Software Architecture-Design Document

2016/4/29
CS586:Project

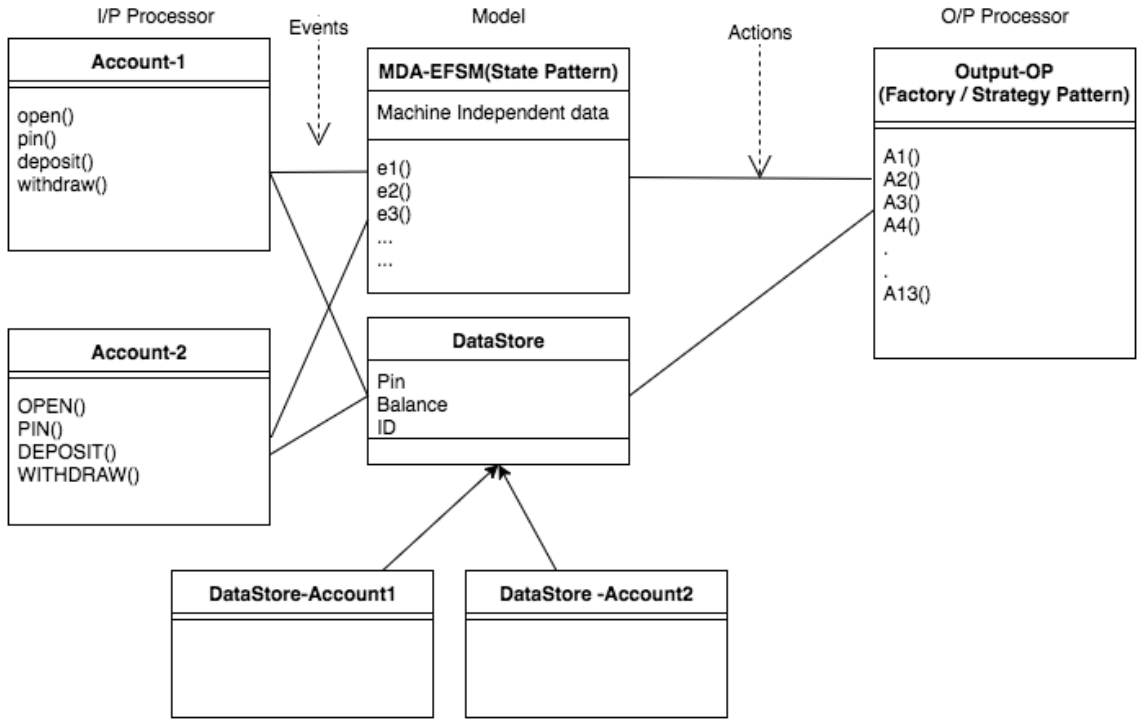
A20354692
Fei Shen(fshen4@hawk.iit.edu)

1 Model Driven Architecture of the ATM ACCOUNT Components

1.1 General MDA Architecture of ATM ACCOUNT Components

This project implements two ACCOUNTs implemented using Model Driven Architecture. The ACCOUNTs differ in their user interface specifications, but are similar in behavior. The main goal is to separate the Platform specific portion of the system from the Platform independent portion.

The general outline of the implementation is to use Model-Driven-Architecture as the overall design, and use an Extended-Finite-State-Machine to capture the states in which the model can be in at any point of time. The project implements the Output Processor using Strategy pattern, and uses Abstract Factory Pattern to select and initialize a list of output actions for each ACCOUNT.



1.2 MDA-EFSM model for the Account compone

The main purpose of MDA-EFSM is to capture the meta-behavior of all Accounts. States, Events, Actions, pseudo-code and State diagrams are the same as provided in the class.

1. List of Events of the MDA-EFSM

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

2. List of Actions of the MDA-EFSM

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

3. Psuedo-Code of Account classes:

Account Class:
Purpose:
User interface or input class
Responsibilities:

Responsible for storing data into temporary data store and interface between user and

system Collaborators:
MDA-EFSM, Data Store
Attributes:
Pseudocode:

Operations of the Input Processor (Account-1)

```
create()
{
m->create();
}
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

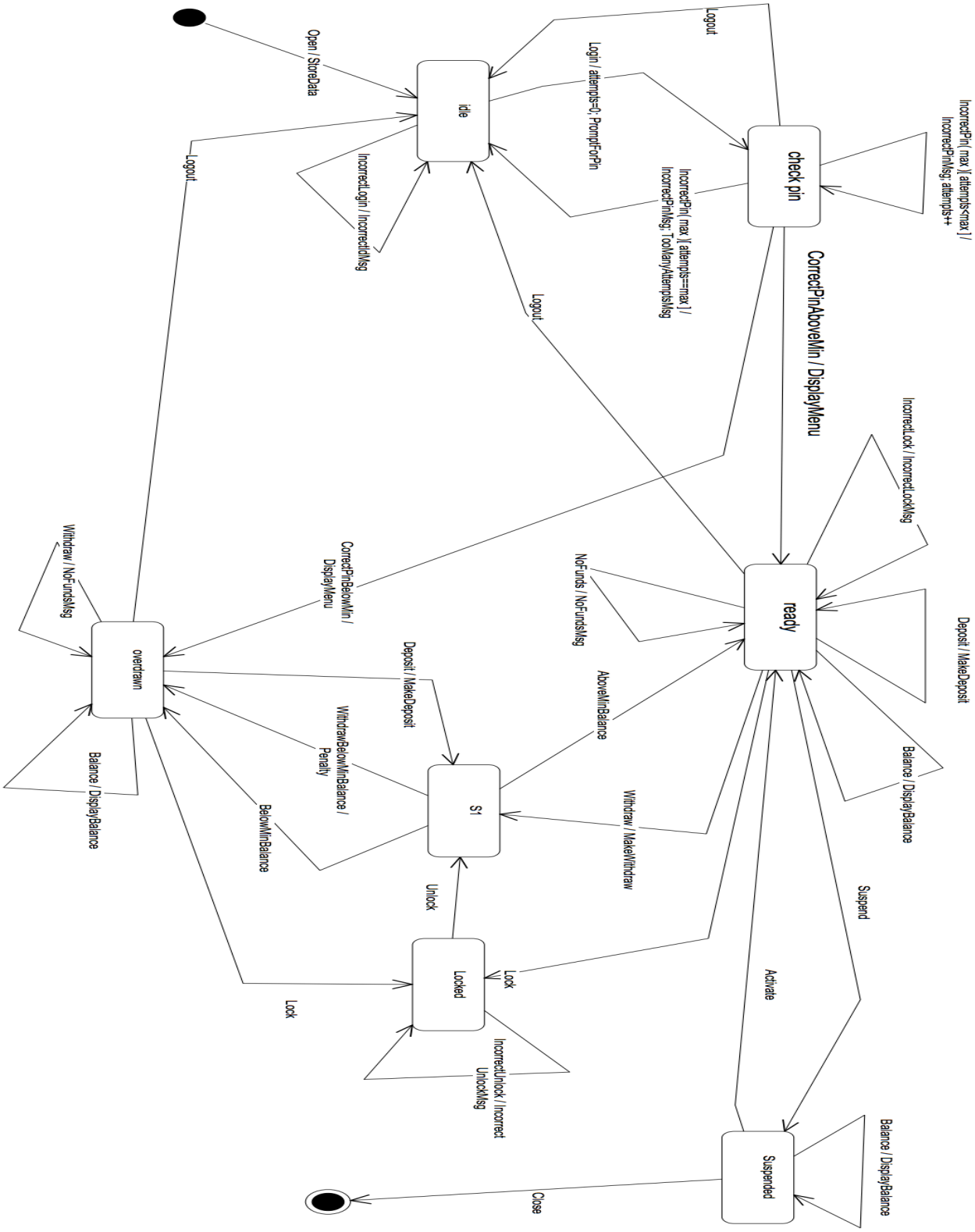
Operations of the Input Processor (Account-2)

```
create() {m->create();}
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

4. State Diagram of MDA-EFSM



1.3 Class Diagram of the MDA-ATM components:

Below are 6 cumulative class diagrams.

1. High-Level Class Diagram.
2. First Class Diagram showing the associations of
   (a) Account1, Account2
   (b) DataStore.
   (c) MDA_EFSM and State – *This models the State Pattern*.
3. Third Class Diagram showing the associations of
   (a) Output
   (b) Strategy Classes :: *Incorrect_Pin_Msg, Incorrect_ID_Msg, Store_Pin, Store_ID*
   (c) Abstract and Concrete Factory Classes – This models the Abstract and Strategy Pattern.
4. Fourth Class Diagram showing the associations of
   (d) Output
   (e) Strategy Classes:: *Too_Many_Attempts_Msg,Store_Balance,Display_Balance, Display_Menu.*
   (f) Abstract and Concrete Factory Classes – This models the Abstract and Strategy Pattern.
5. Fifth Class Diagram showing the associations of
   (g) Output
   (h) Strategy Classes :: *Prompt_For_PIN, Make_Deposit, Make_Withdraw, Penalty.*
   (i) Abstract and Concrete Factory Classes – This models the Abstract and Strategy Pattern.
6. Sixth Class Diagram showing the associations of
   (j) Output
   (k) Strategy Classes :: *Incorrect_Lock_Msg, Incorrect_Unlock_Msg, No_Funds_Msg*.
   (l) Abstract and Concrete Factory Classes – This models the Abstract and Strategy Pattern.

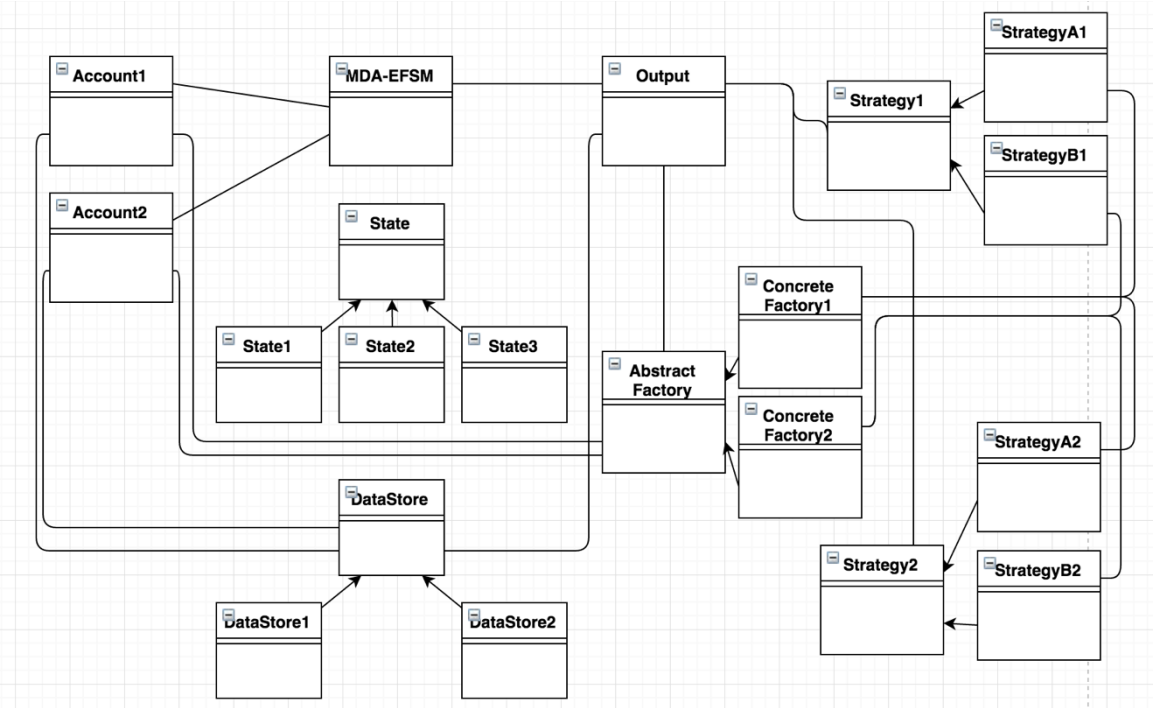


Figure 1:: High Level Class Diagram.

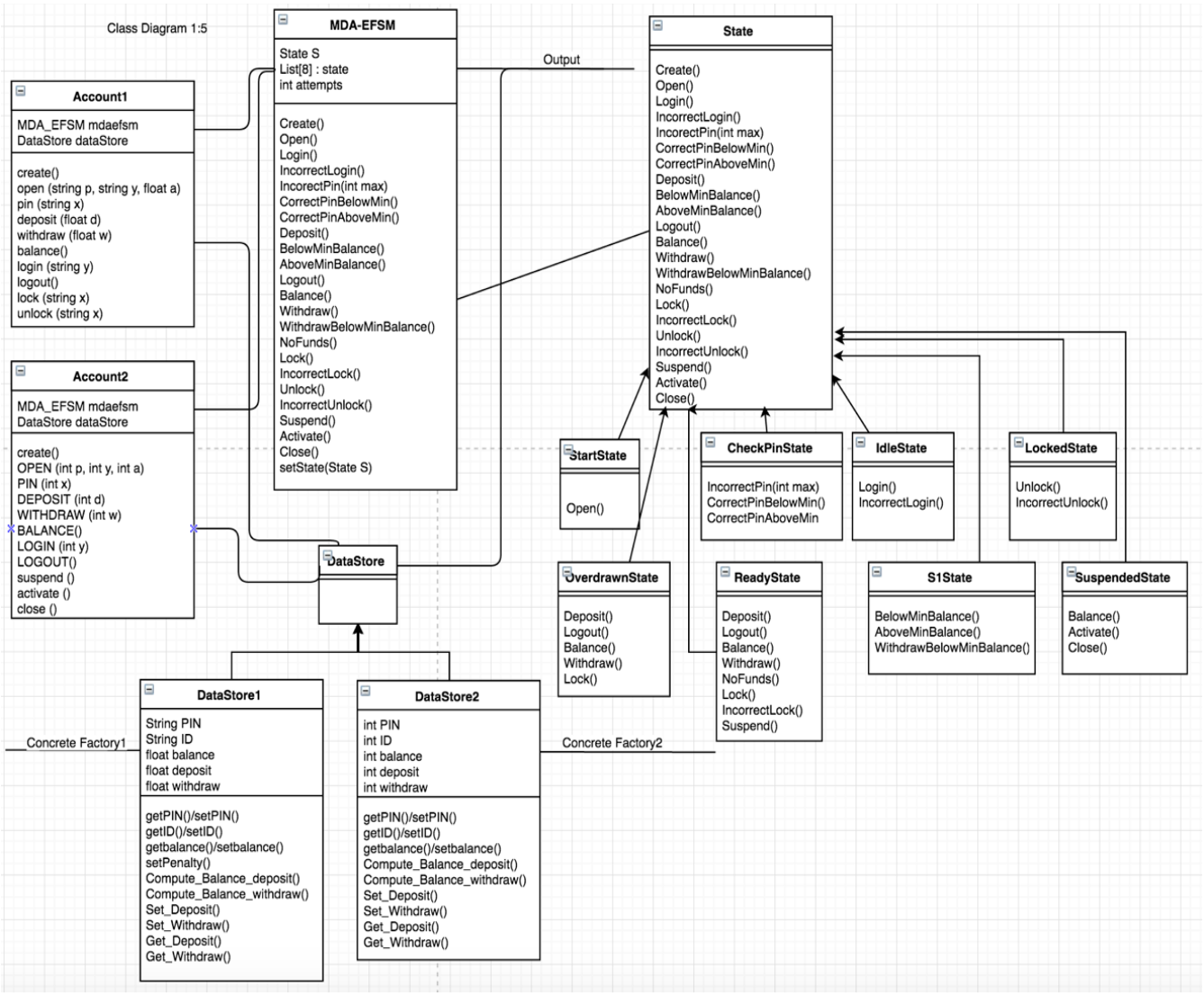


Figure 2:: a) Account– Input Processor , Platform dependent. b)MDA-EFSM—Platform Independent, c) DataStore – Storage for all 3 input processors, d) State – To Manage Events and actions. Depicting STATE Pattern

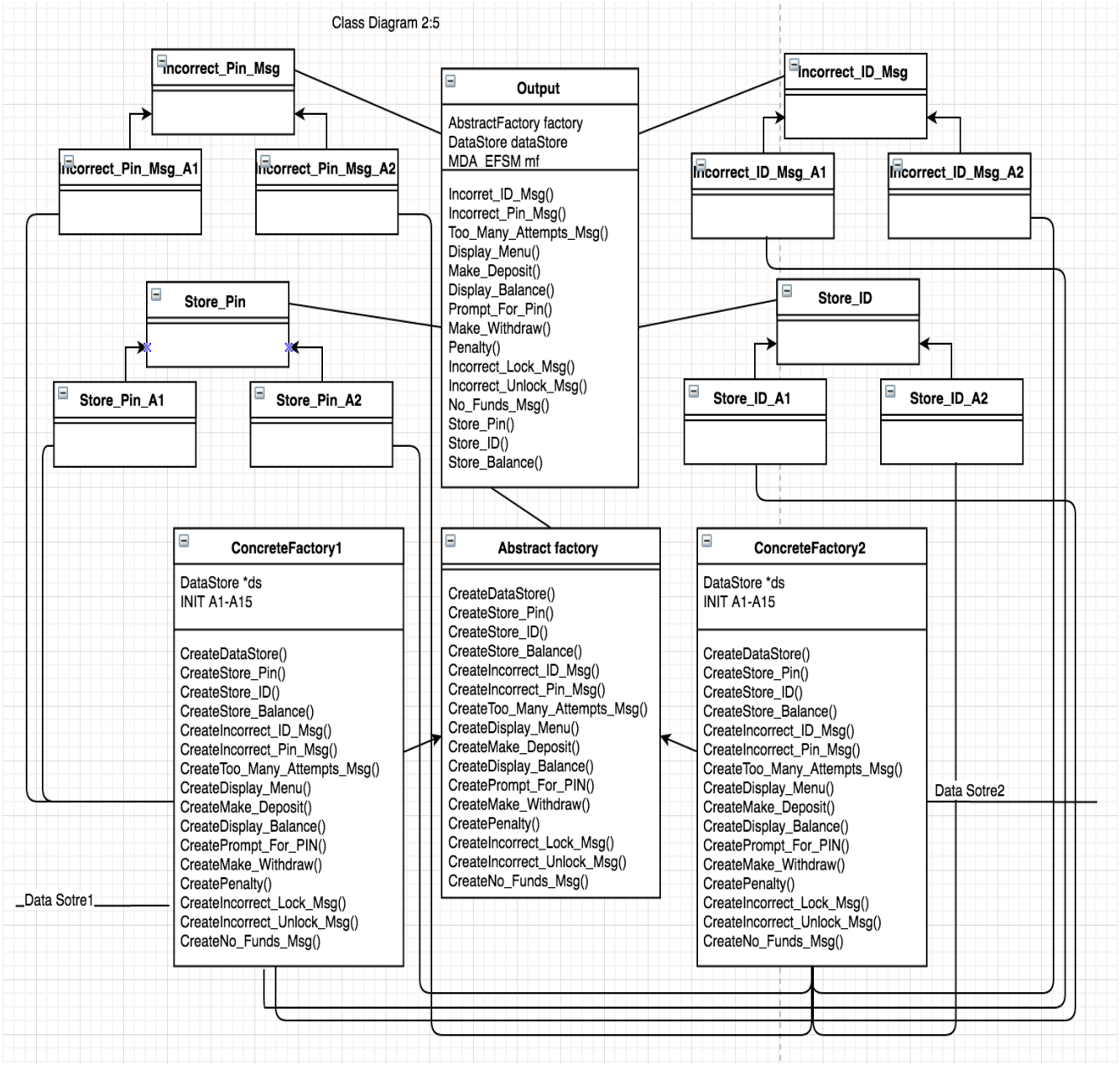


Figure 3:: Output Actions, depicting Strategy and Abstract Factory Pattern

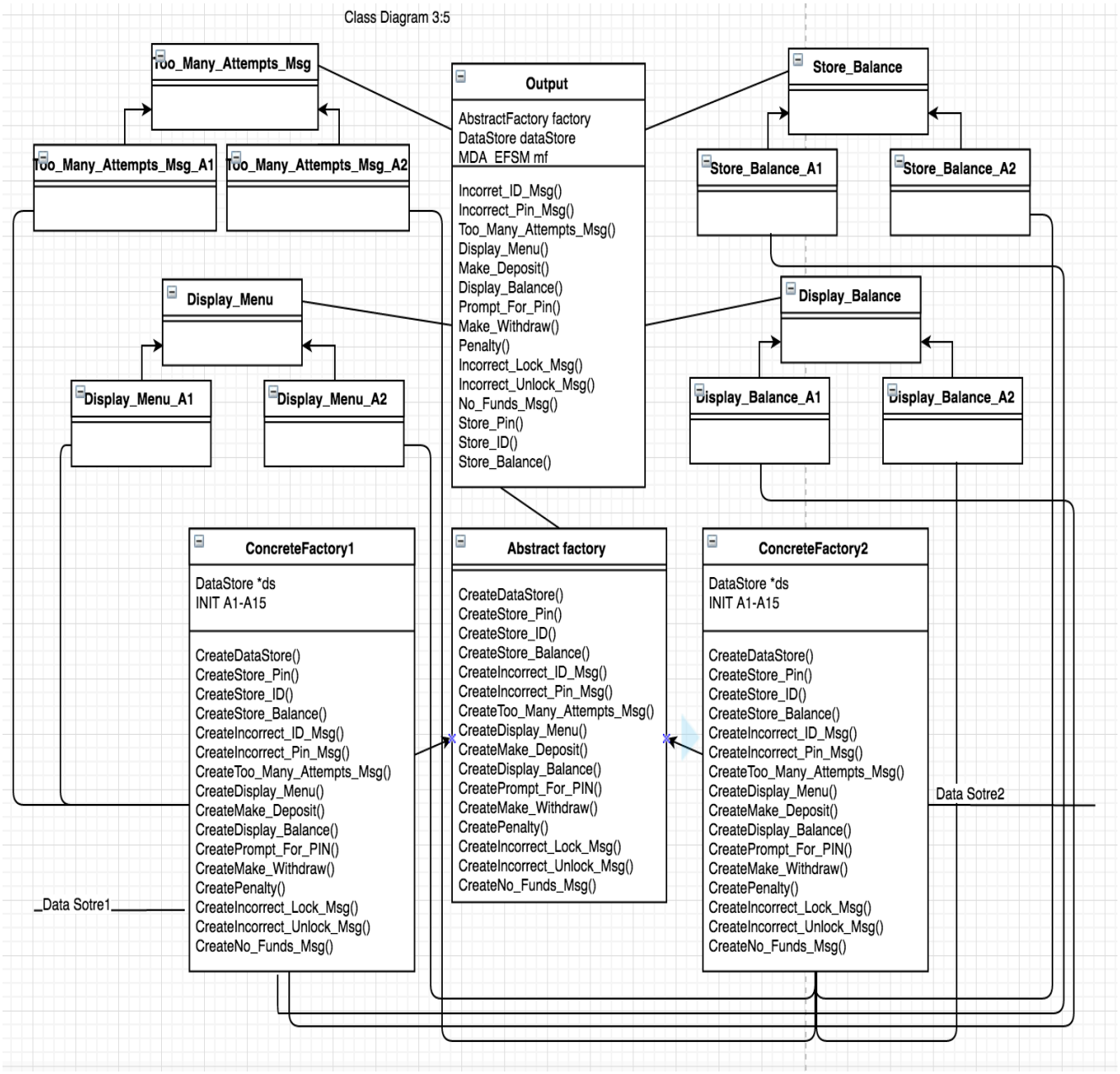


Figure 3(continue):: Output Actions, depicting Strategy and Abstract Factory Pattern

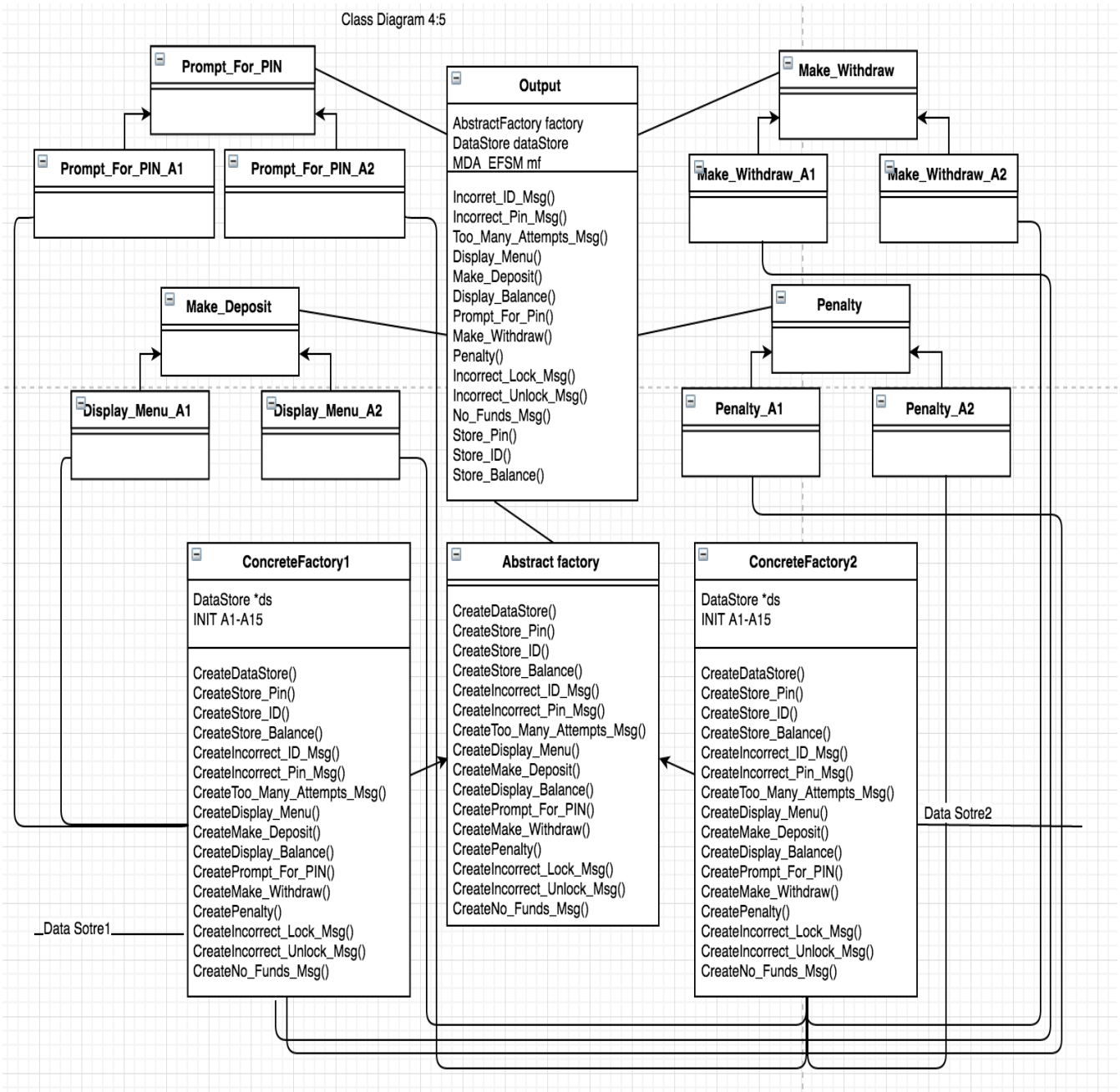


Figure 3(continue):: Output Actions, depicting Strategy and Abstract Factory Pattern

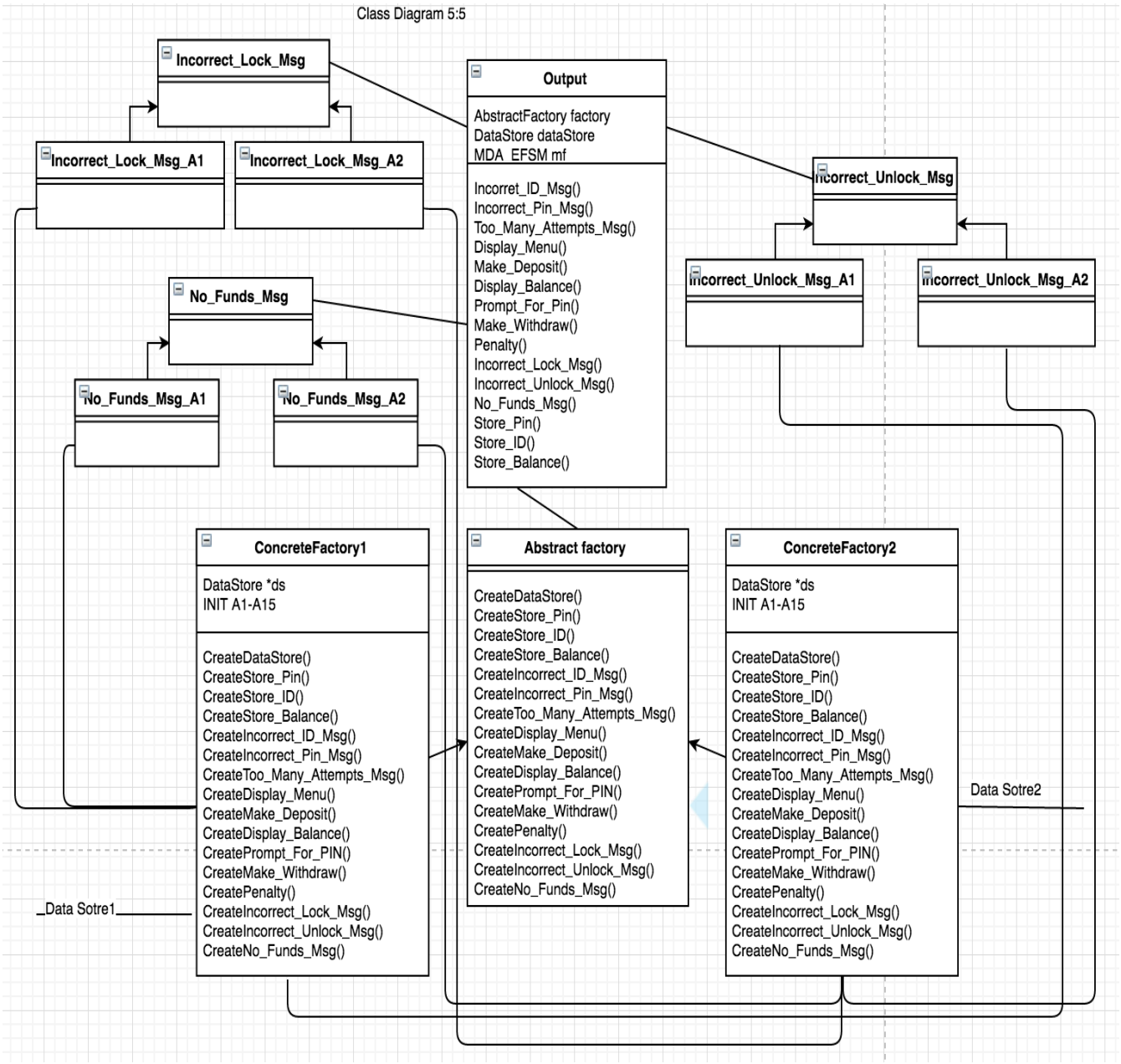


Figure 3(continue):: Output Actions, depicting Strategy and Abstract Factory Pattern

2 Operations, Attributes, Source Code and Patterns:

For Input Classes, please refer to section 1.2.3

2.1 Data Store Class::

Purpose:
Store data needed by ATM for storing temporary storage and Output class by Abstract Factory and Stragey Pattern for Computing and Storing permanent data.
Collaborators::
Account1, Account2, Output, ConcreteFactory1, ConcreteFactory2, Strategy Classes for Implementing Actions.
Attributes::

DataStore1::

```
/* Temporary Storage variables */
public String temp_PIN;
public String temp_ID;
public float temp_balance;
public float temp_deposit;
public float temp_withdraw;


/* Permanent Storage Variables ******/
public String PIN;// PIN String p
public String ID;// ID String y
public float balance; // a is the Balance
public float deposit; // Deposit variable
public float withdraw; // Withdraw variable
```

DataStore2::

```
/* Temporary Storage variables */
public int temp_PIN;
public int temp_ID;
public int temp_balance;
public int temp_deposit;
public int temp_withdraw;

/* Permanent Storage Variables ******/
public int PIN;// PIN String p
public int ID;// ID String y
public int balance; // a is the Balance
public int deposit; // Deposit variable
public int withdraw; // Withdraw variable
```

DataStore1

```
public float getbalance()
{
	return this.balance;
}
public String getPIN()
{
	return this.PIN;
}
public String getID()
{
	return this.ID;
}
public float setbalance()
{
	return this.balance = this.temp_balance;
}
public String setPIN()
{
	return this.PIN = this.temp_PIN;
}
public String setID()
{
	return this.ID = this.temp_ID;
}
public float setPenalty()
{
	this.balance = this.balance - 20;
	this.temp_balance = balance;
	return this.balance;
}
public void Compute_Balance_deposit()
{
	this.balance = this.balance +
this.deposit;
	this.temp_balance = this.balance;
}
public void Compute_Balance_withdraw()
{
	this.balance = this.balance -
this.withdraw;
	this.temp_balance = this.balance;
}
public void Set_Deposit()
{
	this.deposit = this.temp_deposit;
}
public void Set_Withdraw()
{
this.withdraw = this.temp_withdraw;
}
public float Get_Deposit()
{
	return this.deposit;
}
```

DataStore2

```
public int getbalance()
{
	return balance;
}
public int getPIN()
{
	return PIN;
}
public int getID()
{
	return ID;
}
public int setbalance()
{
	return this.balance =
this.temp_balance;
}
public int setPIN()
{
	return this.PIN = this.temp_PIN;
}
public int setID()
{
	return this.ID = this.temp_ID;
}
public void Compute_Balance_deposit()
{
	this.balance = this.balance +
this.deposit;
	this.temp_balance = this.balance;
}
public void Compute_Balance_withdraw()
{
	this.balance = this.balance -
this.withdraw;
	this.temp_balance = this.balance;
}
public void Set_Deposit()
{
	this.deposit = this.temp_deposit;
}
public void Set_Withdraw()
{
this.withdraw = this.temp_withdraw;
}
public float Get_Deposit()
{
	return this.deposit;
}
public float Get_Withdraw()
{
	return this.withdraw;
}
```

```
public float Get_Withdraw()
{
	return this.withdraw;
}
```

2.2 MDA-EFSM Class::

Purpose:
Manages state machine for State pattern. Should be independent of different types of atm machines.
Responsibilities:
Captures platform independent behavior.
Collaborators::
Account1,Account2,Output Attributes::

```
State list[8];     //Pointer to all states
State efsmState; //Pointer to current state public int attempts;
AbstractFactory factory =null; // initializes Abstract Factory Output output = null;
// initializes output
```

Psuedocode::

MDA-EFSM

```
public MDAEFSM(AbstractFactory factory,Output output) {
efsmState = startState;
attempts = 0;
this.factory = factory;
this.output = output;
}
public void create()
{
	efsmState.Create();
	printCurrentState();
}
public void Open()
{
	efsmState.Open();
	printCurrentState();
}
public void Login()
{
	efsmState.Login();
	attempts = 0;
	printCurrentState();
}
public void IncorrectLogin()
{
	efsmState.IncorrectLogin();
	printCurrentState();
}
public void IncorrectPin(int max)
{
	efsmState.IncorrectPin(max);
	printCurrentState();
}
public void CorrectPinBelowMin()
{
	efsmState.CorrectPinBelowMin();
	printCurrentState();
}
public void CorrectPinAboveMin()
{
	efsmState.CorrectPinAboveMin();
	printCurrentState();
```

```
}
public void Deposit()
{
	efsmState.Deposit();
	printCurrentState();
}
public void BelowMinBalance()
{
	efsmState.BelowMinBalance();
	printCurrentState();
}
public void AboveMinBalance()
{
	efsmState.AboveMinBalance();
	printCurrentState();
}
public void Logout()
{
	efsmState.Logout();
	printCurrentState();
}
public void Balance()
{
	efsmState.Balance();
	printCurrentState();
}
public void Withdraw()
{
	efsmState.Withdraw();
	printCurrentState();
}
public void WithdrawBelowMinBalance()
{
	efsmState.WithdrawBelowMinBalance();
	printCurrentState();
}
public void NoFunds()
{
	efsmState.NoFunds();
	printCurrentState();
}
public void Lock()
{
	efsmState.Lock();
	printCurrentState();
}
public void IncorrectLock()
{
	efsmState.IncorrectLock();
	printCurrentState();
}
public void Unlock()
{
	efsmState.Unlock();
	printCurrentState();
}
```

```
public void IncorrectUnlock()
{
	efsmState.IncorrectUnlock();
	printCurrentState();
}
public void Suspend()
{
	efsmState.Suspend();
	printCurrentState();
}
public void Activate()
{
	efsmState.Activate();
	printCurrentState();
}
public void Close()
{
	efsmState.Close();
	printCurrentState();
}
public void setState(State efsmState)
{
this.efsmState = efsmState;
}
public State getMachineState() {
return efsmState;
}
public void printCurrentState(){
	System.out.println("--- Current State : "+
efsmState.getClass().getName()+"---");
}
```

2.3 State Class::

MDA_EFSM mdaefsm;//Pointer to MDA_EFSM

StartState CLASS

```
public void Open() {
	mdaefsm.output.Store_Pin();
	mdaefsm.output.Store_ID();
	mdaefsm.output.Store_Balance();
	mdaefsm.setState(mdaefsm.getIdleState());
}
```

IdleState CLASS

```
public void Login() {
	mdaefsm.attempts = 0;
	mdaefsm.output.Prompt_For_Pin();
	mdaefsm.setState(mdaefsm.getCheckPinSta
te());
}
public void IncorrectLogin() {
	mdaefsm.output.Incorrect_ID_Msg();
}
```

CheckPinState CLASS

```
public void IncorrectPin(int max)
{
if( mdaefsm.attempts < max )
{
	mdaefsm.attempts++;
	mdaefsm.output.Incorrect_Pin_Msg();}
else if( mdaefsm.attempts >= max )
{
	mdaefsm.output.Incorrect_Pin_Msg();
	mdaefsm.output.Too_Many_Attempts_Msg();
	mdaefsm.setState(mdaefsm.getIdleState());
}
}
public void CorrectPinBelowMin()
{
	mdaefsm.output.Display_Menu();
	mdaefsm.setState(mdaefsm.getOverdrawnState());
}
public void CorrectPinAboveMin()
{
	mdaefsm.output.Display_Menu();
	mdaefsm.setState(mdaefsm.getReadyState());
}
public void Logout() {
	mdaefsm.setState(mdaefsm.getIdleState());
}
```

ReadyState CLASS

```
public void Deposit()
{
mdaefsm.output.Make_Deposit();
}
public void Logout()
{
mdaefsm.setState(mdaefsm.getIdleState());
}
public void Balance()
{
mdaefsm.output.Display_Balance();
}
public void Withdraw()
{
mdaefsm.output.Make_Withdraw();
mdaefsm.setState(mdaefsm.getS1State());
}
public void NoFunds() {
mdaefsm.output.No_Funds_Msg();}
public void Lock() {
mdaefsm.setState(mdaefsm.getLockedState());
}
public void IncorrectLock() {
mdaefsm.output.Incorrect_Lock_Msg();
}
public void Suspend() {
mdaefsm.setState(mdaefsm.getSuspendedState());
}
```

LockedState CLASS

```
public void Unlock() {
System.out.println("\n MDA_EFSM::LockedState::Unlock
function ");
mdaefsm.setState(mdaefsm.getS1State());
}
public void IncorrectUnlock() {
System.out.println("\n
MDA_EFSM::LockedState::IncorrectLock function ");
mdaefsm.output.Incorrect_Unlock_Msg();
}
```

OverdrawnState CLASS

```
public void Deposit() {
mdaefsm.output.Make_Deposit();
mdaefsm.setState(mdaefsm.getS1State());
}
public void Logout() {
mdaefsm.setState(mdaefsm.getIdleState());
}
public void Balance() {
mdaefsm.output.Display_Balance();
}
public void Withdraw() {
mdaefsm.output.No_Funds_Msg();
}
public void Lock() {
mdaefsm.setState(mdaefsm.getLockedState());
}
```

S1State CLASS

```
public void BelowMinBalance()
{
```

SuspendedState CLASS

```
public void Balance() {
mdaefsm.output.Display_Balance();
```

```
mdaefsm.setState(mdaefsm.getOverdrawnState());
}
public void AboveMinBalance()
{
mdaefsm.setState(mdaefsm.getReadyState());
}
public void WithdrawBelowMinBalance() {
mdaefsm.output.Penalty();
mdaefsm.setState(mdaefsm.getOverdrawnState());
}
```

```
}
public void Activate() {
mdaefsm.setState(mdaefsm.getReadyState());
}
public void Close() {
System.exit(0);
}
```

2.4 Output Class::

Purpose:
Calls Actions of the MDA-EFSM

Responsibilities:
Collaborators::
All strategy abstract classes, DataStore, Abstract Factory class.
Attributes::

```
AbstractFactory factory =null; // Pointer to appropriate concrete factory class
DataStore dataStore = null;   // Pointer to Data Store
```

Pseudocode::

```
public void Incorrect_ID_Msg()
{
	Incorrect_ID_Msg incorrect_id = factory.CreateIncorrect_ID_Msg();
	incorrect_id.Incorrect_ID_Msg();
}
public void Incorrect_Pin_Msg()
{
	Incorrect_Pin_Msg incorrect_pin = factory.CreateIncorrect_Pin_Msg();
	incorrect_pin.Incorrect_Pin_Msg();
}
public void Too_Many_Attempts_Msg()
{
	Too_Many_Attempts_Msg too_many_attempts = factory.CreateToo_Many_Attempts_Msg();
	too_many_attempts.Too_Many_Attempts_Msg();
}
public void Display_Menu()
{
	Display_Menu disp_menu = factory.CreateDisplay_Menu();
	disp_menu.Display_Menu();
}
public void Make_Deposit()
{
	Make_Deposit make_deposit = factory.CreateMake_Deposit();
	make_deposit.Make_Deposit(dataStore);
}
public void Display_Balance()
{
	Display_Balance disp_bal = factory.CreateDisplay_Balance();
	disp_bal.Display_Balance(dataStore);
}
public void Prompt_For_Pin()
{
	Prompt_For_PIN prompt_pin = factory.CreatePrompt_For_PIN();
	prompt_pin.Prompt_For_PIN();
}
public void Make_Withdraw()
{
	Make_Withdraw make_withdraw = factory.CreateMake_Withdraw();
	make_withdraw.Make_Withdraw(dataStore);
}
public void Penalty()
{
	Penalty penalty = factory.CreatePenalty();
	penalty.Penalty(dataStore);
}
public void Incorrect_Lock_Msg()
{
	Incorrect_Lock_Msg incorrect_lock = factory.CreateIncorrect_Lock_Msg();
```

```
        incorrect_lock.Incorrect_Lock_Msg();
}
public void Incorrect_Unlock_Msg()
{
        Incorrect_Unlock_Msg incorrect_unlock = factory.CreateIncorrect_Unlock_Msg();
        incorrect_unlock.Incorrect_Unlock_Msg();
}
public void No_Funds_Msg()
{
        No_Funds_Msg no_funds_msg = factory.CreateNo_Funds_Msg();
        no_funds_msg.No_Funds_Msg();
}
public void Store_Pin()
{
        Store_Pin store_pin = factory.CreateStore_Pin();
        store_pin.Store_Pin(dataStore);
}
public void Store_ID()
{
        Store_ID store_id = factory.CreateStore_ID();
        store_id.Store_ID(dataStore);
}
public void Store_Balance()
{
        Store_Balance store_bal = factory.CreateStore_Balance();
        store_bal.Store_Balance(dataStore);
}
```

2.5 Abstract and Concrete Factory Class::

Purpose:
To group strategies for a particular Acount. Responsibilities:
Collaborators::
All respective strategy classes needed by Account1, Account2, DataStores needed by Account1, Account2.
Attributes::
Pseudocode::

```
AbstractFactory CLASS
public interface AbstractFactory
{
public DataStore CreateDataStore();
public Store_Pin CreateStore_Pin();
public Store_ID CreateStore_ID();
public Store_Balance CreateStore_Balance();
public Incorrect_ID_Msg CreateIncorrect_ID_Msg();
public Incorrect_Pin_Msg CreateIncorrect_Pin_Msg();
public Too_Many_Attempts_Msg CreateToo_Many_Attempts_Msg();
public Display_Menu CreateDisplay_Menu();
public Make_Deposit CreateMake_Deposit();
public Display_Balance CreateDisplay_Balance();
public Prompt_For_PIN CreatePrompt_For_PIN();
public Make_Withdraw CreateMake_Withdraw();
public Penalty CreatePenalty();
public Incorrect_Lock_Msg CreateIncorrect_Lock_Msg();
public Incorrect_Unlock_Msg CreateIncorrect_Unlock_Msg();
public No_Funds_Msg CreateNo_Funds_Msg();
```

```
}
```

ConcreteFactory1 CLASS

```
public DataStore CreateDataStore()
{
	return(this.dataStore);
}
public DataStore GetDataStore()
{
	return this.dataStore;
}
public Incorrect_Pin_Msg
CreateIncorrect_Pin_Msg()
{
	return this.incorrect_pin;
}
public Too_Many_Attempts_Msg
CreateToo_Many_Attempts_Msg()
{
	return this.too_many_attempts_msg;
}
public Display_Menu CreateDisplay_Menu()
{
	return this.disp_menu;
}
public Store_Pin CreateStore_Pin()
{
	return this.store_pin;
}
public Store_Balance CreateStore_Balance()
{
	return this.store_bal;
}
public Prompt_For_PIN CreatePrompt_For_PIN()
{
	return this.prompt_pin;
}
public Display_Balance CreateDisplay_Balance()
{
	return this.disp_bal;
}
public Make_Deposit CreateMake_Deposit()
{
	return this.make_deposit;
}
public Make_Withdraw CreateMake_Withdraw()
{
	return this.make_withdraw;
}
public Penalty CreatePenalty()

	return this.penalty;
}
public Store_ID CreateStore_ID() {
	return this.store_id;
}
```

ConcreteFactory2 CLASS

```
public DataStore CreateDataStore()
{
	return(this.dataStore);
}
public DataStore GetDataStore()
{
	return this.dataStore;
}
public Incorrect_Pin_Msg
CreateIncorrect_Pin_Msg()
{
	return this.incorrect_pin;
}
public Too_Many_Attempts_Msg
CreateToo_Many_Attempts_Msg()
{
	return this.too_many_attempts_msg;
}
public Display_Menu CreateDisplay_Menu()
{
	return this.disp_menu;
}
public Store_Pin CreateStore_Pin()
{
	return this.store_pin;
}
public Store_Balance CreateStore_Balance()
{
	return this.store_bal;
}
public Prompt_For_PIN CreatePrompt_For_PIN()
{
	return this.prompt_pin;
}
public Display_Balance CreateDisplay_Balance()
{
	return this.disp_bal;
}
public Make_Deposit CreateMake_Deposit()
{
	return this.make_deposit;
}
public Make_Withdraw CreateMake_Withdraw()
{
	return this.make_withdraw;
}
public Penalty CreatePenalty()
{
	return this.penalty;
}
public Store_ID CreateStore_ID() {
	return this.store_id;
}
```

```
public Incorrect_ID_Msg
CreateIncorrect_ID_Msg() {
	return this.incorrect_id;
}
public Incorrect_Lock_Msg
CreateIncorrect_Lock_Msg() {
	return this.incorrect_lock;
}
public Incorrect_Unlock_Msg
CreateIncorrect_Unlock_Msg() {
	return this.incorrect_unlock;
}
public No_Funds_Msg CreateNo_Funds_Msg() {
	return this.no_funds;
}
```

```
public Incorrect_ID_Msg
CreateIncorrect_ID_Msg() {
	return this.incorrect_id;
}
public Incorrect_Lock_Msg
CreateIncorrect_Lock_Msg() {
	return this.incorrect_lock;
}
public Incorrect_Unlock_Msg
CreateIncorrect_Unlock_Msg() {
	return this.incorrect_unlock;
}
public No_Funds_Msg CreateNo_Funds_Msg() {
	return this.no_funds;
}
```

2.6 Strategy Classes::

Purpose:
Implement different functionalities through Actions.
Responsibilities:
Collaborators::
Concrete Factory Classes.
Attributes::
Pseudocode::

Display_Balance STRATEGY CLASS	
Display_Balance_A1	Display_Balance_A2

public void Display_Balance(DataStore dataStore)
{
System.out.println("Account 1:: Balance is " + ((DataStore1)dataStore).getbalance());
}
	public void Display_Balance(DataStore dataStore)
	{
	System.out.println("Account 2:: Balance is " + ((DataStore2)dataStore).getbalance());
	}

Display_Menu STRATEGY CLASS	
Display_Menu_A1	Display_Menu_A2

public void Display_Menu()
{
System.out.println("Account 1:: Transaction Menu ");
System.out.println(" 1:: Balance ");
System.out.println(" 2:: Deposit ");
System.out.println(" 3:: Withdraw ");
System.out.println(" 4:: Lock ");
System.out.println(" 5:: Logout ");
}
	public void Display_Menu()
	{
	System.out.println("Account 2:: Transaction Menu ");
	System.out.println(" 1:: Balance ");
	System.out.println(" 2:: Deposit ");
	System.out.println(" 3:: Withdraw ");
	System.out.println(" 4:: Suspend ");
	System.out.println(" 5:: Logout ");
	}

Incorrect_ID_Msg STRATEGY CLASS	
Incorrect_ID_Msg_A1	Incorrect_ID_Msg_A2

public void Incorrect_ID_Msg()
{
System.out.println("Account 1:: Incorrect ID ");
}
	public void Incorrect_ID_Msg()
	{
	System.out.println("Account 2:: Incorrect ID ");
	}

Incorrect_Lock_Msg STRATEGY CLASS	

Incorrect_Lock_Msg_A1	Incorrect_Lock_Msg_A2

public void Incorrect_Lock_Msg()
{
System.out.println("Account 1:: Incorrect Lock ");
}
	public void Incorrect_Lock_Msg()
	{
	System.out.println("Account 2:: Incorrect Lock ");
	}

Incorrect_Pin_Msg STRATEGY CLASS	
Incorrect_Pin_Msg_A1	Incorrect_Pin_Msg_A2

public void Incorrect_Pin_Msg()
{
System.out.println("Account 1:: Incorrect Pin ");
}
	public void Incorrect_Pin_Msg()
	{
	System.out.println("Account 2:: Incorrect Pin ");
	}

Incorrect_Unlock_Msg STRATEGY CLASS	
Incorrect_Unlock_Msg_A1	Incorrect_Unlock_Msg_A2

public void Incorrect_Unlock_Msg()
{
System.out.println("Account 1:: Incorrect Unlock ");
}
	public void Incorrect_Unlock_Msg()
	{
	System.out.println("Account 2:: Incorrect Unlock ");
	}

Make_Deposit STRATEGY CLASS	
Make_Deposit_A1	Make_Deposit_A2

public void Make_Deposit(DataStore dataStore)
{
((DataStore1)dataStore).Set_Deposit();
((DataStore1)dataStore).Compute_Balance_deposit();
System.out.println("Account 1:: After Deposit, Balance is " + ((DataStore1)dataStore).getbalance());
}
	public void Make_Deposit(DataStore dataStore)
	{
	((DataStore2)dataStore).Set_Deposit();
	((DataStore2)dataStore).Compute_Balance_deposit();
	System.out.println("Account 2:: After Deposit, Balance is " + ((DataStore2)dataStore).getbalance());
	}

Make_Withdraw STRATEGY CLASS	
Make_Withdraw_A1	Make_Withdraw_A2

public void Make_Withdraw(DataStore dataStore)
{
((DataStore1)dataStore).Set_Withdraw();
((DataStore1)dataStore).Compute_Balance_withdraw();
System.out.println("Account 1:: After Withdraw, Balance is " + ((DataStore1)dataStore).getbalance());
}
	public void Make_Withdraw(DataStore dataStore)
	{
	((DataStore2)dataStore).Set_Withdraw();
	((DataStore2)dataStore).Compute_Balance_withdraw();
	System.out.println("Account 2:: After Withdraw, Balance is " + ((DataStore2)dataStore).getbalance());
	}

No_Funds_Msg STRATEGY CLASS	
No_Funds_Msg_A1	No_Funds_Msg_A2

public void No_Funds_Msg()
{
System.out.println("Account 1:: Below mininum balance ");
}
	public void No_Funds_Msg()
	{
	System.out.println("Account 2:: No Funds ");
	}

Penalty STRATEGY CLASS	
Penalty_A1	Penalty_A2

public void Penalty(DataStore dataStore)
{
((DataStore1)dataStore).setPenalty();
System.out.println("Account 1:: Minimum required balance is $500. So Penalty is applied.");
	public void Penalty(DataStore dataStore)
	{
	System.out.println("Account 2:: Minimum required balance is $0.But no Penalty is applied.");
	}

System.out.println("Account 1:: After a Penalty of 20$, Balance is " + ((DataStore1)dataStore).balance); }	
Prompt_For_PIN STRATEGY CLASS	
Prompt_For_PIN_A1	Prompt_For_PIN_A2

public void Prompt_For_PIN()
{
System.out.println("Account 1:: Enter the Pin:: ");
}
	public void Prompt_For_PIN()
	{
	System.out.println("Account 2:: Enter the Pin:: ");
	}

Store_Balance STRATEGY CLASS	
Store_Balance_A1	Store_Balance_A2

public void Store_Balance(DataStore dataStore)
{
((DataStore1)dataStore).setbalance();
System.out.println("Account1:: After Storing the Balance, Balance is " + ((DataStore1)dataStore).getbalance());
}
	public void Store_Balance(DataStore dataStore)
	{
	((DataStore2)dataStore).setbalance();
	System.out.println("Account2:: After Storing the Balance, Balance is " + ((DataStore2)dataStore).getbalance());
	}

Store_ID STRATEGY CLASS	
Store_ID_A1	Store_ID_A2

public void Store_ID(DataStore dataStore)
{
((DataStore1)dataStore).setID();
System.out.println("Account1:: After Storing the ID, ID is " + ((DataStore1)dataStore).ID);
} }
	public void Store_ID(DataStore dataStore)
	{
	((DataStore2)dataStore).setID();
	System.out.println("Account2:: After Storing the ID, ID is " + ((DataStore2)dataStore).ID);
	}

Store_Pin STRATEGY CLASS	
Store_Pin_A1	Store_Pin_A2

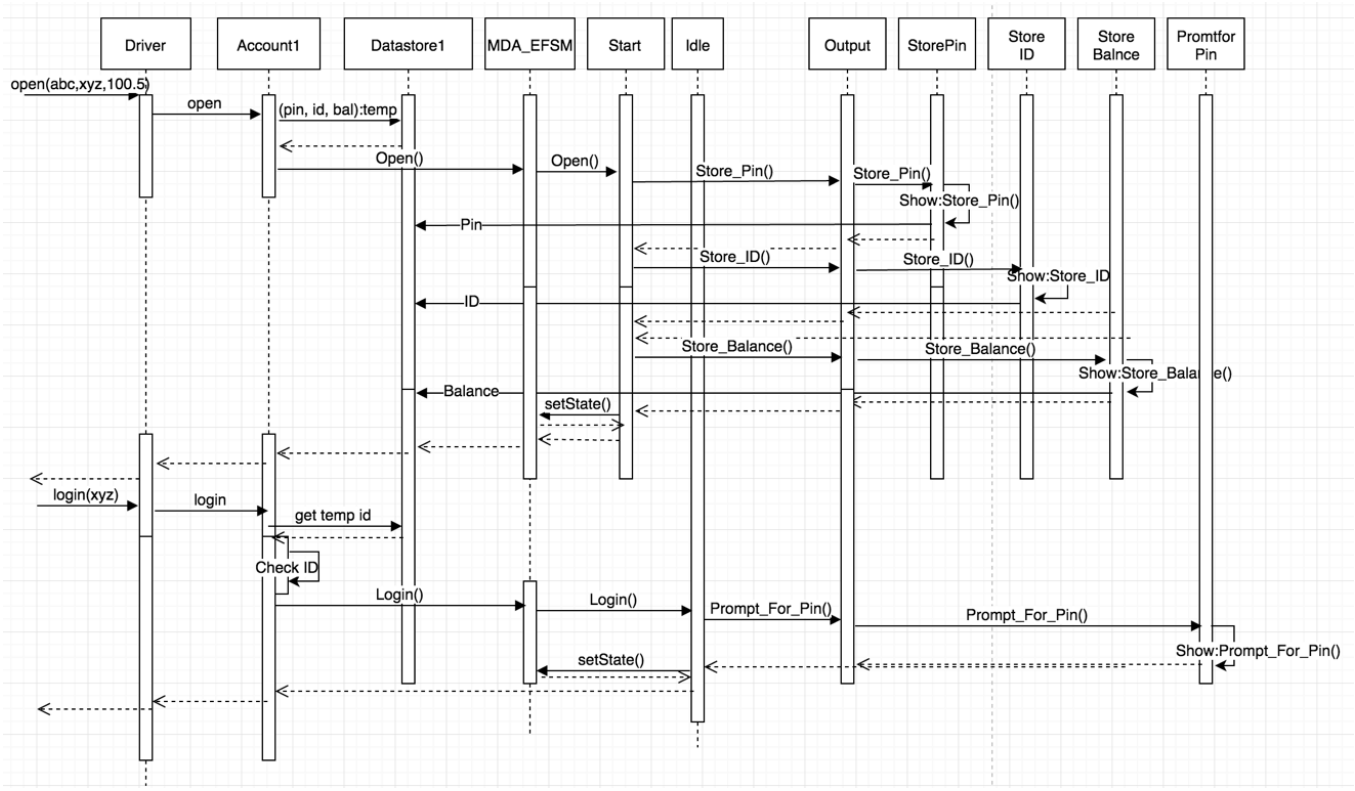
public void Store_Pin(DataStore dataStore)
{
((DataStore1)dataStore).setPIN();
System.out.println("Account1:: After Storing the PIN, PIN is " + ((DataStore1)dataStore).PIN);
}
	public void Store_Pin(DataStore dataStore)
	{
	((DataStore2)dataStore).setPIN();
	System.out.println("Account2:: After Storing the PIN, PIN is " + ((DataStore2)dataStore).PIN);
	}

Too_Many_Attempts_Msg STRATEGY CLASS	
Too_Many_Attempts_Msg_A1	Too_Many_Attempts_Msg_A2

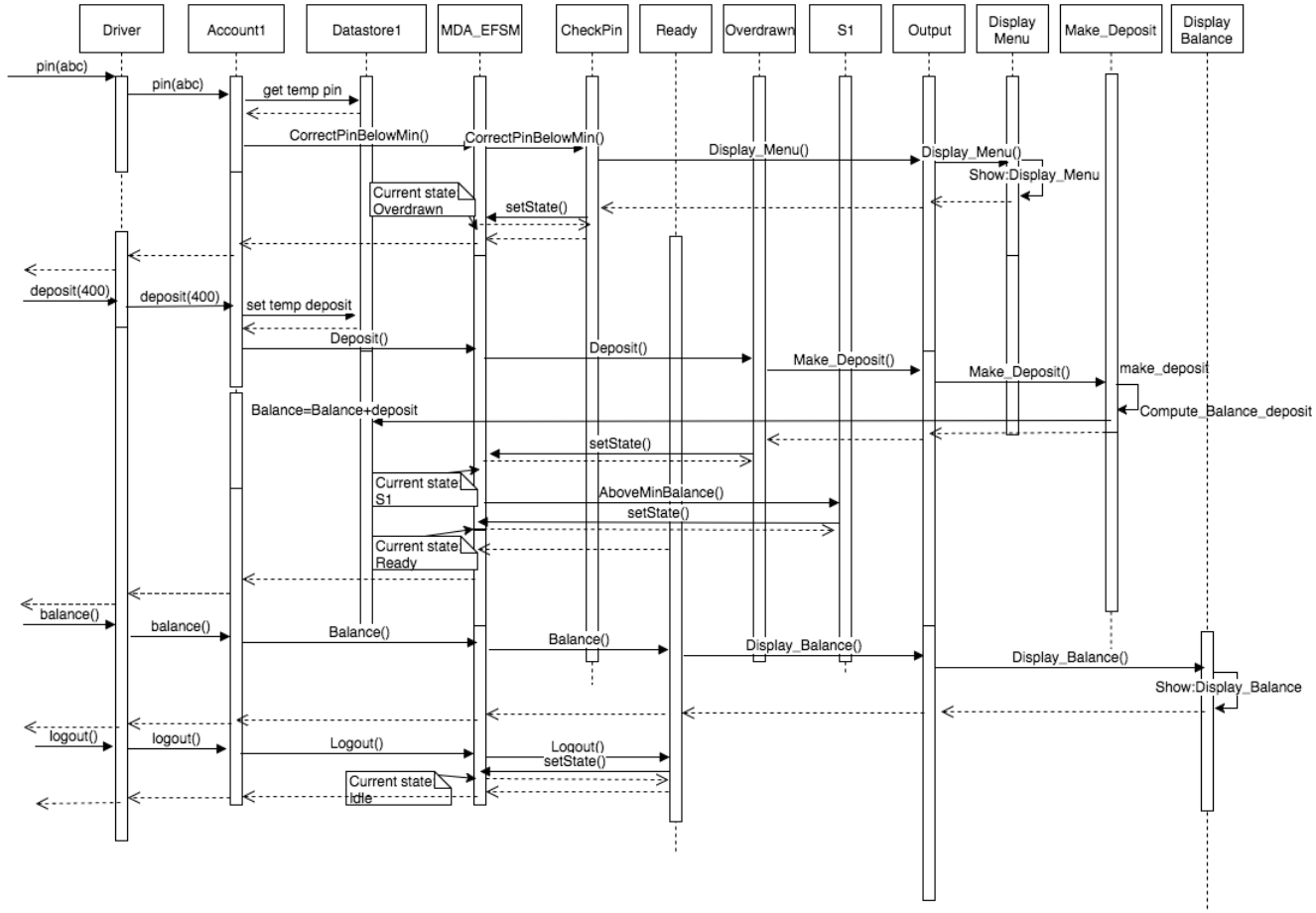
public void Too_Many_Attempts_Msg()
{
System.out.println("Account 1:: Too Many Attempts ");
}
	public void Too_Many_Attempts_Msg()
	{
	System.out.println("Account 2:: Too Many Attempts ");
	}

3 Sequence Diagrams::

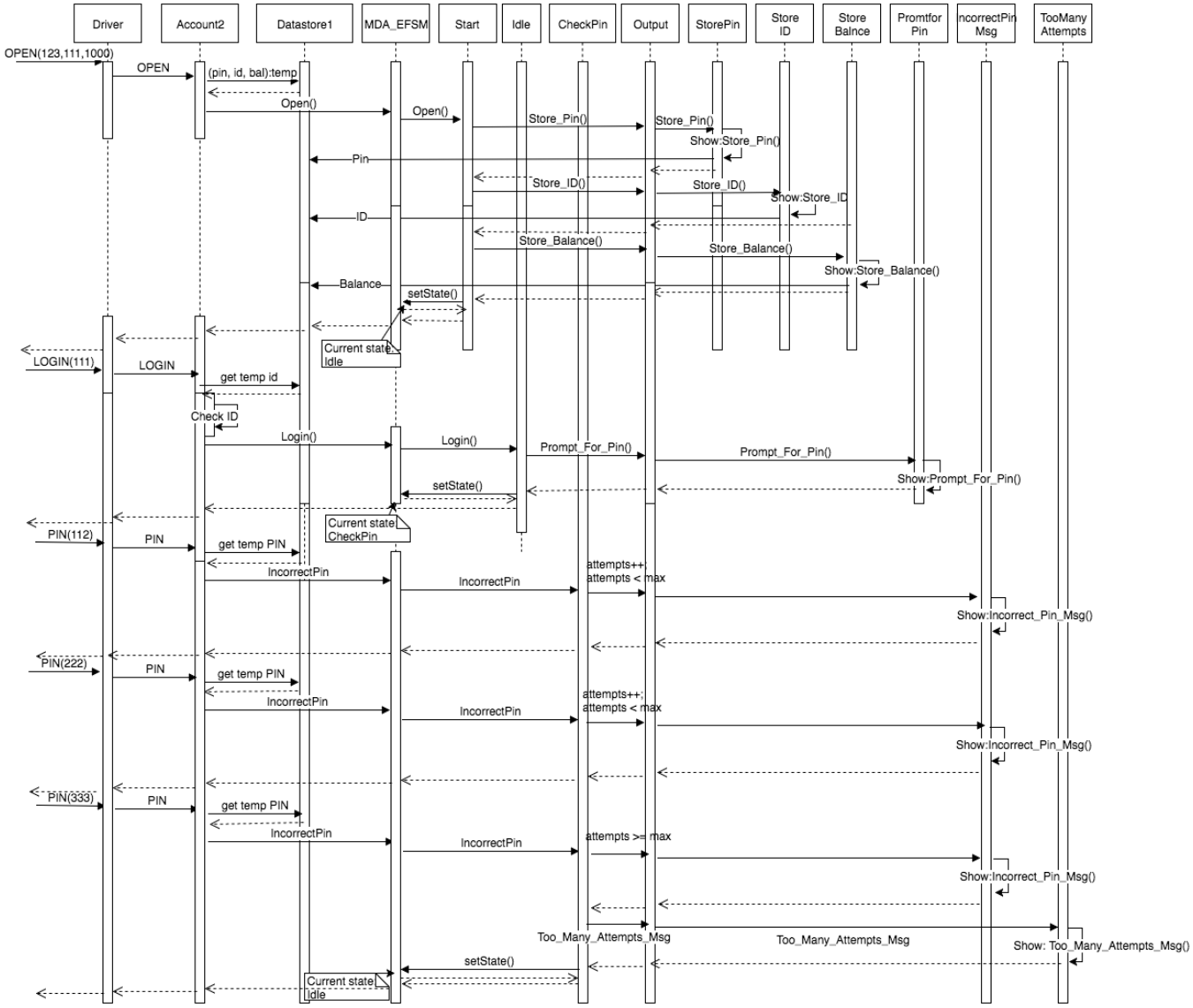
3.1 Scenario 1:: open(abc,xyz,100.5), login(xyz)::


Driver
Account1
Datastore1
MDA_EFSM
Start
Idle
Output
StorePin
Store ID
Store Balnce
Promtfor Pin
open(abc,xyz,100.5)
open
(pin, id, bal):temp
Open()
Open()
Store_Pin()
Store_Pin()
Show:Store_Pin()
Pin
Store_ID()
Store_ID()
Show:Store_ID
ID
Store_Balance()
Store_Balance()
Balance
setState()
login(xyz)
login
get temp id
Check ID
Login()
Login()
Prompt_For_Pin()
Prompt_For_Pin()
Show:Prompt_For_Pin()
setState()

3.2 Scenario 1:: pin(abc),deposit(400), balance(), logout() ::



3.3 Scenario 2:: OPEN(123,111,1000),LOGIN(111), PIN(112), PIN(222), PIN(333) ::



4 Source Code::

4.1 Driver.java

Main Interface File for the User

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

import mda_efsm.MDAEFSM;
import output.Output;
import abstract_factory.ConcreteFactory1;
import abstract_factory.ConcreteFactory2;
import account.Account1;
import account.Account2;
```

```
/*
 * CLASS : Driver ( Main function Program)
 */
public class Driver
{
public static void main( String[] args) throws IOException
{

	BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
	String input = null;

		System.out.println(" ******** Select Account *******" );
		System.out.println("1. Account - 1" );
		System.out.println("2. Account - 2" );

		input = bufferedReader.readLine();
		if(input.equalsIgnoreCase("1"))
		{
		ConcreteFactory1 factory = new ConcreteFactory1();
		Output output = new Output(factory,factory.GetDataStore());
		MDAEFSM mdaefsm = new MDAEFSM(factory,output);
		Account1 a1 = new Account1(mdaefsm,factory.GetDataStore());

		float balance,deposit,withdraw;
		String PIN, ID;

		System.out.println("************************************");
	System.out.println("Account-1" );
	System.out.println("MENU of Operations" );
	System.out.println("0. open(string,string,float)" );
	System.out.println("1. pin(string)" );
	System.out.println("2. deposit(float)" );
	System.out.println("3. withdraw(float)" );
	System.out.println("4. balance()" );
	System.out.println("5. login(string)" );
	System.out.println("6. logout()" );
	System.out.println("7. lock(string)" );
	System.out.println("8. unlock(string)" );
	System.out.println("q. Quit the demo program" );
	System.out.println("Please make a note of these operations" );
	System.out.println(" Account-1 Execution" );

	while (true) {

		System.out.println("Select Operation: ");
		System.out.println("0-open,1-pin,2-deposit,3-withdraw,4-balance,5-login,6-
logout,7-lock,8-unlock,q-Quit");

				input = bufferedReader.readLine();

	if(input.isEmpty()) continue;
	if(input.charAt(0) == 'q')
		break;

	switch(input.charAt(0))
```

```
      {
                    case '0': //open
                                System.out.println("\nOperation:open (string p, string y,
float a)");
                                System.out.println(" Enter value of the parameter PIN:");
                                input = bufferedReader.readLine();
                                PIN = input;
                                System.out.println(" Enter value of the parameter ID:");
                                input = bufferedReader.readLine();
                                ID = input;
                                System.out.println(" Enter value of the parameter balance:");
                                input = bufferedReader.readLine();
                                balance = Float.parseFloat(input);
                                a1.open(PIN, ID, balance);
                                break;
                     case '1'://pin
                                System.out.println("Operation:pin(string x)");
                                System.out.println("Enter value of the parameter PIN:");
                                input = bufferedReader.readLine();
                                PIN = input;
                                a1.pin(PIN);
                                break;
                    case '2'://deposit
                                System.out.println("Operation:deposit(float d)");
                                System.out.println("Enter value of the parameter Deposit:");
                                input = bufferedReader.readLine();
                                deposit = Float.parseFloat(input);
                                a1.deposit(deposit);
                                break;
                    case '3':// withdraw
                                System.out.println("Operation:withdraw(Float w)");
                                System.out.println("Enter value of the parameter Withdraw:");
                                input = bufferedReader.readLine();
                                withdraw = Float.parseFloat(input);
                                a1.withdraw(withdraw);
                                break;
                    case '4':// balance
                                System.out.println("Operation:balance()");
                                a1.balance();
                                break;
                    case '5':// login
                                System.out.println("Operation:login(string x)");
                                System.out.println("Enter value of the parameter ID:");
                                input = bufferedReader.readLine();
                                ID = input;
                                a1.login(ID);
                                break;
                    case '6':// logout
                                System.out.println("Operation:logout");
                                a1.logout();
                                break;
                    case '7':// lock(string x)
                                System.out.println("Operation:lock");
                                System.out.println("Enter value of the parameter PIN:");
                                input = bufferedReader.readLine();
                                PIN = input;
```

```
                                a1.lock(PIN);
                                break;
                        case '8':// unlock(string x)
                                System.out.println("Operation:unlock");
                                System.out.println("Enter value of the parameter PIN:");
                                input = bufferedReader.readLine();
                                PIN = input;
                                a1.unlock(PIN);
                                break;
                        default:
                                System.out.println("Invalid Choice");
                                break;
        }
        }
        System.out.println("Thanks for using Account - 1" );
                }
                else if(input.equalsIgnoreCase("2"))
                {
                ConcreteFactory2 factory = new ConcreteFactory2();
                Output output = new Output(factory,factory.GetDataStore());
                MDAEFSM mdaefsm = new MDAEFSM(factory,output);
                Account2 atm = new Account2(mdaefsm,factory.GetDataStore());

                int balance,deposit,withdraw;
                int PIN, ID;

                System.out.println("************************************");
        System.out.println("Account-2" );
        System.out.println("MENU of Operations" );
        System.out.println("0. OPEN(int,int,int)" );
        System.out.println("1. PIN(string)" );
        System.out.println("2. DEPOSIT(float)" );
        System.out.println("3. WITHDRAW(float)" );
        System.out.println("4. BALANCE()" );
        System.out.println("5. LOGIN(string)" );
        System.out.println("6. LOGOUT()" );
        System.out.println("7. suspend()" );
        System.out.println("8. activate()" );
        System.out.println("9. close()" );
        System.out.println("q. Quit the demo program" );
        System.out.println("Please make a note of these operations" );
        System.out.println(" Account-2 Execution" );
        while (true)
        {

                System.out.println("Select Operation: ");
                System.out.println("0-OPEN,1-LOGIN,2-PIN,3-DEPOSIT,4-WITHDRAW,5-BALANCE,6-
LOGOUT,7-suspend,8-activate,9-close,q-Quit");

                                input = bufferedReader.readLine();

        if(input.isEmpty()) continue;
        if(input.charAt(0) == 'q')
                break;

        switch(input.charAt(0))
```

```
    {

                    case '0': //OPEN
                                System.out.println("\nOperation:OPEN(int p, int y, int a)");
                                System.out.println(" Enter value of the parameter p:");
                                input = bufferedReader.readLine();
                                PIN = Integer.parseInt(input);
                                System.out.println(" Enter value of the parameter y:");
                                input = bufferedReader.readLine();
                                ID = Integer.parseInt(input);
                                System.out.println(" Enter value of the parameter a:");
                                input = bufferedReader.readLine();
                                balance = Integer.parseInt(input);
                                atm.OPEN(PIN, ID, balance);
                                break;
                    case '1':     //LOGIN
                                        System.out.println("Operation:LOGIN(int y)");
                                        System.out.println("Enter value of the parameter y:");
                                        input = bufferedReader.readLine();
                                        ID = Integer.parseInt(input);
                                        atm.LOGIN(ID);
                                        break;
                    case '2':     //PIN
                                        System.out.println("Operation:PIN(String x)");
                                        System.out.println("Enter value of the parameter x:");
                                        input = bufferedReader.readLine();
                                PIN = Integer.parseInt(input);
                                        atm.PIN(PIN);
                                        break;
                                case '3':// DEPOSIT
                                        System.out.println("Operation:DEPOSIT(int d)");
                                        System.out.println("Enter value of the parameter d:");
                                        input = bufferedReader.readLine();
                                        deposit = Integer.parseInt(input);
                                        atm.DEPOSIT(deposit);
                                        break;
                    case '4':// WITHDRAW
                                System.out.println("Operation:WITHDRAW(int w)");
                                System.out.println("Enter value of the parameter w:");
                                input = bufferedReader.readLine();
                                withdraw = Integer.parseInt(input);
                                atm.WITHDRAW(withdraw);
                                break;
                    case '5':// BALANCE
                                System.out.println("Operation:BALANCE()");
                                atm.BALANCE();
                                break;
                    case '6':// LOGOUT
                                System.out.println("Operation:LOGOUT()");
                                atm.LOGOUT();
                                break;
                    case '7':// suspend
                                System.out.println("Operation:suspend()");
                                atm.suspend();
                                break;
                    case '8':// activate
```

```
                                System.out.println("Operation:activate()");
                                atm.activate();
                                break;
                        case '9':// close
                                System.out.println("Operation:close()");
                                atm.close();
                                break;
                              default:
                                System.out.println("Invalid Choice");
                                break;
        }
        }
                }
}
}
```

4.1 AbstractFactory.java

```
package abstract_factory;
import data_store.*;
import strategy.*;

/*
 * INTERFACE:: Abstract Factory PATTERN starts here
 */
public interface AbstractFactory
{
        public DataStore CreateDataStore();
        public Store_Pin CreateStore_Pin();
        public Store_ID CreateStore_ID();
        public Store_Balance CreateStore_Balance();
        public Incorrect_ID_Msg CreateIncorrect_ID_Msg();
        public Incorrect_Pin_Msg CreateIncorrect_Pin_Msg();
        public Too_Many_Attempts_Msg CreateToo_Many_Attempts_Msg();
        public Display_Menu CreateDisplay_Menu();
        public Make_Deposit CreateMake_Deposit();
        public Display_Balance CreateDisplay_Balance();
        public Prompt_For_PIN CreatePrompt_For_PIN();
        public Make_Withdraw CreateMake_Withdraw();
        public Penalty CreatePenalty();
        public Incorrect_Lock_Msg CreateIncorrect_Lock_Msg();
        public Incorrect_Unlock_Msg CreateIncorrect_Unlock_Msg();
        public No_Funds_Msg CreateNo_Funds_Msg();
}
```

5.2.1 ConcreteFactory1.java

```
package abstract_factory;
import data_store.DataStore;
import data_store.DataStore1;
import strategy.*;

/***
 *
 * @author fshen4
 * For Account 1
 *
 */
```

```
public class ConcreteFactory1 implements AbstractFactory
{
	DataStore dataStore = new DataStore1();
	Store_Pin store_pin= new Store_Pin_A1();
	Store_ID store_id = new Store_ID_A1();
	Store_Balance store_bal = new Store_Balance_A1();
	Incorrect_ID_Msg incorrect_id = new Incorrect_ID_Msg_A1();
	Incorrect_Pin_Msg incorrect_pin = new Incorrect_Pin_Msg_A1();
	Too_Many_Attempts_Msg too_many_attempts_msg = new Too_Many_Attempts_Msg_A1();
	Display_Menu disp_menu = new Display_Menu_A1();
	Make_Deposit make_deposit = new Make_Deposit_A1();
	Display_Balance disp_bal = new Display_Balance_A1();
	Prompt_For_PIN prompt_pin = new Prompt_For_PIN_A1();
	Make_Withdraw make_withdraw = new Make_Withdraw_A1();
	Penalty penalty = new Penalty_A1();
	Incorrect_Lock_Msg incorrect_lock = new Incorrect_Lock_Msg_A1();
	Incorrect_Unlock_Msg incorrect_unlock = new Incorrect_Unlock_Msg_A1();
	No_Funds_Msg no_funds = new No_Funds_Msg_A1();

	public void ConcreteFactory()
	{

	}
	public DataStore CreateDataStore()
	{
		return(this.dataStore);
	}
	public DataStore GetDataStore()
	{
		return this.dataStore;
	}
	public Incorrect_Pin_Msg CreateIncorrect_Pin_Msg()
	{
		return this.incorrect_pin;
	}
	public Too_Many_Attempts_Msg CreateToo_Many_Attempts_Msg()
	{
		return this.too_many_attempts_msg;
	}
	public Display_Menu CreateDisplay_Menu()
	{
		return this.disp_menu;
	}
	public Store_Pin CreateStore_Pin()
	{
		return this.store_pin;
	}
	public Store_Balance CreateStore_Balance()
	{
		return this.store_bal;
	}
	public Prompt_For_PIN CreatePrompt_For_PIN()
	{
		return this.prompt_pin;
	}
```

```
	public Display_Balance CreateDisplay_Balance()
	{
		return this.disp_bal;
	}
	public Make_Deposit CreateMake_Deposit()
	{
		return this.make_deposit;
	}
	public Make_Withdraw CreateMake_Withdraw()
	{
		return this.make_withdraw;
	}
	public Penalty CreatePenalty()
	{
		return this.penalty;
	}
	public Store_ID CreateStore_ID() {
		return this.store_id;
	}
	public Incorrect_ID_Msg CreateIncorrect_ID_Msg() {
		return this.incorrect_id;
	}
	public Incorrect_Lock_Msg CreateIncorrect_Lock_Msg() {
		return this.incorrect_lock;
	}
	public Incorrect_Unlock_Msg CreateIncorrect_Unlock_Msg() {
		return this.incorrect_unlock;
	}
	public No_Funds_Msg CreateNo_Funds_Msg() {
		return this.no_funds;
	}
}
```

5.2.2 ConcreteFactory2.java

```
package abstract_factory;
import data_store.DataStore;
import data_store.DataStore2;
import strategy.*;


/***
 *
 * @author fshen4
 * For Account 2
 *
 */

public class ConcreteFactory2 implements AbstractFactory
{

	DataStore dataStore = new DataStore2();
	Store_Pin store_pin= new Store_Pin_A2();
	Store_ID store_id = new Store_ID_A2();
	Store_Balance store_bal = new Store_Balance_A2();
	Incorrect_ID_Msg incorrect_id = new Incorrect_ID_Msg_A2();
	Incorrect_Pin_Msg incorrect_pin = new Incorrect_Pin_Msg_A2();
	Too_Many_Attempts_Msg too_many_attempts_msg = new Too_Many_Attempts_Msg_A2();
```

```
	Display_Menu disp_menu = new Display_Menu_A2();
	Make_Deposit make_deposit = new Make_Deposit_A2();
	Display_Balance disp_bal = new Display_Balance_A2();
	Prompt_For_PIN prompt_pin = new Prompt_For_PIN_A2();
	Make_Withdraw make_withdraw = new Make_Withdraw_A2();
	Penalty penalty = new Penalty_A2();
	Incorrect_Lock_Msg incorrect_lock = new Incorrect_Lock_Msg_A2();
	Incorrect_Unlock_Msg incorrect_unlock = new Incorrect_Unlock_Msg_A2();
	No_Funds_Msg no_funds = new No_Funds_Msg_A2();

	public void ConcreteFactory()
	{

	}
	public DataStore CreateDataStore()
	{
		return(this.dataStore);
	}
	public DataStore GetDataStore()
	{
		return this.dataStore;
	}
	public Incorrect_Pin_Msg CreateIncorrect_Pin_Msg()
	{
		return this.incorrect_pin;
	}
	public Too_Many_Attempts_Msg CreateToo_Many_Attempts_Msg()
	{
		return this.too_many_attempts_msg;
	}
	public Display_Menu CreateDisplay_Menu()
	{
		return this.disp_menu;
	}
	public Store_Pin CreateStore_Pin()
	{
		return this.store_pin;
	}
	public Store_Balance CreateStore_Balance()
	{
		return this.store_bal;
	}
	public Prompt_For_PIN CreatePrompt_For_PIN()
	{
		return this.prompt_pin;
	}
	public Display_Balance CreateDisplay_Balance()
	{
		return this.disp_bal;
	}
	public Make_Deposit CreateMake_Deposit()
	{
		return this.make_deposit;
	}
	public Make_Withdraw CreateMake_Withdraw()
	{
```

```
                return this.make_withdraw;
        }
        public Penalty CreatePenalty()
        {
                return this.penalty;
        }
        public Store_ID CreateStore_ID() {
                return this.store_id;
        }
        public Incorrect_ID_Msg CreateIncorrect_ID_Msg() {
                return this.incorrect_id;
        }
        public Incorrect_Lock_Msg CreateIncorrect_Lock_Msg() {
                return this.incorrect_lock;
        }
        public Incorrect_Unlock_Msg CreateIncorrect_Unlock_Msg() {
                return this.incorrect_unlock;
        }
        public No_Funds_Msg CreateNo_Funds_Msg() {
                return this.no_funds;
        }
}
/**************ABSTRACT AND CONCRETE FACTORY PATTERN ENDS HERE *****************/
```

5.3 DataStore.java

```
package data_store;
/*
 * ABSTRACT CLASS : DataStore
 */
public abstract class DataStore{

}
```

5.3.1 DataStore1.java

```
package data_store;

/*
 * CONCRETE CLASS : DataStore -1
 */
public class DataStore1 extends DataStore
{

        /* Temporary Storage variables */
        public String temp_PIN;
        public String temp_ID;
        public float temp_balance;
        public float temp_deposit;
        public float temp_withdraw;


        /* Permanent Storage Variables ******/
        public String PIN;// PIN String p
        public String ID;// ID String y
        public float balance; // a is the Balance
        public float deposit; // Deposit variable
        public float withdraw; // Withdraw variable
```

```
	public float getbalance()
	{
		return this.balance;
	}

	public String getPIN()
	{
		return this.PIN;
	}

	public String getID()
	{
		return this.ID;
	}

	public float setbalance()
	{
		return this.balance = this.temp_balance;
	}

	public String setPIN()
	{
		return this.PIN = this.temp_PIN;
	}
	public String setID()
	{
		return this.ID = this.temp_ID;
	}

	public float setPenalty()
	{
		this.balance = this.balance - 20;
		this.temp_balance = balance;
		return this.balance;
	}

	public void Compute_Balance_deposit()
	{
		this.balance = this.balance + this.deposit;
		this.temp_balance = this.balance;
	}
	public void Compute_Balance_withdraw()
	{
		this.balance = this.balance - this.withdraw;
		this.temp_balance = this.balance;
	}

	public void Set_Deposit()
	{
		this.deposit = this.temp_deposit;
	}

	public void Set_Withdraw()
	{
	this.withdraw = this.temp_withdraw;
```

```
	}
	public float Get_Deposit()
	{
		return this.deposit;
	}

	public float Get_Withdraw()
	{
		return this.withdraw;
	}
}
```

5.3.2 DataStore2.java

```
package data_store;

/*
 * CONCRETE CLASS : DataStore -2
 */
public class DataStore2 extends DataStore {

	/* Temporary Storage variables */
	public int temp_PIN;
	public int temp_ID;
	public int temp_balance;
	public int temp_deposit;
	public int temp_withdraw;


	/* Permanent Storage Variables ******/
	public int PIN;// PIN String p
	public int ID;// ID String y
	public int balance; // a is the Balance
	public int deposit; // Deposit variable
	public int withdraw; // Withdraw variable

	public int getbalance()
	{
		return balance;
	}

	public int getPIN()
	{
		return PIN;
	}

	public int getID()
	{
		return ID;
	}

	public int setbalance()
	{
		return this.balance = this.temp_balance;
	}

	public int setPIN()
	{
```

```
		return this.PIN = this.temp_PIN;
	}

	public int setID()
	{
		return this.ID = this.temp_ID;
	}

	public void Compute_Balance_deposit()
	{
		this.balance = this.balance + this.deposit;
		this.temp_balance = this.balance;
	}

	public void Compute_Balance_withdraw()
	{
		this.balance = this.balance - this.withdraw;
		this.temp_balance = this.balance;
	}

	public void Set_Deposit()
	{
		this.deposit = this.temp_deposit;
	}

	public void Set_Withdraw()
	{
	this.withdraw = this.temp_withdraw;
	}

	public float Get_Deposit()
	{
		return this.deposit;
	}

	public float Get_Withdraw()
	{
		return this.withdraw;
	}
}
/*************DATA STORE CLASSES DESCRIPTION ENDS HERE *****************/
```

5.4.1 ATM1.java

```
package account;
import mda_efsm.MDAEFSM;
import data_store.DataStore;
import data_store.DataStore1;


/*
 * CLASS : Account1 Implementation for collecting parameters
 * from the UI through Driver.java and invoking right event in MDA-EFSM
 */

public class Account1
{
```

```
	/* Pointer to MDA-EFSM */
	MDAEFSM mdaefsm = null;
	/* Pointer to DataStore */
	DataStore dataStore = null;

	public Account1(MDAEFSM mdaefsm,DataStore dataStore)
	{
this.mdaefsm = mdaefsm;
this.dataStore = dataStore;
this.create();
}

	public void create()
	{
		mdaefsm.create();
}

public void open(String p, String y, float a)
{
	// store p, y and a in temp data store
	((DataStore1)dataStore).temp_PIN = p;
	((DataStore1)dataStore).temp_ID = y;
	((DataStore1)dataStore).temp_balance = a;
	mdaefsm.Open();
}

public void pin( String x )
{
	if( x.equals(((DataStore1)dataStore).temp_PIN ) )
	{
		if( ((DataStore1)dataStore).temp_balance <= 500 )
			mdaefsm.CorrectPinBelowMin();
		else
			mdaefsm.CorrectPinAboveMin();
	}
	else
	{
		mdaefsm.IncorrectPin(3);
	}
}

public void deposit(float d)
{
	((DataStore1)dataStore).temp_deposit = d;
	mdaefsm.Deposit();

	if( ((DataStore1)dataStore).temp_balance > 500 )
	{
		mdaefsm.AboveMinBalance();
	}
	else
	{
		mdaefsm.BelowMinBalance();
	}
}
```

```
public void withdraw(float w)
{
	((DataStore1)dataStore).temp_withdraw = w;
	mdaefsm.Withdraw();

	if( ((DataStore1)dataStore).balance > 500 )
	{
		mdaefsm.AboveMinBalance();
	}
	else
	{
		mdaefsm.WithdrawBelowMinBalance();
	}
}

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

public void login(String y)
{
	if (y.equals(((DataStore1)dataStore).temp_ID))
	{
		mdaefsm.Login();
	}
	else
	{
		mdaefsm.IncorrectLogin();
	}
}

public void logout()
{
		mdaefsm.Logout();
}

public void lock(String x)
{
	if( x.equals(((DataStore1)dataStore).temp_PIN ) )
		mdaefsm.Lock();
	else
		mdaefsm.IncorrectLock();
}

public void unlock(String x)
{
	if( x.equals(((DataStore1)dataStore).temp_PIN ) )
	{
		mdaefsm.Unlock();

		if( ((DataStore1)dataStore).balance > 500 )
		{
			mdaefsm.AboveMinBalance();
		}
		else
```

```
                {
                        mdaefsm.BelowMinBalance();
                }
        }
        else
                mdaefsm.IncorrectUnlock();
}

}
```

5.4.2 ATM2.java

```
package account;

import mda_efsm.MDAEFSM;
import data_store.DataStore;
import data_store.DataStore2;

/*
 * CLASS : Account2 Implementation for collecting parameters
 * from the UI through Driver.java and invoking right event in MDA-EFSm
 */

public class Account2
{

        /* Pointer to MDA-EFSM */
        MDAEFSM mdaefsm = null;
        /* Pointer to DataStore */
        DataStore dataStore = null;

        public Account2(MDAEFSM mdaefsm,DataStore dataStore)
        {
this.mdaefsm = mdaefsm;
this.dataStore = dataStore;
this.create();
}

        public void create()
        {
                mdaefsm.create();
}
public void OPEN(int p, int y, int a)
{
        ((DataStore2)dataStore).temp_PIN = p;
        ((DataStore2)dataStore).temp_ID = y;
        ((DataStore2)dataStore).temp_balance = a;
        mdaefsm.Open();
}

public void PIN(int x)
{
        if( x == ((DataStore2)dataStore).temp_PIN )
        {
                        mdaefsm.CorrectPinAboveMin();
        }
        else
        {
```

```
                mdaefsm.IncorrectPin(2);
        }
}

public void DEPOSIT(int d)
{
        ((DataStore2)dataStore).temp_deposit = d;
        mdaefsm.Deposit();
}

public void WITHDRAW(int w)
{
        ((DataStore2)dataStore).temp_withdraw = w;

        if(((DataStore2)dataStore).balance > 0)
        {
                mdaefsm.Withdraw();
                mdaefsm.AboveMinBalance();
        }
        else
                mdaefsm.NoFunds();
}

public void BALANCE()
{
        mdaefsm.Balance();
}

public void LOGIN(int y)
{
        if (y == ((DataStore2)dataStore).ID)
        {
                mdaefsm.Login();
        }
        else
                mdaefsm.IncorrectLogin();
}

public void LOGOUT()
{
        mdaefsm.Logout();
}

public void suspend()
{
        mdaefsm.Suspend();
}

public void activate()
{
        mdaefsm.Activate();
}

public void close()
{
        mdaefsm.Close();
```

```
}
}
/******************Accounts END HERE ************************/
```

5.5 MDA-EFSM (STATE PATTERN)

5.5.1 MDAEFSM.java

```
package mda_efsm;
import abstract_factory.AbstractFactory;
import output.Output;

/*
 *CLASS : MDAEFSM ( STATE PATTERN )
 */

public class MDAEFSM {

State startState = new StartState(this);
State idleState = new IdleState(this);
State checkpinState = new CheckPinState(this);
State readyState = new ReadyState(this);
State s1State = new S1State(this);
State lockedState = new LockedState(this);
State overdrawnState = new OverdrawnState(this);
State suspendedState = new SuspendedState(this);

State efsmState = null;

//State List[8] ;
	public int attempts;

	AbstractFactory factory =null;
	Output output = null;

public MDAEFSM(AbstractFactory factory,Output output) {
efsmState = startState;
attempts = 0;
this.factory = factory;
this.output = output;
}

public void create()
{
	efsmState.Create();
	printCurrentState();
}
public void Open()
{
	efsmState.Open();
	printCurrentState();
}
public void Login()
{
	efsmState.Login();
	attempts = 0;
	printCurrentState();
}
```

```
public void IncorrectLogin()
{
	efsmState.IncorrectLogin();
	printCurrentState();
}
public void IncorrectPin(int max)
{
	efsmState.IncorrectPin(max);
	printCurrentState();
}
public void CorrectPinBelowMin()
{
	efsmState.CorrectPinBelowMin();
	printCurrentState();
}
public void CorrectPinAboveMin()
{
	efsmState.CorrectPinAboveMin();
	printCurrentState();
}
public void Deposit()
{
	efsmState.Deposit();
	printCurrentState();
}
public void BelowMinBalance()
{
	efsmState.BelowMinBalance();
	printCurrentState();
}
public void AboveMinBalance()
{
	efsmState.AboveMinBalance();
	printCurrentState();
}
public void Logout()
{
	efsmState.Logout();
	printCurrentState();
}
public void Balance()
{
	efsmState.Balance();
	printCurrentState();
}
public void Withdraw()
{
	efsmState.Withdraw();
	printCurrentState();
}
public void WithdrawBelowMinBalance()
{
	efsmState.WithdrawBelowMinBalance();
	printCurrentState();
}
public void NoFunds()
```

```
{
	efsmState.NoFunds();
	printCurrentState();
}
public void Lock()
{
	efsmState.Lock();
	printCurrentState();
}
public void IncorrectLock()
{
	efsmState.IncorrectLock();
	printCurrentState();
}
public void Unlock()
{
	efsmState.Unlock();
	printCurrentState();
}
public void IncorrectUnlock()
{
	efsmState.IncorrectUnlock();
	printCurrentState();
}
public void Suspend()
{
	efsmState.Suspend();
	printCurrentState();
}
public void Activate()
{
	efsmState.Activate();
	printCurrentState();
}
public void Close()
{
	efsmState.Close();
	printCurrentState();
}

public void setState(State efsmState)
{
this.efsmState = efsmState;
}
public State getMachineState() {
return efsmState;
}

public State getStartState() {
return startState;
}
public State getIdleState() {
return idleState;
}
public State getCheckPinState() {
return checkpinState;
```

```
}
public State getReadyState() {
return readyState;
}
public State getS1State() {
return s1State;
}
public State getOverdrawnState() {
return overdrawnState;
}
public State getLockedState() {
return lockedState;
}
public State getSuspendedState() {
return suspendedState;
}

public void printCurrentState(){
	System.out.println("--- Current State : "+ efsmState.getClass().getName()+"---");
}

}
```

5.5.2 State.java

```
package mda_efsm;

/*
 * ABSTRACT CLASS : State( STATE PATTERN )
 */

public interface State
{

	public void Create();
	public void Open();
	public void Login();
	public void IncorrectLogin();
	public void IncorrectPin(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();
	public void Close();

}
```

5.5.3 StartState.java

```java
package mda_efsm;

/***
 *
 * @author fshen4
 * CLASS : StartState ( STATE_PATTERN )
 *
 */
public class StartState implements State
{
	MDAEFSM mdaefsm =null;

public StartState(MDAEFSM mdaefsm)
{
this.mdaefsm =mdaefsm;
}

	public void Open() {
		// TODO Auto-generated method stub
	mdaefsm.output.Store_Pin();
	mdaefsm.output.Store_ID();
	mdaefsm.output.Store_Balance();
	mdaefsm.setState(mdaefsm.getIdleState());

	}

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

	}

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

	}

	public void IncorrectPin(int max) {
		// TODO Auto-generated method stub

	}

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

	}

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

	}

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

	}
```

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}
```

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

	}

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

	}

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

	}

	public void Create() {
		// TODO Auto-generated method stub
	}
}
```

5.5.4 IdleState.java

```
package mda_efsm;

/***
 *
 * @author fshen4
 * CLASS : IdleState ( STATE_PATTERN )
 *
 */
public class IdleState implements State
{
	MDAEFSM mdaefsm =null;

public IdleState(MDAEFSM mdaefsm)
{
this.mdaefsm =mdaefsm;
}

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

	}

	public void Login() {
		// TODO Auto-generated method stub
	mdaefsm.attempts = 0;
	mdaefsm.output.Prompt_For_Pin();
	mdaefsm.setState(mdaefsm.getCheckPinState());

	}

	public void IncorrectLogin() {
		// TODO Auto-generated method stub
		mdaefsm.output.Incorrect_ID_Msg();

	}
```

```
	public void IncorrectPin(int max) {
		// TODO Auto-generated method stub

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

	public void Lock() {
```

```
		// TODO Auto-generated method stub

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

}
```

5.5.5 CheckPinState.java

```
package mda_efsm;

/***
 *
 * @author fshen4
 * CLASS : CheckPinState ( STATE_PATTERN )
 *
 */
public class CheckPinState implements State
{
MDAEFSM mdaefsm =null;

public CheckPinState(MDAEFSM mdaefsm)
{
this.mdaefsm =mdaefsm;
```

```
}


public void IncorrectPin(int max)
{
	if( mdaefsm.attempts < max )
	{
		mdaefsm.attempts++;
		mdaefsm.output.Incorrect_Pin_Msg();
	}
	else if( mdaefsm.attempts >= max )
	{
		mdaefsm.output.Incorrect_Pin_Msg();
		mdaefsm.output.Too_Many_Attempts_Msg();
		mdaefsm.setState(mdaefsm.getIdleState());
	}
}

public void CorrectPinBelowMin()
{
	mdaefsm.output.Display_Menu();
	mdaefsm.setState(mdaefsm.getOverdrawnState());
}

public void CorrectPinAboveMin()
{
	mdaefsm.output.Display_Menu();
	mdaefsm.setState(mdaefsm.getReadyState());
}

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

	}

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

	}

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

	}

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

	}

	public void Logout() {
		// TODO Auto-generated method stub
	mdaefsm.setState(mdaefsm.getIdleState());
	}

	public void Balance() {
```

```
		// TODO Auto-generated method stub

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}


	public void BelowMinBalance() {
```

```
		// TODO Auto-generated method stub

	}


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

	}


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

	}
}
```

5.5.6 ReadyState.java

```
package mda_efsm;

/***
 *
 * @author fshen4
 * CLASS : ReadyState ( STATE_PATTERN )
 *
 */
public class ReadyState implements State
{
	MDAEFSM mdaefsm =null;

public ReadyState(MDAEFSM mdaefsm)
{
this.mdaefsm =mdaefsm;
}

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

	}

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

	}

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

	}

	public void IncorrectPin(int max) {
		// TODO Auto-generated method stub

	}

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

```
	}

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

	}

	public void Deposit() {
		// TODO Auto-generated method stub
	mdaefsm.output.Make_Deposit();

	}

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

	}

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

	}

	public void Logout() {
		// TODO Auto-generated method stub
	mdaefsm.setState(mdaefsm.getIdleState());

	}

	public void Balance() {
		// TODO Auto-generated method stub
	mdaefsm.output.Display_Balance();
	}

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

	mdaefsm.output.Make_Withdraw();
	mdaefsm.setState(mdaefsm.getS1State());
	}

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

	}

	public void NoFunds() {
		// TODO Auto-generated method stub
		mdaefsm.output.No_Funds_Msg();
	}

	public void Lock() {
		// TODO Auto-generated method stub
	mdaefsm.setState(mdaefsm.getLockedState());
```

```
	}

	public void IncorrectLock() {
		// TODO Auto-generated method stub
		mdaefsm.output.Incorrect_Lock_Msg();

	}

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

	}

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

	}

	public void Suspend() {
		// TODO Auto-generated method stub
		mdaefsm.setState(mdaefsm.getSuspendedState());
	}

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

	}

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

	}

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

	}
}
```

5.5.7 OverdrawnState.java

```
package mda_efsm;

/***
 *
 * @author fshen4
 * CLASS : OverdrawnState ( STATE_PATTERN )
 *
 */
public class OverdrawnState implements State
{
	MDAEFSM mdaefsm =null;

public OverdrawnState(MDAEFSM mdaefsm)
{
this.mdaefsm =mdaefsm;
}
```

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

	}

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

	}

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

	}

	public void IncorrectPin(int max) {
		// TODO Auto-generated method stub

	}

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

	}

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

	}

	public void Deposit() {
		// TODO Auto-generated method stub
	mdaefsm.output.Make_Deposit();
	mdaefsm.setState(mdaefsm.getS1State());

	}

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

	}

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

	}

	public void Logout() {
		// TODO Auto-generated method stub
	mdaefsm.setState(mdaefsm.getIdleState());

	}

	public void Balance() {
		// TODO Auto-generated method stub
	mdaefsm.output.Display_Balance();
```

```
	}

	public void Withdraw() {
		// TODO Auto-generated method stub
	mdaefsm.output.No_Funds_Msg();
	}

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

	}

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

	}

	public void Lock() {
		// TODO Auto-generated method stub
	mdaefsm.setState(mdaefsm.getLockedState());
	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

```
	}
}
```

5.5.8 LockedState.java

```
package mda_efsm;

/***
 *
 * @author fshen4
 * CLASS : LockedState ( STATE_PATTERN )
 *
 */
public class LockedState implements State
{
	MDAEFSM mdaefsm =null;

public LockedState(MDAEFSM mdaefsm)
{
this.mdaefsm =mdaefsm;
}

public void IncorrectPin(int max)
{
}

public void CorrectPinBelowMin()
{
}

public void CorrectPinAboveMin()
{
}

public void BelowMinBalance()
{
}

public void AboveMinBalance()
{
}

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

	}

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

	}

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

	}
```

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	public void Unlock() {
		// TODO Auto-generated method stub
	System.out.println("\n MDA_EFSM::LockedState::Unlock function ");
	mdaefsm.setState(mdaefsm.getS1State());

	}

	public void IncorrectUnlock() {
		// TODO Auto-generated method stub
	System.out.println("\n MDA_EFSM::LockedState::IncorrectLock function ");
	mdaefsm.output.Incorrect_Unlock_Msg();

	}

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

```
	}

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

	}

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

	}

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

	}
}
```

5.5.9 S1State.java

```
package mda_efsm;

/***
 *
 * @author fshen4
 * CLASS : S1State ( STATE_PATTERN )
 *
 */
public class S1State implements State
{
	MDAEFSM mdaefsm =null;

public S1State(MDAEFSM mdaefsm)
{
this.mdaefsm =mdaefsm;
}

public void BelowMinBalance()
{
	mdaefsm.setState(mdaefsm.getOverdrawnState());
}

public void AboveMinBalance()
{
	mdaefsm.setState(mdaefsm.getReadyState());
}

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

	}

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

	}

	public void IncorrectLogin() {
```

```
		// TODO Auto-generated method stub

	}

	public void IncorrectPin(int max) {
		// TODO Auto-generated method stub

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}

	public void WithdrawBelowMinBalance() {
		// TODO Auto-generated method stub
	mdaefsm.output.Penalty();
	mdaefsm.setState(mdaefsm.getOverdrawnState());
	}

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

	}

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

	}

	public void IncorrectLock() {
```

```
		// TODO Auto-generated method stub

	}

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

	}

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

	}

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

	}

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

	}

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

	}

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

	}
}
```

5.5.10 SuspendedState.java

```
package mda_efsm;

/***
 *
 * @author fshen4
 * CLASS : S2State ( STATE_PATTERN )
 *
 */
public class SuspendedState implements State
{
	MDAEFSM mdaefsm =null;

public SuspendedState(MDAEFSM mdaefsm)
{
this.mdaefsm =mdaefsm;
}

public void BelowMinBalance()
{
}
```

```
public void AboveMinBalance()
{
}

public void lock()
{
}

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

	}

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

	}

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

	}

	public void IncorrectPin(int max) {
		// TODO Auto-generated method stub

	}

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

	}

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

	}

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

	}

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

	}

	public void Balance() {
		// TODO Auto-generated method stub
	mdaefsm.output.Display_Balance();
	}

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

```
	}

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

	}

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

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

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

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

	}

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

	}

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

	}

	public void Activate() {
		// TODO Auto-generated method stub
	mdaefsm.setState(mdaefsm.getReadyState());
	}

	public void Close() {
		// TODO Auto-generated method stub
		System.exit(0);

	}

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

	}

}
/************MDA_EFSM:: STATE PATTERN ENDS HERE **************************/
```

4.3 Output.java

```
/***
*** Output Actions triggering for STRATEGY and FACTORY PATTERN To Implement ***/

public void Incorrect_ID_Msg()
{
	System.out.println("\n OUTPUT:: Action Incorrect_ID_Msg");
	Incorrect_ID_Msg incorrect_id = factory.CreateIncorrect_ID_Msg();
	incorrect_id.Incorrect_ID_Msg();
}

public void Incorrect_Pin_Msg()
{
	System.out.println("\n OUTPUT:: Action Incorrect_Pin_Msg");
	Incorrect_Pin_Msg incorrect_pin = factory.CreateIncorrect_Pin_Msg();
	incorrect_pin.Incorrect_Pin_Msg();
}

public void Too_Many_Attempts_Msg()
{
	System.out.println("\n OUTPUT:: Action Too_Many_Attempts_Msg");
	Too_Many_Attempts_Msg too_many_attempts = factory.CreateToo_Many_Attempts_Msg();
	too_many_attempts.Too_Many_Attempts_Msg();
}

public void Display_Menu()
{
	System.out.println("\n OUTPUT:: Action Display_Menu");
	Display_Menu disp_menu = factory.CreateDisplay_Menu();
	disp_menu.Display_Menu();
}

public void Make_Deposit()
{
	System.out.println("\n OUTPUT:: Action Make_Deposit ");
	Make_Deposit make_deposit = factory.CreateMake_Deposit();
	make_deposit.Make_Deposit(dataStore);
}

public void Display_Balance()
{
	System.out.println("\n OUTPUT:: Action Display_Balance ");
	Display_Balance disp_bal = factory.CreateDisplay_Balance();
	disp_bal.Display_Balance(dataStore);
}

public void Prompt_For_Pin()
{
	System.out.println("\n OUTPUT:: Action Prompt_For_Pin ");
	Prompt_For_PIN prompt_pin = factory.CreatePrompt_For_PIN();
	prompt_pin.Prompt_For_PIN();
}

public void Make_Withdraw()
{
	System.out.println("\n OUTPUT:: Action Make_Withdraw ");
```

```
        Make_Withdraw make_withdraw = factory.CreateMake_Withdraw();
        make_withdraw.Make_Withdraw(dataStore);
}

public void Penalty()
{
        System.out.println("\n OUTPUT:: Action Penalty");
        Penalty penalty = factory.CreatePenalty();
        penalty.Penalty(dataStore);
}

public void Incorrect_Lock_Msg()
{
        System.out.println("\n OUTPUT:: Action Incorrect_Lock_Msg");
        Incorrect_Lock_Msg incorrect_lock = factory.CreateIncorrect_Lock_Msg();
        incorrect_lock.Incorrect_Lock_Msg();
}

public void Incorrect_Unlock_Msg()
{
        System.out.println("\n OUTPUT:: Action Incorrect_Unlock_Msg");
        Incorrect_Unlock_Msg incorrect_unlock = factory.CreateIncorrect_Unlock_Msg();
        incorrect_unlock.Incorrect_Unlock_Msg();
}

public void No_Funds_Msg()
{
        System.out.println("\n OUTPUT:: Action No_Funds_Msg");
        No_Funds_Msg no_funds_msg = factory.CreateNo_Funds_Msg();
        no_funds_msg.No_Funds_Msg();
}

public void Store_Pin()
{
        System.out.println("\n OUTPUT:: Action Store_Pin");
        Store_Pin store_pin = factory.CreateStore_Pin();
        store_pin.Store_Pin(dataStore);
}

public void Store_ID()
{
        System.out.println("\n OUTPUT:: Action Store_ID");
        Store_ID store_id = factory.CreateStore_ID();
        store_id.Store_ID(dataStore);
}

public void Store_Balance()
{
        System.out.println("\n OUTPUT:: Action Store_Balance");
        Store_Balance store_bal = factory.CreateStore_Balance();
        store_bal.Store_Balance(dataStore);
}
```

4.4 STRATEGY PATTERN

1. Display_Balance.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Display_Balance (STRAGTEGY PATTERN)
 */

public abstract class Display_Balance
{
	public abstract void Display_Balance(DataStore dataStore);
}
```

2. Display_Balance_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Display_Balance_A1 (STRATEGY PATTERN)
 */
public class Display_Balance_A1 extends Display_Balance
{
	public void Display_Balance(DataStore dataStore)
	{
		System.out.println("Account 1:: Balance is " +
((DataStore1)dataStore).getbalance() );
	}
}
```

3. Display_Balance_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Display_Balance_A2 (STRATEGY PATTERN)
 */
public class Display_Balance_A2 extends Display_Balance
{
	public void Display_Balance(DataStore dataStore)
	{
		System.out.println("Account 2:: Balance is " +
((DataStore2)dataStore).getbalance() );
	}
}
```

```
/************END OF DISPLAY BALANCE FUNCTIONALITY ************************/
```

4. Display_Menu.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Display_Menu (STRAGTEGY PATTERN)
 */

public abstract class Display_Menu
{
	public abstract void Display_Menu();
```

```
}
```

5. Display_Menu_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Display_Menu_A1 (STRATEGY PATTERN)
 */
public class Display_Menu_A1 extends Display_Menu
{
	public void Display_Menu()
	{
		System.out.println("Account 1:: Transaction Menu " );
		System.out.println(" 1:: Balance " );
		System.out.println(" 2:: Deposit " );
		System.out.println(" 3:: Withdraw " );
		System.out.println(" 4:: Lock " );
		System.out.println(" 5:: Logout " );
	}
}
```

6. Display_Menu_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Display_Menu_A2 (STRATEGY PATTERN)
 */
public class Display_Menu_A2 extends Display_Menu
{
	public void Display_Menu()
	{
		System.out.println("Account 2:: Transaction Menu " );
		System.out.println(" 1:: Balance " );
		System.out.println(" 2:: Deposit " );
		System.out.println(" 3:: Withdraw " );
		System.out.println(" 4:: Suspend " );
		System.out.println(" 5:: Logout " );
	}
}

/**********END OF DISPLAY MENU **********************************************/
```

7. Incorrect_ID_Msg.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Incorrect_ID_Msg (STRAGTEGY PATTERN)
 */

public abstract class Incorrect_ID_Msg
{
	public abstract void Incorrect_ID_Msg();
}
```

8. Incorrect_ID_Msg_A1.java

```
package strategy;
import data_store.*;
```

```
/*
 *CLASS : Incorrect_ID_Msg_A1 (STRATEGY PATTERN)
 */
public class Incorrect_ID_Msg_A1 extends Incorrect_ID_Msg
{
	public void Incorrect_ID_Msg()
	{
		System.out.println("Account 1:: Incorrect ID " );
	}
}
```

9. Incorrect_ID_Msg_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Incorrect_ID_Msg_A2 (STRATEGY PATTERN)
 */
public class Incorrect_ID_Msg_A2 extends Incorrect_ID_Msg
{
	public void Incorrect_ID_Msg()
	{
		System.out.println("Account 2:: Incorrect ID " );
	}
}
```

10. Incorrect_Lock_Msg.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Incorrect_Lock_Msg (STRAGTEGY PATTERN)
 */

public abstract class Incorrect_Lock_Msg
{
	public abstract void Incorrect_Lock_Msg();
}
```

11. Incorrect_Lock_Msg_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Incorrect_Lock_Msg_A1 (STRATEGY PATTERN)
 */
public class Incorrect_Lock_Msg_A1 extends Incorrect_Lock_Msg
{
	public void Incorrect_Lock_Msg()
	{
		System.out.println("Account 1:: Incorrect Lock " );
	}
}
```

12. Incorrect_Lock_Msg_A2.java

```
package strategy;
import data_store.*;
```

```
/*
 *CLASS : Incorrect_Lock_Msg_A2 (STRATEGY PATTERN)
 */
public class Incorrect_Lock_Msg_A2 extends Incorrect_Lock_Msg
{
	public void Incorrect_Lock_Msg()
	{
		System.out.println("Account 2:: Incorrect Lock " );
	}
}
```

13. Incorrect_Pin_Msg.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Incorrect_Pin_Msg (STRAGTEGY PATTERN)
 */

public abstract class Incorrect_Pin_Msg
{
	public abstract void Incorrect_Pin_Msg();
}
```

14. Incorrect_Pin_Msg_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Incorrect_Pin_Msg_A1 (STRATEGY PATTERN)
 */
public class Incorrect_Pin_Msg_A1 extends Incorrect_Pin_Msg
{
	public void Incorrect_Pin_Msg()
	{
		System.out.println("Account 1:: Incorrect Pin " );
	}
}
```

15. Incorrect_Pin_Msg_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Incorrect_Pin_Msg_A2 (STRATEGY PATTERN)
 */
public class Incorrect_Pin_Msg_A2 extends Incorrect_Pin_Msg
{
	public void Incorrect_Pin_Msg()
	{
		System.out.println("Account 2:: Incorrect Pin " );
	}
}
```

16. Incorrect_Unlock_Msg.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Incorrect_Unlock_Msg (STRAGTEGY PATTERN)
```

```
 */

public abstract class Incorrect_Unlock_Msg
{
	public abstract void Incorrect_Unlock_Msg();
}
```

17. Incorrect_Unlock_Msg_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Incorrect_Unlock_Msg_A1 (STRATEGY PATTERN)
 */
public class Incorrect_Unlock_Msg_A1 extends Incorrect_Unlock_Msg
{
	public void Incorrect_Unlock_Msg()
	{
		System.out.println("Account 1:: Incorrect Unlock " );
	}
}
```

18. Incorrect_Unlock_Msg_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Incorrect_Unlock_Msg_A2 (STRATEGY PATTERN)
 */
public class Incorrect_Unlock_Msg_A2 extends Incorrect_Unlock_Msg
{
	public void Incorrect_Unlock_Msg()
	{
		System.out.println("Account 2:: Incorrect Unlock " );
	}
}
```

19. Make_Deposit.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Make_Deposit (STRAGTEGY PATTERN)
 */

public abstract class Make_Deposit
{
	public abstract void Make_Deposit(DataStore dataStore);
}
```

20. Make_Deposit_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Make_Deposit_A1 (STRATEGY PATTERN)
 */
public class Make_Deposit_A1 extends Make_Deposit
{
	public void Make_Deposit(DataStore dataStore)
```

```
	{
		((DataStore1)dataStore).Set_Deposit();
		((DataStore1)dataStore).Compute_Balance_deposit();
		System.out.println("Account 1:: After Deposit, Balance is " +
((DataStore1)dataStore).getbalance() );
	}
}
```

21. Make_Deposit_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Make_Deposit_A2 (STRATEGY PATTERN)
 */
public class Make_Deposit_A2 extends Make_Deposit
{
	public void Make_Deposit(DataStore dataStore)
	{
		((DataStore2)dataStore).Set_Deposit();
		((DataStore2)dataStore).Compute_Balance_deposit();
		System.out.println("Account 2:: After Deposit, Balance is " +
((DataStore2)dataStore).getbalance() );
	}
}
```

22. Make_Withdraw.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Make_Withdraw (STRAGTEGY PATTERN)
 */
public abstract class Make_Withdraw
{
	public abstract void Make_Withdraw(DataStore dataStore);
}
```

23. Make_Withdraw_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Make_Withdraw_A1 (STRATEGY PATTERN)
 */
public class Make_Withdraw_A1 extends Make_Withdraw
{
	public void Make_Withdraw(DataStore dataStore)
	{
		((DataStore1)dataStore).Set_Withdraw();
		((DataStore1)dataStore).Compute_Balance_withdraw();
		System.out.println("Account 1:: After Withdraw, Balance is " +
((DataStore1)dataStore).getbalance() );
	}
}
```

24. Make_Withdraw_A2.java

```
package strategy;
import data_store.*;
```

```
/*
 *CLASS : Make_Withdraw_A2 (STRATEGY PATTERN)
 */
public class Make_Withdraw_A2 extends Make_Withdraw
{
	public void Make_Withdraw(DataStore dataStore)
	{
		((DataStore2)dataStore).Set_Withdraw();
		((DataStore2)dataStore).Compute_Balance_withdraw();
		System.out.println("Account 2:: After Withdraw, Balance is " +
((DataStore2)dataStore).getbalance() );
	}
}
```

25. No_Funds_Msg.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Too_Many_Attempts_Msg (STRAGTEGY PATTERN)
 */

public abstract class No_Funds_Msg
{
	public abstract void No_Funds_Msg();
}
```

26. No_Funds_Msg_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : No_Funds_Msg_A1 (STRATEGY PATTERN)
 */
public class No_Funds_Msg_A1 extends No_Funds_Msg
{
	public void No_Funds_Msg()
	{
		System.out.println("Account 1:: Below mininum balance " );
	}
}
```

27. No_Funds_Msg_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : No_Funds_Msg_A2 (STRATEGY PATTERN)
 */
public class No_Funds_Msg_A2 extends No_Funds_Msg
{
	public void No_Funds_Msg()
	{
		System.out.println("Account 2:: No Funds " );
	}
}
```

28. Penalty.java

```
package strategy;
import data_store.*;
```

```java
/*
 *ABSTRACT CLASS : Penalty (STRAGTEGY PATTERN)
 */
public abstract class Penalty
{
	public abstract void Penalty(DataStore dataStore);
}
```

29. Penalty_A1.java

```java
package strategy;
import data_store.*;

/*
 *CLASS : Penalty_A1 (STRATEGY PATTERN)
 */
public class Penalty_A1 extends Penalty
{
	public void Penalty(DataStore dataStore)
	{
		((DataStore1)dataStore).setPenalty();
		System.out.println("Account 1:: Minimum required balance is $500. So Penalty
is applied.");
		System.out.println("Account 1:: After a Penalty of 20$, Balance is " +
((DataStore1)dataStore).balance );
	}
}
```

30. Penalty_A2.java

```java
package strategy;
import data_store.*;

/*
 *CLASS : Penalty_A2 (STRATEGY PATTERN)
 */
public class Penalty_A2 extends Penalty
{
	public void Penalty(DataStore dataStore)
	{
		System.out.println("Account 2:: Minimum required balance is $0.But no Penalty
is applied.");
	}
}
```

31. Prompt_For_PIN.java

```java
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Prompt_For_PIN (STRAGTEGY PATTERN)
 */

public abstract class Prompt_For_PIN
{
	public abstract void Prompt_For_PIN();
}
```

32. Prompt_For_PIN_A1.java

```java
package strategy;
import data_store.*;
```

```
/*
 *CLASS : Prompt_For_PIN_A1 (STRATEGY PATTERN)
 */
public class Prompt_For_PIN_A1 extends Prompt_For_PIN
{
	public void Prompt_For_PIN()
	{
		System.out.println("Account 1:: Enter the Pin:: " );
	}
}
```

33. Prompt_For_PIN_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Prompt_For_PIN_A2 (STRATEGY PATTERN)
 */
public class Prompt_For_PIN_A2 extends Prompt_For_PIN
{
	public void Prompt_For_PIN()
	{
		System.out.println("Account 2:: Enter the Pin:: " );
	}
}
```

34. Store_Balance.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Store_Balance (STRAGTEGY PATTERN)
 */

public abstract class Store_Balance
{
	public abstract void Store_Balance(DataStore dataStore);
}
```

35. Store_Balance_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Store_Balance_Account1 (STRATEGY PATTERN)
 */
public class Store_Balance_A1 extends Store_Balance
{
	public void Store_Balance(DataStore dataStore)
	{
		((DataStore1)dataStore).setbalance();
		System.out.println("Account1:: After Storing the Balance, Balance is " +
((DataStore1)dataStore).getbalance() );
	}
}
```

36. Store_Balance_A2.java

```
package strategy;
import data_store.*;
```

```
/*
 *CLASS : Store_Balance_Account2 (STRATEGY PATTERN)
 */
public class Store_Balance_A2 extends Store_Balance
{
	public void Store_Balance(DataStore dataStore)
	{
		((DataStore2)dataStore).setbalance();
		System.out.println("Account2:: After Storing the Balance, Balance is " +
((DataStore2)dataStore).getbalance() );
	}
}
```

37. Store_ID.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Store_Balance (STRAGTEGY PATTERN)
 */

public abstract class Store_ID
{
	public abstract void Store_ID(DataStore dataStore);
}
```

38. Store_ID_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Store_ID_Account1 (STRATEGY PATTERN)
 */
public class Store_ID_A1 extends Store_ID
{
	public void Store_ID(DataStore dataStore)
	{
		((DataStore1)dataStore).setID();
		System.out.println("Account1:: After Storing the ID, ID is " +
((DataStore1)dataStore).ID );
	}
}
```

39. Store_ID_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Store_ID_Account2 (STRATEGY PATTERN)
 */
public class Store_ID_A2 extends Store_ID
{
	public void Store_ID(DataStore dataStore)
	{
		((DataStore2)dataStore).setID();
		System.out.println("Account2:: After Storing the ID, ID is " +
((DataStore2)dataStore).ID );
	}
```

```
}
```

40. Store_Pin.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Store_Pin (STRAGTEGY PATTERN)
 */

public abstract class Store_Pin
{
	public abstract void Store_Pin(DataStore datastore);
}
```

41. Store_Pin_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Store_Pin_Account1 (STRATEGY PATTERN)
 */
public class Store_Pin_A1 extends Store_Pin
{
	public void Store_Pin(DataStore dataStore)
	{
		((DataStore1)dataStore).setPIN();
		System.out.println("Account1:: After Storing the PIN, PIN is " +
((DataStore1)dataStore).PIN );
	}
}
```

42. Store_Pin_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Store_Pin_Account2 (STRATEGY PATTERN)
 */
public class Store_Pin_A2 extends Store_Pin
{
	public void Store_Pin(DataStore dataStore)
	{
		((DataStore2)dataStore).setPIN();
		System.out.println("Account2:: After Storing the PIN, PIN is " +
((DataStore2)dataStore).PIN );
	}
}
```

43. Too_Many_Attempts_Msg.java

```
package strategy;
import data_store.*;

/*
 *ABSTRACT CLASS : Too_Many_Attempts_Msg (STRAGTEGY PATTERN)
 */

public abstract class Too_Many_Attempts_Msg
{
	public abstract void Too_Many_Attempts_Msg();
```

```
}
```

44. Too_Many_Attempts_Msg_A1.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Too_Many_Attempts_Msg_A1 (STRATEGY PATTERN)
 */
public class Too_Many_Attempts_Msg_A1 extends Too_Many_Attempts_Msg
{
	public void Too_Many_Attempts_Msg()
	{
		System.out.println("Account 1:: Too Many Attempts " );
	}
}
```

45. Too_Many_Attempts_Msg_A2.java

```
package strategy;
import data_store.*;

/*
 *CLASS : Too_Many_Attempts_Msg_A2 (STRATEGY PATTERN)
 */
public class Too_Many_Attempts_Msg_A2 extends Too_Many_Attempts_Msg
{
	public void Too_Many_Attempts_Msg()
	{
		System.out.println("Account 2:: Too Many Attempts " );
	}
}
/**********END OF STRATEGY PATTERN FUNCTIONALITY ****************************/
```