		}

		}

	@Override
	public void CorrectPinBelowMin(){
		op.showdisplaymenu();
```

```
	}
	@Override
	public void CorrectPinAboveMin(){
		op.showdisplaymenu();

	}


}
package MDAEFSM;

//ready state class that allows user to perform deposit withdraw in account
public class Ready extends State{

	public Ready(MDAEFSM mdaefsm) {
		// TODO Auto-generated constructor stub
		id=3;
	}

	@Override
	public void Deposit(){
		op.showMakedeposit();
	}

	@Override
	public void Balance(){
		op.showDisplaybalance();
	}
	@Override
	public void Withdraw(){
		op.showMakewithdraw();
	}
	@Override
	public void IncorrectLock(){
		op.showIncorrectlock();
	}
	@Override
	public void NoFunds(){
		op.showNofunds();
	}

}
package MDAEFSM;


//overdrawn state class that allows user to perform deposit withdraw or check balance in account
public class Overdrawn extends State {
```

```
	public Overdrawn(MDAEFSM mdaefsm) {
		// TODO Auto-generated constructor stub
		id=5;
	}
	@Override
	public void Deposit(){
		op.showMakedeposit();
	}

	@Override
	public void Balance(){
		op.showDisplaybalance();
	}
	@Override
	public void Withdraw(){
		System.out.println("inside overdrawn withdraw");
		op.showNofunds();
	}
	@Override
	public void IncorrectLock(){
		op.showIncorrectlock();
	}
}
package MDAEFSM;


//intermediate s1 state class that perform withdraw with penalty if balance is less
public class S1 extends State{

	public S1(MDAEFSM mdaefsm) {
		// TODO Auto-generated constructor stub
		id=4;

	}

	@Override
	public void WithdrawBelowMinBalance(){
		op.showPenalty();
	}


}
package MDAEFSM;

//locked state class that puts user in lock state and if user unlocks with incorrectpin
//calls action to display incorrect
public class Locked extends State{
```

```
	public Locked(MDAEFSM mdaefsm) {
		// TODO Auto-generated constructor stub
	id=6;
	}

	@Override
	public void IncorrectUnlock(){
		op.showIncorrectunlock();
	}

}
package MDAEFSM;

//suspended state allows operation display balance
public class Suspended extends State{

	public Suspended(MDAEFSM mdaefsm) {
		// TODO Auto-generated constructor stub

		id=7;
	}
	@Override
	public void Balance(){
		op.showDisplaybalance();
	}

}
package MDAEFSM;

public class Stop extends State{

	public Stop() {
		id=8;
	}

}
```

Source Code Responsible for Abstract Factory Pattern

```
package Factory;

//abstract class
import Actions.DisplayBalance1;
import Actions.DisplayBalanceSuper;
import Actions.DisplayMenu1;
```

```
import Actions.DisplayMenuSuper;
import Actions.IncorrectIdMsg1;
import Actions.IncorrectIdMsgSuper;
import Actions.IncorrectLockMsg1;
import Actions.IncorrectLockMsgSuper;
import Actions.IncorrectPinMsg1;
import Actions.IncorrectPinMsgSuper;
import Actions.IncorrectUnlockMsg1;
import Actions.IncorrectUnlockMsgSuper;
import Actions.MakeDeposit1;
import Actions.MakeDepositSuper;
import Actions.MakeWithdraw1;
import Actions.MakeWithdrawSuper;
import Actions.NoFundsMsg1;
import Actions.NoFundsMsgSuper;
import Actions.Penalty1;
import Actions.PenaltySuper;
import Actions.PromptsForPin1;
import Actions.PromptsForPinSuper;
import Actions.StoreData1;
import Actions.StoreDataSuper;
import Actions.TooManyAttemptsMsg1;
import Actions.TooManyAttemptsMsgSuper;
import DataStore.AccountData1;
import DataStore.AccountDataSuper;

public class AbstractFactory {

	public AccountDataSuper getAccountData(){
		return null;
		}


	public DisplayBalanceSuper getDisplayBalance(){
		return null;
		}


	public DisplayMenuSuper getDisplayMenu(){
		return null;
			}

	public IncorrectIdMsgSuper getIncorrectIdMsg(){
		return null;

	}

	public IncorrectLockMsgSuper getIncorrectLockMsg(){
```

```
        return null;

}


public IncorrectPinMsgSuper getIncorrectPinMsg(){

        return null;

}


public IncorrectUnlockMsgSuper getIncorrectUnlockMsg(){

        return null;

}


public MakeDepositSuper getDeposit(){
        return null;

}


public MakeWithdrawSuper getWithdraw(){
        return null;

}



public NoFundsMsgSuper getNoFundsMsg(){
        return null;

}

public PenaltySuper getPenalty(){
        return null;
}

public StoreDataSuper getStoreData(){
        return null;
}

public PromptsForPinSuper getPromptForPin(){
        return null;
```

```
	}

	public TooManyAttemptsMsgSuper getTooManyAttempts(){
		return null;
	}
}
package Factory;

import Actions.DisplayBalance1;
import Actions.DisplayBalanceSuper;
import Actions.DisplayMenu1;
import Actions.DisplayMenuSuper;
import Actions.IncorrectIdMsg1;
import Actions.IncorrectIdMsgSuper;
import Actions.IncorrectLockMsg1;
import Actions.IncorrectLockMsgSuper;
import Actions.IncorrectPinMsg1;
import Actions.IncorrectPinMsgSuper;
import Actions.IncorrectUnlockMsg1;
import Actions.IncorrectUnlockMsgSuper;
import Actions.MakeDeposit1;
import Actions.MakeDepositSuper;
import Actions.MakeWithdraw1;
import Actions.MakeWithdrawSuper;
import Actions.NoFundsMsg1;
import Actions.NoFundsMsgSuper;
import Actions.Penalty1;
import Actions.PenaltySuper;
import Actions.PromptsForPin1;
import Actions.PromptsForPinSuper;
import Actions.StoreData1;
import Actions.StoreDataSuper;
import Actions.TooManyAttemptsMsg1;
import Actions.TooManyAttemptsMsgSuper;
import DataStore.AccountData1;
import DataStore.AccountDataSuper;

//class to create object for action of each account 1 events
public class ClientFactory1 extends AbstractFactory{
	@Override
	public AccountDataSuper getAccountData(){
		System.out.println("create acc1 object");
		return new AccountData1();
	}

	@Override
	public DisplayBalanceSuper getDisplayBalance(){
		return new DisplayBalance1();
```

```
}

@Override
public DisplayMenuSuper getDisplayMenu(){
        return new DisplayMenu1();
        }

@Override
public IncorrectIdMsgSuper getIncorrectIdMsg(){
        return new IncorrectIdMsg1();
}
@Override
public IncorrectLockMsgSuper getIncorrectLockMsg(){
        return new IncorrectLockMsg1();
}

@Override
public IncorrectPinMsgSuper getIncorrectPinMsg(){
        return new IncorrectPinMsg1();
}

@Override
public IncorrectUnlockMsgSuper getIncorrectUnlockMsg(){
        return new IncorrectUnlockMsg1();
}

@Override
public MakeDepositSuper getDeposit(){
        return new MakeDeposit1();
}

@Override
public MakeWithdrawSuper getWithdraw(){
        return new MakeWithdraw1();
}

@Override
public NoFundsMsgSuper getNoFundsMsg(){
        return new NoFundsMsg1();
}

@Override
public PenaltySuper getPenalty(){
        return new Penalty1();
}

@Override
public StoreDataSuper getStoreData(){
```

```
		return new StoreData1();
	}

	@Override
	public PromptsForPinSuper getPromptForPin(){
		return new PromptsForPin1();
	}

	@Override
	public TooManyAttemptsMsgSuper getTooManyAttempts(){
		return new TooManyAttemptsMsg1();
	}


}
package Factory;

import Actions.DisplayBalance2;
import Actions.DisplayBalanceSuper;
import Actions.DisplayMenu2;
import Actions.DisplayMenuSuper;
import Actions.IncorrectIdMsg2;
import Actions.IncorrectIdMsgSuper;
import Actions.IncorrectPinMsg2;
import Actions.IncorrectPinMsgSuper;
import Actions.MakeDeposit2;
import Actions.MakeDepositSuper;
import Actions.MakeWithdraw2;
import Actions.MakeWithdrawSuper;
import Actions.NoFundsMsg2;
import Actions.NoFundsMsgSuper;
import Actions.PenaltySuper;
import Actions.PromptsForPin2;
import Actions.PromptsForPinSuper;
import Actions.StoreData2;
import Actions.StoreDataSuper;
import Actions.TooManyAttemptsMsg2;
import Actions.TooManyAttemptsMsgSuper;
import DataStore.AccountData2;
import DataStore.AccountDataSuper;

//class to create object for action of each account 2 events
public class ClientFactory2 extends AbstractFactory{

	@Override
	public AccountDataSuper getAccountData(){
		return new AccountData2();
	}
```

```
	@Override
	public DisplayBalanceSuper getDisplayBalance(){
		return new DisplayBalance2();
	}
	@Override
	public DisplayMenuSuper getDisplayMenu(){
		return new DisplayMenu2();
		}
	@Override
	public IncorrectIdMsgSuper getIncorrectIdMsg(){
		return new IncorrectIdMsg2();
	}

	@Override
	public IncorrectPinMsgSuper getIncorrectPinMsg(){
		return new IncorrectPinMsg2();
	}

	@Override
	public MakeDepositSuper getDeposit(){
		return new MakeDeposit2();
	}
	@Override
	public MakeWithdrawSuper getWithdraw(){
		return new MakeWithdraw2();
	}
	@Override
	public NoFundsMsgSuper getNoFundsMsg(){
		return new NoFundsMsg2();
	}
	@Override
	public StoreDataSuper getStoreData(){
		return new StoreData2();
	}
	@Override
	public PromptsForPinSuper getPromptForPin(){
		return new PromptsForPin2();
	}
	@Override
	public TooManyAttemptsMsgSuper getTooManyAttempts(){
		return new TooManyAttemptsMsg2();
	}

}
package DataStore;

public abstract class AccountDataSuper {
```

```
public float getTempbalance() {
	// TODO Auto-generated method stub
	return 0;
}

public void setPin2(int pin) {
	// TODO Auto-generated method stub

}

public int getTempbalance2() {
	// TODO Auto-generated method stub
	return 0;
}

public void setTempbalance2(int tempbalance) {
	// TODO Auto-generated method stub

}

public int getTempuserid2() {
	// TODO Auto-generated method stub
	return 0;
}

public void setTempuserid2(int tempuserid) {
	// TODO Auto-generated method stub

}

public int getPin2() {
	// TODO Auto-generated method stub
	return 0;
}

public int getBalance2() {
	// TODO Auto-generated method stub
	return 0;
}

public void setBalance2(int balance) {
	// TODO Auto-generated method stub

}

public int getTempdeposit2() {
	// TODO Auto-generated method stub
```

```
		return 0;
	}

	public void setTempdeposit2(int tempdeposit) {
		// TODO Auto-generated method stub

	}

	public int getTempwithdraw2() {
		// TODO Auto-generated method stub
		return 0;
	}

	public void setTempwithdraw2(int tempwithdraw) {
		// TODO Auto-generated method stub

	}

	public int getTemppin2() {
		// TODO Auto-generated method stub
		return 0;
	}

	public void setTemppin2(int temppin) {
		// TODO Auto-generated method stub

	}

	public int getUserid2() {
		// TODO Auto-generated method stub
		return 0;
	}

	public void setUserid2(int userid) {
		// TODO Auto-generated method stub

	}

	public void setTempbalance(float tempbalance) {
		// TODO Auto-generated method stub

	}

	public String getTempuserid() {
		// TODO Auto-generated method stub
		return null;
	}
```

```
public void setTempuserid(String tempuserid) {
	// TODO Auto-generated method stub

}

public String getPin() {
	// TODO Auto-generated method stub
	return null;
}

public void setPin(String pin) {
	// TODO Auto-generated method stub

}

public float getBalance() {
	// TODO Auto-generated method stub
	return 0;
}

public void setBalance(float balance) {
	// TODO Auto-generated method stub

}

public float getTempdeposit() {
	// TODO Auto-generated method stub
	return 0;
}

public void setTempdeposit(float tempdeposit) {
	// TODO Auto-generated method stub

}

public float getTempwithdraw() {
	// TODO Auto-generated method stub
	return 0;
}

public void setTempwithdraw(float tempwithdraw) {
	// TODO Auto-generated method stub

}

public String getTemppin() {
	// TODO Auto-generated method stub
	return null;
```

```
	}

	public void setTemppin(String temppin) {
		// TODO Auto-generated method stub

	}

	public String getUserid() {
		// TODO Auto-generated method stub
		return null;
	}

	public void setUserid(String userid) {
		// TODO Auto-generated method stub

	}


}
package DataStore;

public class AccountData1 extends AccountDataSuper{
//class to store all data for account1
	float tempbalance;
	String tempuserid;
	String pin;
	float balance;
	float tempdeposit;
	float tempwithdraw;
	String temppin;
	String userid;

	@Override
	public float getTempbalance() {
		return tempbalance;
	}
	@Override
	public void setTempbalance(float tempbalance) {
		this.tempbalance = tempbalance;
	}

	@Override
	public String getTempuserid() {
		return tempuserid;
	}
	@Override
	public void setTempuserid(String tempuserid) {
		this.tempuserid = tempuserid;
```

```
}

@Override
public String getPin() {
	return pin;
}
@Override
public void setPin(String pin) {
	this.pin = pin;
}

@Override
public float getBalance() {
	return balance;
}

@Override
public void setBalance(float balance) {
	this.balance = balance;
}

@Override
public float getTempdeposit() {
	return tempdeposit;
}

@Override
public void setTempdeposit(float tempdeposit) {
	this.tempdeposit = tempdeposit;
}

@Override
public float getTempwithdraw() {
	return tempwithdraw;
}

@Override
public void setTempwithdraw(float tempwithdraw) {
	this.tempwithdraw = tempwithdraw;
}

@Override
public String getTemppin() {
	return temppin;
}

@Override
public void setTemppin(String temppin) {
```

```
		this.temppin = temppin;
	}

	@Override
	public String getUserid() {
		return userid;
	}

	@Override
	public void setUserid(String userid) {
		this.userid = userid;
	}

}
package DataStore;
//class to store all data for account2
public class AccountData2 extends AccountDataSuper {
	int tempbalance;
	int tempuserid;
	int pin;
	int balance;
	int tempdeposit;
	int tempwithdraw;
	int temppin;
	int userid;
	@Override
	public int getTempbalance2() {
		return tempbalance;
	}
	@Override
	public void setTempbalance2(int tempbalance) {
		this.tempbalance = tempbalance;
	}
	@Override
	public int getTempuserid2() {
		return tempuserid;
	}
	@Override
	public void setTempuserid2(int tempuserid) {
		this.tempuserid = tempuserid;
	}
	@Override
	public int getPin2() {
		return pin;
	}
	@Override
	public void setPin2(int pin) {
		this.pin = pin;
```

```
	}
	@Override
	public int getBalance2() {
		return balance;
	}
	@Override
	public void setBalance2(int balance) {
		this.balance = balance;
	}
	@Override
	public int getTempdeposit2() {
		return tempdeposit;
	}
	@Override
	public void setTempdeposit2(int tempdeposit) {
		this.tempdeposit = tempdeposit;
	}
	@Override
	public int getTempwithdraw2() {
		return tempwithdraw;
	}
	@Override
	public void setTempwithdraw2(int tempwithdraw) {
		this.tempwithdraw = tempwithdraw;
	}
	@Override
	public int getTemppin2() {
		return temppin;
	}
	@Override
	public void setTemppin2(int temppin) {
		this.temppin = temppin;
	}
	@Override
	public int getUserid2() {
		return userid;
	}
	@Override
	public void setUserid2(int userid) {
		this.userid = userid;
	}


}
```

Source Code Responsible for Stratergy Pattern

```
package Actions;

import DataStore.AccountDataSuper;
import Factory.AbstractFactory;

//this class has pointer to each action super class to call respective action
///it also contains pointer to abstractfactory and data store
//abstractfactory for creating each action class object
//datastore to store the data
public class OutputProcessor {
	private AccountDataSuper ds;
	private AbstractFactory af;
	private DisplayBalanceSuper displaybalance;
	private DisplayMenuSuper displaymenu;
	private IncorrectIdMsgSuper incorrectid;
	private IncorrectPinMsgSuper incorrectpin;
	private IncorrectLockMsgSuper incorrectlock;
	private IncorrectUnlockMsgSuper incorrectunlock;
	private MakeDepositSuper makedeposit;
	private MakeWithdrawSuper makewithdraw;
	private NoFundsMsgSuper nofunds;
	private PenaltySuper penalty;
	private PromptsForPinSuper promptsforpin;
	private StoreDataSuper storedata;
	private TooManyAttemptsMsgSuper toomanyattempts;

	public void showDisplaybalance() {
		displaybalance=af.getDisplayBalance();
		displaybalance.showDisplaybalance(ds);
	}

	public void showdisplaymenu() {

		displaymenu=af.getDisplayMenu();
		displaymenu.showDisplayMenu();
	}

	public void showIncorrectid() {
		System.out.println("inside output processor incorrect id" +af);
		incorrectid=af.getIncorrectIdMsg();
		incorrectid.showIncorrectIdMsg();
	}
```

```
public void showIncorrectpin() {
		incorrectpin=af.getIncorrectPinMsg();
		incorrectpin.showIncorrectPinMsg();
}
public void showIncorrectlock() {
		System.out.println("inside output processor incorrect lock" );
		incorrectlock=af.getIncorrectLockMsg();
		incorrectlock.showIncorrectLockMsg();
}

public void showIncorrectunlock() {
		incorrectunlock=af.getIncorrectUnlockMsg();
		incorrectunlock.showIncorrectUnlockMsg();
}

public void showMakedeposit() {
		makedeposit=af.getDeposit();

		makedeposit.showMakeDeposit(ds);
}

public void showMakewithdraw() {
		makewithdraw=af.getWithdraw();
		makewithdraw.showMakeWithdraw(ds);
}
public void showNofunds() {
		nofunds=af.getNoFundsMsg();
		nofunds.showNoFundsMsg();
}

public void showPenalty() {
		penalty=af.getPenalty();
		penalty.showPenalty(ds);
}

public void showPromptsforpin() {
		System.out.println("inside prompts for pin");

		promptsforpin=af.getPromptForPin();
		promptsforpin.showPromptsForPin();
}

public void showStoredata() {
		System.out.println("inside store data"+af + ds);
		System.out.println(ds.getTempuserid2());

		storedata=af.getStoreData();
		storedata.showStoreData(ds);
```

```
	}

	public void showToomanyattempts() {
		toomanyattempts=af.getTooManyAttempts();
	toomanyattempts.showTooManyAttempsMsg();
	}

	public void setData(AbstractFactory af,AccountDataSuper ds)
	{
		this.af=af;
		this.ds=ds;
		System.out.println(ds.getTempuserid2());
	}



}
package Actions;

import DataStore.AccountDataSuper;

public abstract class DisplayBalanceSuper {


public void showDisplaybalance(AccountDataSuper ds) {
	// TODO Auto-generated method stub

}
}
package Actions;

import DataStore.AccountDataSuper;
//displays the balance in account1
public class DisplayBalance1 extends DisplayBalanceSuper{
@Override
	public void showDisplaybalance(AccountDataSuper ds) {
		System.out.println("balance is"+ds.getBalance());
	}
}
package Actions;

import DataStore.AccountDataSuper;

//displays the balance in account2
public class DisplayBalance2 extends DisplayBalanceSuper {
	@Override
	public void showDisplaybalance(AccountDataSuper ds) {
		System.out.println("balance is"+ds.getBalance2());
```

```
	}

}
package Actions;

public class DisplayMenuSuper {

public void showDisplayMenu() {

	}
}

package Actions;

//displays the menu for account1
public class DisplayMenu1 extends DisplayMenuSuper{

	@Override
public void showDisplayMenu() {
		System.out.println("Transaction Perform " );
		System.out.println("   1: Check Balance " );
		System.out.println("   2: Deposit " );
		System.out.println("   3: Withdraw " );
		System.out.println("   4: Lock Account" );

	}

}
package Actions;


//displays the menu for account2
public class DisplayMenu2 extends DisplayMenuSuper{


	@Override
	public void showDisplayMenu() {
		System.out.println("Transaction Perform " );
		System.out.println("   1: Check Balance " );
		System.out.println("   2: Deposit " );
		System.out.println("   3: Withdraw " );
		System.out.println("   4: Lock Account" );

	}

}
package Actions;
```

```
public abstract class IncorrectIdMsgSuper {

	public void showIncorrectIdMsg() {
		// TODO Auto-generated method stub

	}
}
package Actions;


//displays the incorrectidmsg for account1

public class IncorrectIdMsg1 extends IncorrectIdMsgSuper{

	@Override
	public void showIncorrectIdMsg() {
		System.out.println("Account1:Incorrect Id");

	}

}

package Actions;

//displays the incorrectidmsg for account2

public class IncorrectIdMsg2 extends IncorrectIdMsgSuper{

	@Override
	public void showIncorrectIdMsg() {

		System.out.println("Account2:Incorrect Id");
	}
}
package Actions;

public abstract class IncorrectLockMsgSuper {

	public void showIncorrectLockMsg() {
		// TODO Auto-generated method stub

	}


}
package Actions;
```

```
//displays the incorrectlockmsg for account1

public class IncorrectLockMsg1 extends IncorrectLockMsgSuper{

	@Override
	public void showIncorrectLockMsg()
	{
		System.out.println("Account1:Incorrect LockMsg");

	}

}
package Actions;

import DataStore.AccountDataSuper;

public class MakeWithdrawSuper {



	public void showMakeWithdraw(AccountDataSuper ds) {
		// TODO Auto-generated method stub

	}
}
package Actions;

import DataStore.AccountDataSuper;


//perform withdraw to accountdata1 and update balance
public class MakeWithdraw1 extends MakeWithdrawSuper{

	@Override
	public void showMakeWithdraw(AccountDataSuper ds)
	{float balance=ds.getBalance()-ds.getTempwithdraw();

	ds.setBalance(balance);
	System.out.println("Account1 Balance:"+ds.getBalance());


	}
}
package Actions;

import DataStore.AccountDataSuper;
```

```
//perform withdraw to accountdata1 and update balance
public class MakeWithdraw2 extends MakeWithdrawSuper {

	@Override
	public void showMakeWithdraw(AccountDataSuper ds)
	{
		int balance=ds.getBalance2()-ds.getTempwithdraw2();
		ds.setBalance2(balance);
		System.out.println("Account2 Balance:"+ds.getBalance2());

	}
}
package Actions;

import DataStore.AccountDataSuper;

public class MakeDepositSuper {




	public void showMakeDeposit(AccountDataSuper ds) {
		// TODO Auto-generated method stub

	}
}
package Actions;

import DataStore.AccountDataSuper;
//perform deposit to accountdata1 and update balance

public class MakeDeposit1 extends MakeDepositSuper {

	@Override
	public void showMakeDeposit(AccountDataSuper ds)
	{
		float balance=ds.getBalance()+ds.getTempdeposit();
		ds.setBalance(balance);
		System.out.println("Updated Balance :"+ds.getBalance());
	}
}
package Actions;

import DataStore.AccountDataSuper;


//perform deposit to accountdata1 and update balance
```

```
public class MakeDeposit2 extends MakeDepositSuper{


	@Override
	public void showMakeDeposit(AccountDataSuper ds)
	{
		int balance=ds.getBalance2()+ds.getTempdeposit2();
		ds.setBalance2(balance);
		System.out.println("Updated Balance :"+ds.getBalance2());

	}

}
package Actions;

public class IncorrectUnlockMsgSuper {

	public void showIncorrectUnlockMsg()
	{

	}

}
package Actions;

//displays incorrect unclock messsage for account1
public class IncorrectUnlockMsg1 extends IncorrectUnlockMsgSuper{


	@Override
	public void showIncorrectUnlockMsg()
	{
		System.out.println("Account1:Incorrect UnlockMsg");

	}
}
package Actions;

public class IncorrectPinMsgSuper {

	public void showIncorrectPinMsg() {
		// TODO Auto-generated method stub

	}

}
package Actions;
```

```
//displays the incorrectpinmsg for account1

public class IncorrectPinMsg1 extends IncorrectPinMsgSuper{


	@Override
	public void showIncorrectPinMsg()
	{
		System.out.println("Account1:Incorrect PinMsg");

	}

}
package Actions;

//displays the incorrectpinmsg for account2

public class IncorrectPinMsg2 extends IncorrectPinMsgSuper{


	@Override
	public void showIncorrectPinMsg()
	{
		System.out.println("Account2:Incorrect PinMsg");

	}
}
package Actions;

public class TooManyAttemptsMsgSuper {

	public void showTooManyAttempsMsg()
	{

	}
}
package Actions;


//display toomany attempts message to user
public class TooManyAttemptsMsg1  extends TooManyAttemptsMsgSuper{

	@Override
	public void showTooManyAttempsMsg()
	{
		System.out.println("too many attempts done on account1");
	}
}
```

```
package Actions;


//display toomanyattempts message to user
public class TooManyAttemptsMsg2 extends TooManyAttemptsMsgSuper{

	@Override
	public void showTooManyAttempsMsg()
	{
		System.out.println("too many attempts done on account2");

	}
}
package Actions;

import DataStore.AccountDataSuper;

public class StoreDataSuper {



	public void showStoreData(AccountDataSuper ds) {
		// TODO Auto-generated method stub

	}

}
package Actions;

import DataStore.AccountDataSuper;
//store account data pin userid balance in accountdata
public class StoreData1 extends StoreDataSuper {

	@Override
	public void showStoreData(AccountDataSuper ds)
	{
		ds.setUserid(ds.getTempuserid());
		ds.setPin(ds.getTemppin());
		ds.setBalance(ds.getTempbalance());
		System.out.println("Account1:The data  stored");
		System.out.println("Account1"+ds.getUserid());

	}

}
package Actions;

import DataStore.AccountDataSuper;
```

```
//store account data pin userid balance in accountdata
public class StoreData2 extends StoreDataSuper{

	@Override
	public void showStoreData(AccountDataSuper ds)
	{
		ds.setUserid2(ds.getTempuserid2());
		ds.setPin2(ds.getTemppin2());
		ds.setBalance2(ds.getTempbalance2());
		System.out.println("Account2:The data  stored");
		System.out.println("Account2"+ds.getUserid2());

	}

}
package Actions;

public class PromptsForPinSuper {


	public void showPromptsForPin()
	{

	}
}
package Actions;
//propmpts user for pin
public class PromptsForPin1 extends PromptsForPinSuper{

	@Override
	public void showPromptsForPin()
	{
	System.out.println("prompt user for pin account1");
	}
}
package Actions;


//propmpts user for pin
public class PromptsForPin2 extends PromptsForPinSuper{

	@Override
	public void showPromptsForPin()
	{
		System.out.println("prompt user for pin account2");

	}
```

```
}
package Actions;

import DataStore.AccountDataSuper;

public class PenaltySuper {




	public void showPenalty(AccountDataSuper ds) {
		// TODO Auto-generated method stub

	}
}
package Actions;

import DataStore.AccountDataSuper;
//perform penalty and reduce balance ammount by 20
public class Penalty1 extends PenaltySuper{
	@Override
	public void showPenalty(AccountDataSuper ds)
	{
		float balance=ds.getBalance()-20;
	ds.setBalance(balance);
	System.out.println("Updated Balance:"+ds.getBalance());
}

	}

package Actions;

public class NoFundsMsgSuper {

	public void showNoFundsMsg()
	{

	}

}
package Actions;


//display no funds message
public class NoFundsMsg1 extends NoFundsMsgSuper{

	@Override
	public void showNoFundsMsg()
```

```
	{
		System.out.println("No Amount in account1 to perform action");
	}
}
package Actions;
//display no funds message
public class NoFundsMsg2 extends NoFundsMsgSuper{

	@Override
	public void showNoFundsMsg()
	{
		System.out.println("No Amount in account2 to perform action");

	}
}
```

Other classes

```
package InputProcessor;

import DataStore.AccountDataSuper;
import Factory.AbstractFactory;
import MDAEFSM.MDAEFSM;
//inputprocessor that invokes the respective events of mdaefsm for account1 user
public class Account1
{
	private AccountDataSuper ds;
	private AbstractFactory af;
	private MDAEFSM m;
	public Account1(MDAEFSM m,AccountDataSuper ds)
	{
		this.m=m;
		this.ds=ds;
	}
public void open(String p, String y, float a) {
		// store p, y and a in temp data store
	 System.out.println("inside account1 open");

		ds.setTemppin(p);
		 ds.setTempbalance(a);
		 ds.setTempuserid(y);
		 m.Open();
		}

public void	pin (String x) {
		 if (x.equals(ds.getPin())) {
			System.out.println("pin is" +x);
```

```
                    System.out.println("pin is" +ds.getPin());
                    if (ds.getBalance() > 500)
                    m.CorrectPinAboveMin();
                    else m.CorrectPinBelowMin();
                    }
                    else m.IncorectPin(3);
                   }

public void deposit (float d) {
                    ds.setTempdeposit(d);
                    m.Deposit();
                    if (ds.getBalance()>500)
                    m.AboveMinBalance();
                    else m.BelowMinBalance();
                   }

public void        withdraw (float w) {
                    ds.setTempwithdraw(w);
                    m.Withdraw();
                    if (ds.getBalance()>500)
                    m.AboveMinBalance();
                    else
                    {
                            System.out.println("withdrawbelowmin balance");
                            m.WithdrawBelowMinBalance();

                    }}
          public void balance() {m.Balance();}

          public void        login (String y) {
                    System.out.println("y "+y);
                    System.out.println("y "+ds.getUserid());
                    if (y.equals(ds.getUserid()))
                    {
                            System.out.println("user id equal inside account 1 login");
                    m.Login();
                    }
                    else m.IncorrectLogin();
                   }
                   public void logout() {m.Logout();}

                   public void lock (String x)
                   {
                    if (ds.getPin().equals(x))
                            m.Lock();
                    else m.IncorrectLock();
                   }
          public void        unlock (String x) {
```

```
                    if (x.equals(ds.getPin())) {
                    m.Unlock();
                    if (ds.getBalance() > 500)
                    m.AboveMinBalance();
                    else m.BelowMinBalance();
                    }
                    else m.IncorrectUnlock();
                   }
        public String showuserid()
        {
                return ds.getUserid();
        }
        public String showpin()
        {
                return ds.getPin();
        }
        public float showBalance()
        {
                return ds.getBalance();
        }
        public float showDeposit()
        {
                return ds.getTempdeposit();
        }
        public float showWithdraw()
        {
                return ds.getTempwithdraw();
        }


}
package InputProcessor;

import DataStore.AccountDataSuper;
import Factory.AbstractFactory;
import MDAEFSM.MDAEFSM;

//inputprocessor that invokes the respective events of mdaefsm for account2 user
public class Account2
{
        private AccountDataSuper ds;
        private AbstractFactory af;
        private MDAEFSM m;
        public Account2(MDAEFSM m,AccountDataSuper ds)
        {
                this.m=m;
                this.ds=ds;
        }
```

```
public void OPEN (int p, int y, int a) {
// store p, y and a in temp data store
 ds.setTemppin2(p);
 ds.setTempuserid2(y);
 System.out.println("userid"+ds.getTempuserid2());
 ds.setTempbalance2(a);
 m.Open();
}
public void PIN (int x) {
 if (x==ds.getPin2())
 m.CorrectPinAboveMin();
 else m.IncorectPin(2);
}
public void DEPOSIT (int d) {
 ds.setTempdeposit2(d);
 m.Deposit();
}
public void WITHDRAW (int w) {
 ds.setTempwithdraw2(w);
 if (ds.getBalance2()>0){
 m.Withdraw();
 m.AboveMinBalance();
 }
 else m.NoFunds();
}
public void BALANCE() {m.Balance();}

public void LOGIN (int y) {
            System.out.println(ds + ""+y+""+ds.getUserid2());
 if (y==ds.getUserid2())
 {            System.out.println("inside acc 2 login");
 m.Login();
}
 else m.IncorrectLogin();
}
public void LOGOUT() {m.Logout();}
public void suspend () {
 m.Suspend();
}
public void activate () {
 m.Activate();
}
public void close () {
 m.Close();
}
```

```
}
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

import Actions.OutputProcessor;
import DataStore.AccountData2;
import DataStore.AccountDataSuper;
import Factory.AbstractFactory;
import Factory.ClientFactory1;
import Factory.ClientFactory2;
import InputProcessor.Account1;
import InputProcessor.Account2;
import MDAEFSM.MDAEFSM;
import MDAEFSM.State;

public class driver {

        public static void main(String args[])
        {
                System.out.println("Enter input");
     System.out.println("1. Account- 1");
     System.out.println("2.  Account- 2");
     System.out.println("3. Exit");
    int choice;

    Scanner input = new Scanner(System.in);
     try {
       choice = input.nextInt();
       if(choice==1)
       {

         OutputProcessor op=new OutputProcessor();
         MDAEFSM m=new MDAEFSM();
         AbstractFactory af=new ClientFactory1();
         AccountDataSuper ds= af.getAccountData();
         op.setData(af,ds);
         State s=new State();
         s.initalize(op, m);
         Account1 a =new Account1(m,ds);
         driver d =new driver();
         d.processOperation(a);
       }
       if(choice==2)
       {
         OutputProcessor op=new OutputProcessor();
         MDAEFSM m=new MDAEFSM();
```

```
        AbstractFactory af=new ClientFactory2();
        AccountDataSuper ds= af.getAccountData();
        op.setData(af, ds);
        State s=new State();
        s.initalize(op, m);

        Account2 a =new Account2(m,ds);
        driver d =new driver();
        d.processOperation2(a);

      }
    }
    catch(Exception e)
    {

    }

        }

        public void processOperation(Account1 a)
        {
                int flag=0;
                while(flag==0)
                {

                System.out.println("Select
Operation:\n1.Open\n2.Pin\n3.Deposit\n4.Withdraw\n5.Balance\n6.Login\n7.Logout\n8.Lock\n9.Unloc
k\n10.exit\n");
                BufferedReader b=new BufferedReader(new InputStreamReader(System.in));
                int process = 0;
                try {
                        process = Integer.parseInt(b.readLine());
                } catch (NumberFormatException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                Scanner input=new Scanner(System.in);
                switch(process){
                case 1:
                        System.out.println("\nEnter the Accountno:");
                        String userid=input.nextLine();
                        System.out.println("\nEnter the pin:");
                        String pin=input.nextLine();
                        System.out.println("\nEnter the Balance:");
                        float balance=input.nextFloat();
```

```
                a.open(pin, userid, balance);
                break;
        case 2:
                System.out.println("\n Enter the pin:");
                pin=input.nextLine();
                a.pin(pin);
                break;
        case 3:
                System.out.println("\n Enter the amount to be deposited:");
                float d=input.nextFloat();
                a.deposit(d);
                break;
        case 4:
                System.out.println("\nPlease Enter the amount to withdraw:");
                float w=input.nextFloat();
                a.withdraw(w);
                break;
        case 5:
                a.balance();
                break;
        case 6:
                System.out.println("Enter the user id for login:");
                userid=input.nextLine();
                a.login(userid);
                break;
        case 7:
                a.logout();
                break;
        case 8:
                System.out.println("Enter the pin to lock the account:");
                pin=input.nextLine();
                a.lock(pin);
                break;
        case 9:
                System.out.println("Enter the pin to unlock the account:");
                pin=input.nextLine();
                a.unlock(pin);
                break;
        case 10:
                flag=1;
        break;
        }
        }

}

public void processOperation2(Account2 a)
{
```

```
		int flag=0;
		while(flag==0)
		{

		System.out.println("Select
Operation:\n1.Open\n2.Pin\n3.Deposit\n4.Withdraw\n5.Balance\n6.Login\n7.Logout\n8.Suspend\n9.A
ctivate\n10.close\n11.exit");
		BufferedReader b=new BufferedReader(new InputStreamReader(System.in));
		int process = 0;
		try {
			process = Integer.parseInt(b.readLine());
		} catch (NumberFormatException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		Scanner input=new Scanner(System.in);
		switch(process){
		case 1:
			System.out.println("\nEnter the Accountno:");
			int userid=input.nextInt();
			System.out.println("\nEnter the pin:");
			int pin=input.nextInt();
			System.out.println("\nEnter the Balance:");
			int balance=input.nextInt();
			a.OPEN(pin, userid, balance);
			break;
		case 2:
			System.out.println("\n Enter the pin:");
			pin=input.nextInt();
			a.PIN(pin);
			break;
		case 3:
			System.out.println("\n Enter the amount to be deposited:");
			int d=input.nextInt();
			a.DEPOSIT(d);
			break;
		case 4:
			System.out.println("\nPlease Enter the amount to withdraw:");
			int w=input.nextInt();
			a.WITHDRAW(w);
			break;
		case 5:
			a.BALANCE();
			break;
		case 6:
```

```
                    System.out.println("Enter the user id for login:");
                    userid=input.nextInt();
                    a.LOGIN(userid);
                    break;
                case 7:
                    a.LOGOUT();
                    break;
                case 8:
                    System.out.println("Suspend the account:");
                    a.suspend();
                    break;
                case 9:
                    System.out.println("Activate an account:");
                    a.activate();
                    break;
                case 10:
                    System.out.println("Close an account:");
                    a.close();
                    break;
                case 11:
                    flag=1;
                break;
                }
                }
        }
}
```