CS-586

Final Project Report

Software System Architecture

Kapeel Daryani
5/5/2014

Contents

Kapeel Daryani
A20313146

Introduction

The requirement of the project is to design three ATM components using Model Driven Architecture (MDA) and Object Oriented Pattern which include state pattern, strategy pattern and abstract factory pattern. The design is further implemented using Java.

The ATM-1 component supports the following operations

• create ()	To create ATM
• card (int x, string y)	To begin the operation where x is the balance and y is pin
• pin (string x)	Pin is required to perform further operations on the ATM
• deposit (int d)	Deposits the amount d
• withdraw(int w)	Withdraws the amount w
• balance()	Displays the current balance
• lock(string x)	Locks the ATM using the pin x
• unlock(string x)	Unlocks the ATM using the pin x
• exit()	Exits the ATM

The following is the state diagram representing the flow for ATM-1

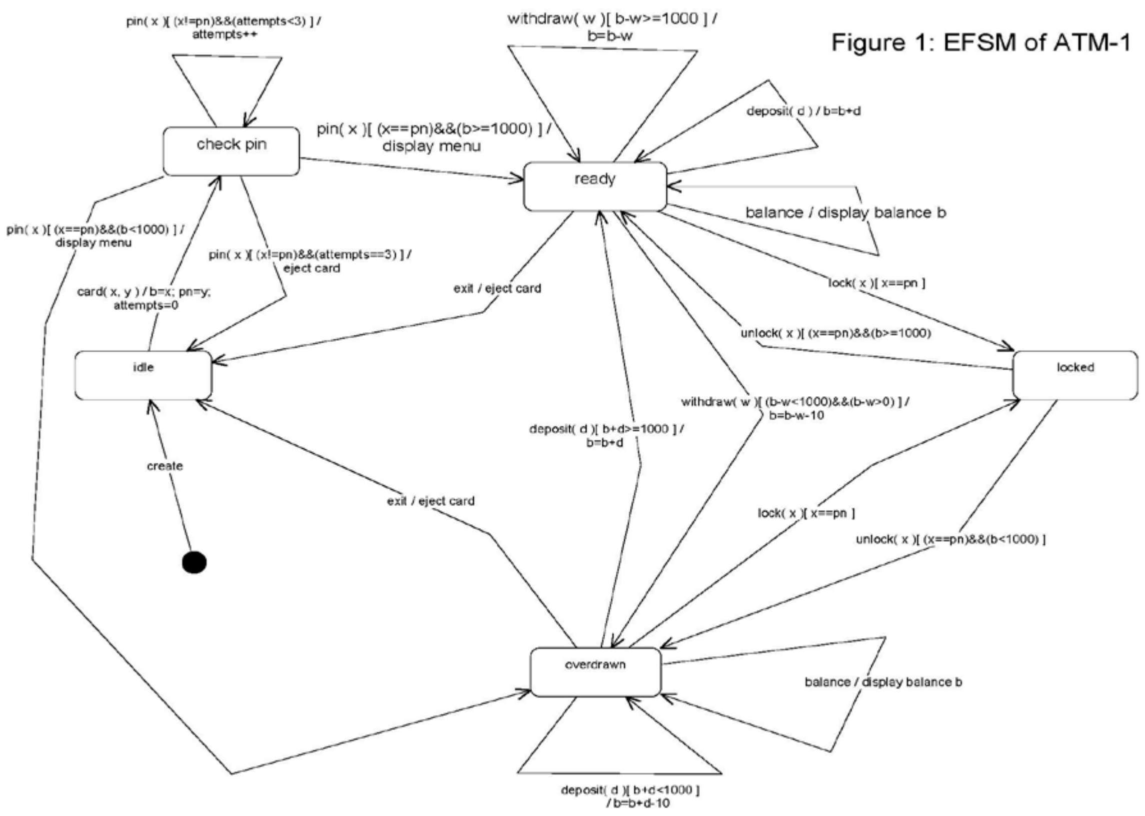

pin(x)[(x!=pn)&&(attempts<3)] /
attempts++
withdraw(w)[b-w>=1000] /
b=b-w
check pin
pin(x)[(x==pn)&&(b>=1000)] /
display menu
deposit(d) / b=b+d
ready
balance / display balance b
pin(x)[(x==pn)&&(b<1000)] /
display menu
pin(x)[(x!=pn)&&(attempts==3)] /
eject card
card(x, y) / b=x; pn=y;
attempts=0
exit / eject card
lock(x)[x==pn]
unlock(x)[(x==pn)&&(b>=1000)
idle
locked
withdraw(w)[(b-w<1000)&&(b-w>0)] /
b=b-w-10
deposit(d)[b+d>=1000] /
b=b+d
create
exit / eject card
lock(x)[x==pn]
unlock(x)[(x==pn)&&(b<1000)]
overdrawn
balance / display balance b
deposit(d)[b+d<1000]
/ b=b+d-10

Figure 1: EFSM of ATM-1

The ATM-2 component supports the following operations

• create()	To create ATM
• CARD (float x, int y)	To begin the operation where x is the balance and y is pin
• PIN (int x)	Pin is required to perform further operations on the ATM
• DEPOSIT (float d)	Deposits the amount d
• WITHDRAW (float w)	Withdraws the amount w
• BALANCE ()	Displays the current balance
• EXIT()	Exits the ATM

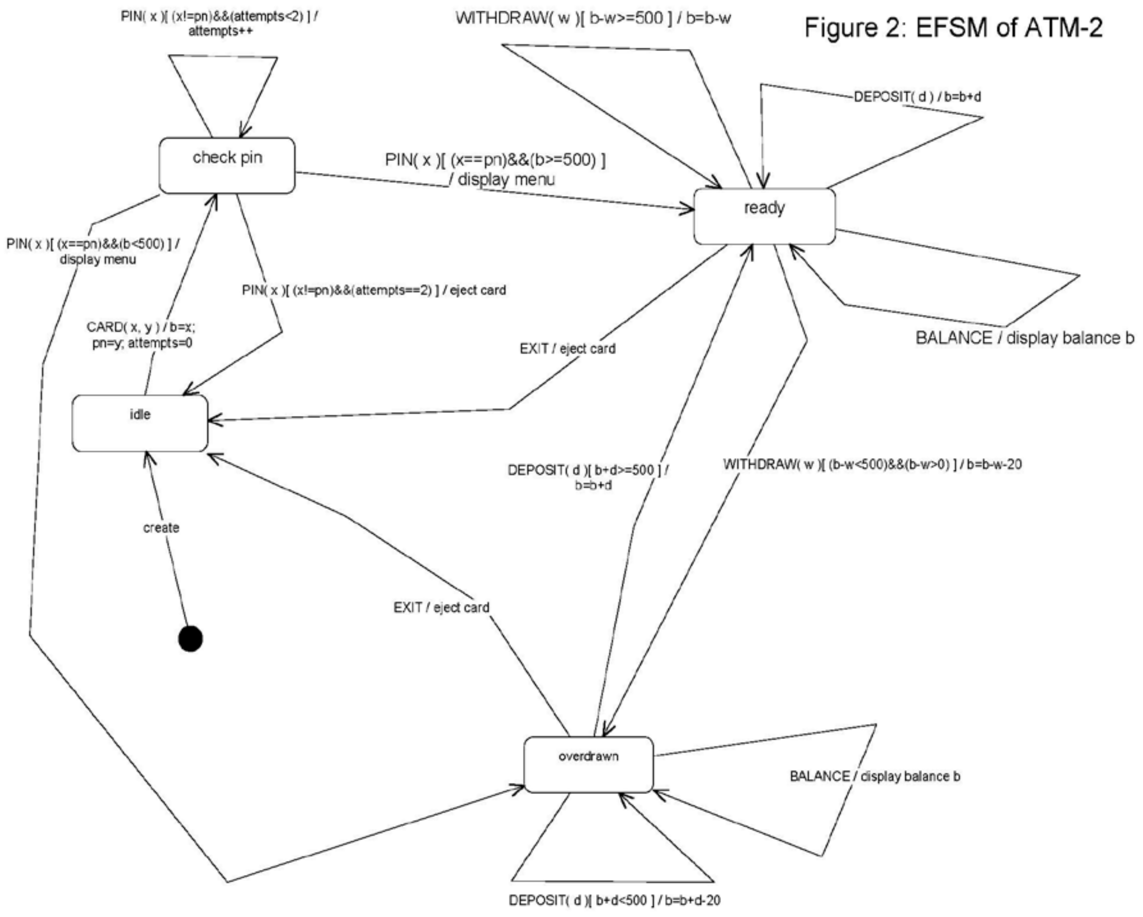

PIN(x)[(x!=pn)&&(attempts<2)] / attempts++
WITHDRAW(w)[b-w>=500] / b=b-w
DEPOSIT(d) / b=b+d
check pin
PIN(x)[(x==pn)&&(b>=500)] / display menu
ready
PIN(x)[(x==pn)&&(b<500)] / display menu
PIN(x)[(x!=pn)&&(attempts==2)] / eject card
CARD(x, y) / b=x; pn=y; attempts=0
EXIT / eject card
BALANCE / display balance b
idle
DEPOSIT(d)[b+d>=500] / b=b+d
WITHDRAW(w)[(b-w<500)&&(b-w>0)] / b=b-w-20
create
EXIT / eject card
overdrawn
BALANCE / display balance b
DEPOSIT(d)[b+d<500] / b=b+d-20

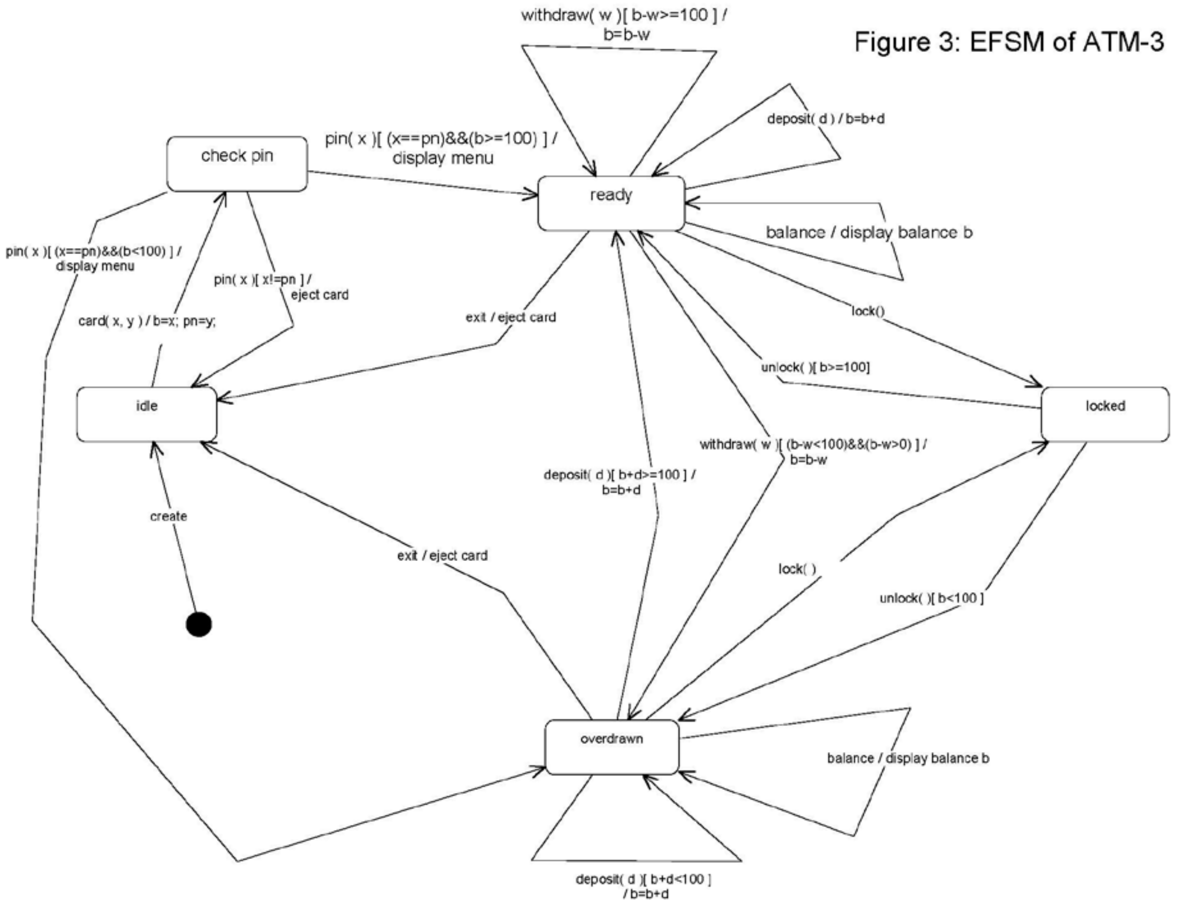
Figure 2: EFSM of ATM-2

Kapeel Daryani
A20313146

The ATM-3 component supports the following operations

• create ()	To create ATM
• card (int x, int y)	To begin the operation where x is the balance and y is pin
• pin (string x)	Pin is required to perform further operations on the ATM
• deposit (int d)	Deposits the amount d
• withdraw(int w)	Withdraws the amount w
• balance()	Displays the current balance
• lock()	Locks the ATM
• unlock()	Unlocks the ATM
• exit()	Exits the ATM

The following is the state diagram representing the flow for ATM-3


withdraw(w)[b-w>=100] / b=b-w
deposit(d) / b=b+d
check pin
pin(x)[(x==pn)&&(b>=100)] / display menu
ready
balance / display balance b
pin(x)[(x==pn)&&(b<100)] / display menu
pin(x)[x!=pn] / eject card
card(x, y) / b=x; pn=y;
exit / eject card
lock()
unlock( )[b>=100]
idle
locked
withdraw(w)[(b-w<100)&&(b-w>0)] / b=b-w
deposit(d)[b+d>=100] / b=b+d
create
exit / eject card
lock( )
unlock( )[b<100]
overdrawn
balance / display balance b
deposit(d)[b+d<100] / b=b+d
Figure 3: EFSM of ATM-3

Kapeel Daryani
A20313146

Model Driven Architecture Design

All the three EFSM (Extended Finite State Machine) of the ATMs are used to design a generalized MDA.

Events in MDA-EFSM

Following events are used in MDA design to represent operations.

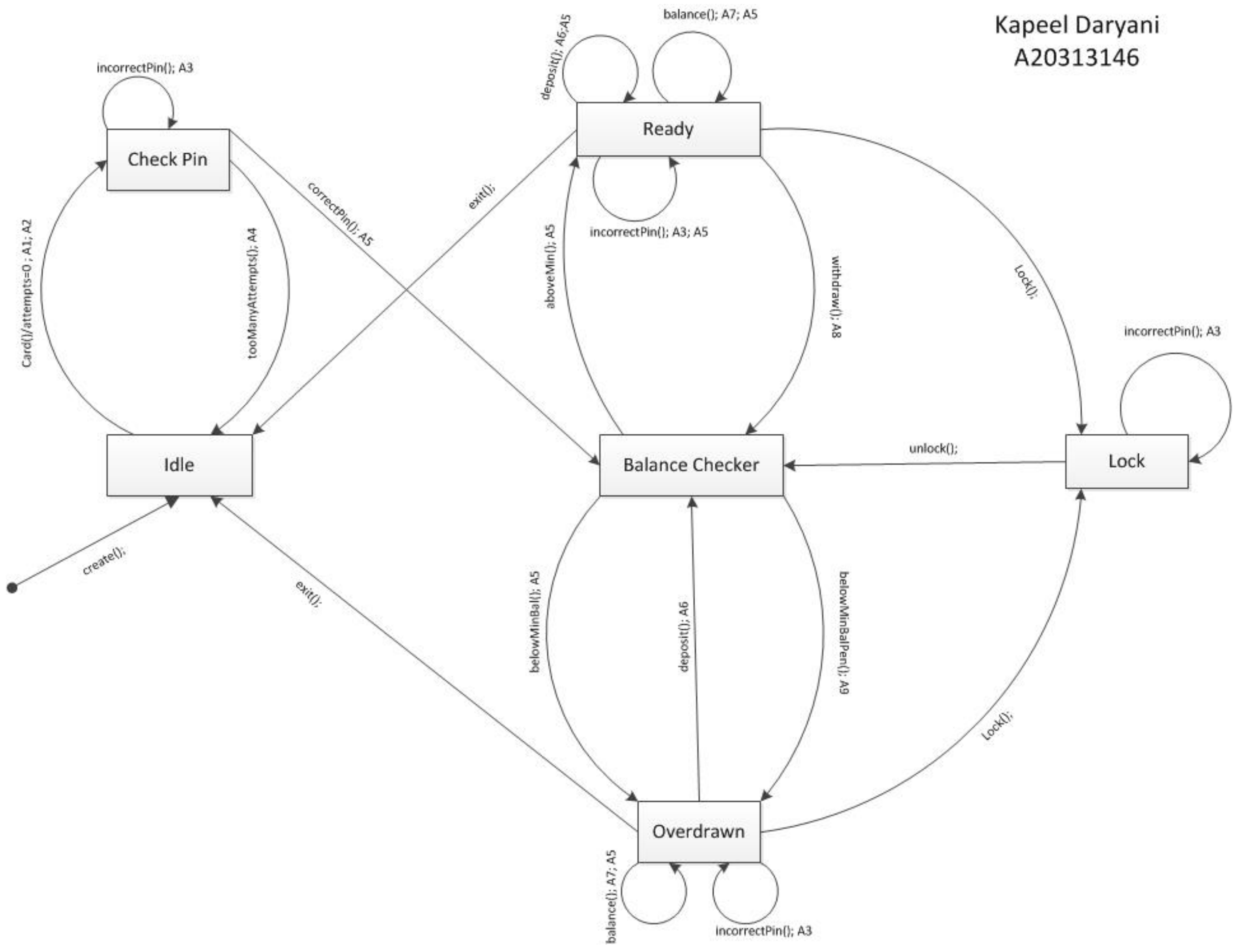
- create()
- card()
- correctPin()
- incorrectPin()
- tooManyAttempts()
- deposit()
- withdraw()
- balance()
- aboveMin()
- belowMinBal()
- belowMinBalPen()
- lock()
- unlock()
- exit()

Actions in MDA-EFSM

Following actions are used to represent the outputs of the ATMs

A1: Storedata()	The pin number and balance is stored here
A2: Promptforpin()	It asks to enter the pin
A3: Incorrectpinmessage()	Prints an error message for incorrect pin
A4: Toomanyattemptsmessage()	Prints an error message for attempting more than maximum and card rejection.
A5: Displaymenu()	Displays the menu.
A6: Makedeposit()	Adds the deposit amount to balance.
A7: Displaybalance()	displays the account balance.
A8: Makewithdrawal()	makes the withdraw by deducting the amount from balance.
A9: Chargepenalty()	charges the penalty by deducting it from the balance
A10: Displaybalbelowmin()	Prints an error message for balance being lower than minimum.

MDA - EFSM State Diagram



Pseudo Code for the ATMs

ATM 1

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

card (int x, string y)
{
store x -> bal;
store y -> pn;
attempts = 0;
m -> card();
}
```

```
pin (string x)
{
if(x == pn)
        m -> correctPin();
else{
        if(attempts<3)
                {
                m -> incorrectPin();
                attempts++;
                }
        else
                {
                 m -> tooManyAttempts();
                attempts=0;
                }
        }
}

deposit (int d)
{
store d;
m -> deposit();
if( bal >= 1000)
        m -> aboveMin();
else
        m -> belowMinBalPen();
}

withdraw(int w)
{
store w;
m -> withdraw();
if(bal >= 1000)
        m -> aboveMin();
else
        m -> belowMinBalPen();
}

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

```
lock(string x)
{
If (x==pn)
        m -> lock();
else
        m -> incorrectPin();
}

unlock(string x)
{
If( x==pn )
        m -> unlock();
        if ( bal>1000 )
                m -> aboveMin();
        else
                m -> belowMinBal();
else
        m -> incorrectPin();
}
exit()
{
m -> exit();
}
```

ATM - 2

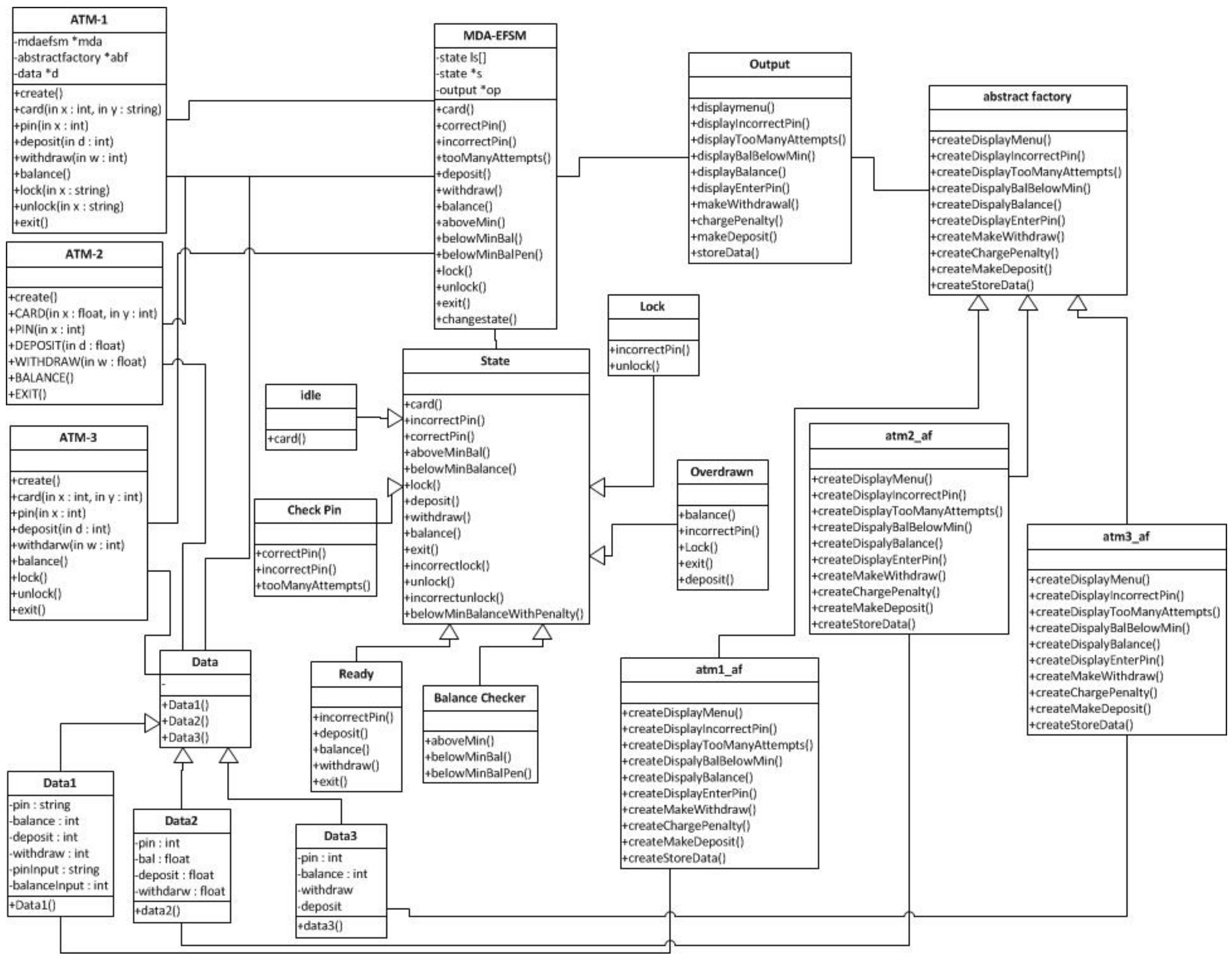
```
create()
{
m -> create();
}

CARD(float x, int y)
{
store x -> bal;
store y -> pn;
attempts = 0;
m -> card();
}

PIN(int x)
{
if (x == pn)
```

```
        m -> correctPin();
else{
        if(attempts<2)
                {
                m -> incorrectPin();
                attempts++;
                }else
                { m -> tooManyAttempts();
                attempts=0;
                }
            }
}

DEPOSIT(float d)
{
store d;
m -> deposit();
        if( bal >= 500)
                m -> aboveMin();
        else
                m -> belowMinBalPen();
}

WITHDRAW(float w)
{
store w;
m -> withdraw();
        if(bal  >= 500)
                m -> aboveMin();
        else
                m -> belowMinBalPen();
}

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

EXIT()
{
m -> exit();
}
```

ATM - 3

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

card(int x, int y)
{
store x -> bal;
store y -> pn;
attempts = 0;
m -> card();
}



pi n(string x)
{
if(x == pn)
        m -> correctPin();
else
        m -> tooManyAttempts();
}
deposit(int d)
{
store d;
m -> deposit();
        if( bal >= 100)
                m -> aboveMin();
        else
                m -> belowMinBal();
}

withdraw(int w)
{
        m -> withdraw();
                if(bal  >=100)
                m -> aboveMin();
                else
                m -> belowMinBal();
}

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

lock()
{
m -> lock();
}

unlock()
{
m -> unlock();
        if ( bal>100 )
                m -> aboveMin();
        else
                m -> belowMinBal(); }
exit()
{
m -> exit()
}
```

Data, State, Abstract and MDA-EFSM Class Diagram:



Data Classes:

Data class and its inherited classes are used to store the balance and pin of the ATM accounts. Following is the pseudo code for DS 1, DS 2, and DS 3.

Pseudo Code:

```
public Data1()
{
    •   Store balance and pin for ATM 1
    •   stores the minimum balance limit, penalty amount and maximum invalid
        attempts allowed for ATM 1
}
```

```
public Data2()
{
    •  Store balance and pin for ATM 2
    •  stores the minimum balance limit, penalty amount and maximum invalid
       attempts allowed for ATM 2
}

public Data3()
{
    •  Store balance and pin for ATM 1
    •  stores the minimum balance limit, penalty amount and maximum invalid
       attempts allowed for ATM 1
}
```

State Classes:

Idle, Check pin, Locked, Overdrawn, Ready and Balance Checker are inherited from the State class. These classes maintain the state of the system at a given point of time.

Pseudo Code:

```
public class idle extends state{

      public  idle(MDA_efsm m)
      {
            mda = m;
      }

      @Override
      public void card()
      {
          •  Change the state to CheckPin.
          •  Point to data class.
      }

      }

class checkpin
{
      public  checkpin(MDA_efsm m)
      {
            mda = m;
      }

      public void incorrectPin(int max)
      {
            if (attempts < max)
            {
                  attempts++;
                  op -> displayIncorrectPin();
```

```
		}
		else
		{
			changeState(0);
			op -> displayTooManyAttempts();
			attempts = 0;
		}
	}

	public void correctPin()
	{
		changeState(2);
		op -> displayMenu();
	}

class locked

	public void incorrectLock()
	{
		op -> displayIncorrectPin();
	}


	public void unlock()
	{
		changeState(2);
		println("You unlocked your account.");
	}


	public void incorrectUnlock()
	{
		op -> displayIncorrectPin();
	}
}

public class overdrawn extends state{

	public void deposit()
	{
		changeState(2);
		op -> makeDeposit();
	}


	public void withdraw()
	{
		op -> dispalyBalBelowMin();
	}

	@Override
	public void balance()
	{
		op -> displayBalance();
	}
```

```
		@Override
		public void lock()
		{
			changeState(5);
			println("You locked your account.");
		}


		public void incorrectLock()
		{
			op -> displayIncorrectPin();
		}


		public void exit()
		{
			changeState(0);
			println("Card ejected");

		}


		public void incorrectPin(int max)
		{
			op -> displayIncorrectPin();
		};

	}

	public class ready extends state{

		public void deposit()
		{
			op -> makeDeposit();

		}

		@Override
		public void withdraw()
		{
			op -> makeWithdrawal();
			changeState(2);

		}


		public void balance()
		{
			op -> displayBalance();
		}


		public void lock()
		{
			changeState(5);
			println("You locked your account.");
	}


	public void exit()
	{
		changeState(0);
		println("Card ejected");
	}


	public void incorrectLock()
	{
		op -> displayIncorrectPin();
	}
}

public class balanceChecker extends state{

	public void aboveMinBalance()
	{
		changeState(3);
	}


	public void belowMinBalance()
	{
		changeState(4);
	}

	public void belowMinBalanceWithPanelty()
	{
		changeState(4);
		op -> chargePenalty();
	}
}
```

Abstract Factory Classes:

Abstract factory classes collaborate the output with MDA-EFSM and data. It helps in flow of data and activates the output to printed.

Pseudo Code:

```
class ATM_Factory1

	public DisplayMenu createDisplayMenu()
	{
	return method for displaying the menu
	}


	public Incorrectpinmessage createDisplayIncorrectPin()
	{
```

```
	return method for error message if wrong pin is entered for ATM 1
	}


	public Toomanyattemptsmessage createDisplayTooManyAttempts()
	{
	return method for message if certain no of tries attempt for pin 
against ATM 1
	}


	public DispalyBalBelowMin createDispalyBalBelowMin()
	{
	return Method for error message if there is minimum balance and 
withdrawn action is made in ATM 1
	}


	public DisplayBalance createDispalyBalance()
	{
	return Method for display the current balance after transaction in ATM 
1
	}


	public Promptforpin createDisplayEnterPin()
	{
	return method that display message if pin is not entered after card is 
applied in ATM 1
	}


	public MakeWithdrawal createMakeWithdraw()
	{
	 return method for action to withdraw amount from current balance 
against ATM 1
	}


	public ChargePenalty createChargePenalty()
	{
	return method for charge the penalty if current balance is below 
minimum amount in ATM 1
	}


	public MakeDeposit createMakeDeposit()
	{
	//return method for add or deposit amount in current balance from ATM 1
	}


	public StoreData createStoreData()
	{
	return method, for store pin and opening balance for ATM 1
	}
}
```

```
class ATM_Factory2

      public DisplayMenu createDisplayMenu()
      {
      return method for displaying the menu for ATM 2
      }


      public Incorrectpinmessage createDisplayIncorrectPin()
      {
      return method for error message if wrong pin is entered for ATM 2
      }


      public Toomanyattemptsmessage createDisplayTooManyAttempts()
      {
      return method for message if certain no of tries attempt for pin
against ATM 2
      }


      public DispalyBalBelowMin createDispalyBalBelowMin()
      {
      return Method for error message if there is minimum balance and
withdrawn action is made in ATM 2
      }


      public DisplayBalance createDispalyBalance()
      {
      return Method for display the current balance after transaction in ATM2
      }


      public Promptforpin createDisplayEnterPin()
      {
      return method that display message if pin is not entered after card is
applied in ATM 2
      }


      public MakeWithdrawal createMakeWithdraw()
      {
      return method for action to withdraw amount from current balance
against ATM 2
      }


      public ChargePenalty createChargePenalty()
      {
      return method for charge the penalty if current balance is below
minimum amount in ATM 2
      }


      public MakeDeposit createMakeDeposit()
```

```
        {
        return method for add or deposit amount in current balance from ATM 2
        }


        public StoreData createStoreData()
        {
        return method, for store pin and opening balance for ATM 2
        }
}

class ATM_Factory3

        public DisplayMenu createDisplayMenu()
        {
        return method for displaying the menu for ATM 3
        }


        public Incorrectpinmessage createDisplayIncorrectPin()
        {
        return method for error message if wrong pin is entered for ATM 3
        }


        public Toomanyattemptsmessage createDisplayTooManyAttempts()
        {
        return method for message if certain no of tries attempt for pin
against ATM 3
        }


        public DispalyBalBelowMin createDispalyBalBelowMin()
        {
        return Method for error message if there is minimum balance and
withdrawn action is made in ATM 3
        }


        public DisplayBalance createDispalyBalance()
        {
        return Method for display the current balance after transaction in ATM3
        }


        public Promptforpin createDisplayEnterPin()
        {
        return method that display message if pin is not entered after card is
applied in ATM 3
        }


        public MakeWithdrawal createMakeWithdraw()
        {
        return method for action to withdraw amount from current balance
against ATM 3
        }
        public ChargePenalty createChargePenalty()
        {
        return method for charge the penalty if current balance is below
minimum amount in ATM 3
        }


        public MakeDeposit createMakeDeposit()
        {
        return method for add or deposit amount in current balance from ATM 3
        }


        public StoreData createStoreData()
        {
        return method, for store pin and opening balance for ATM 3
        }
}
```

MDA-EFSM Class:

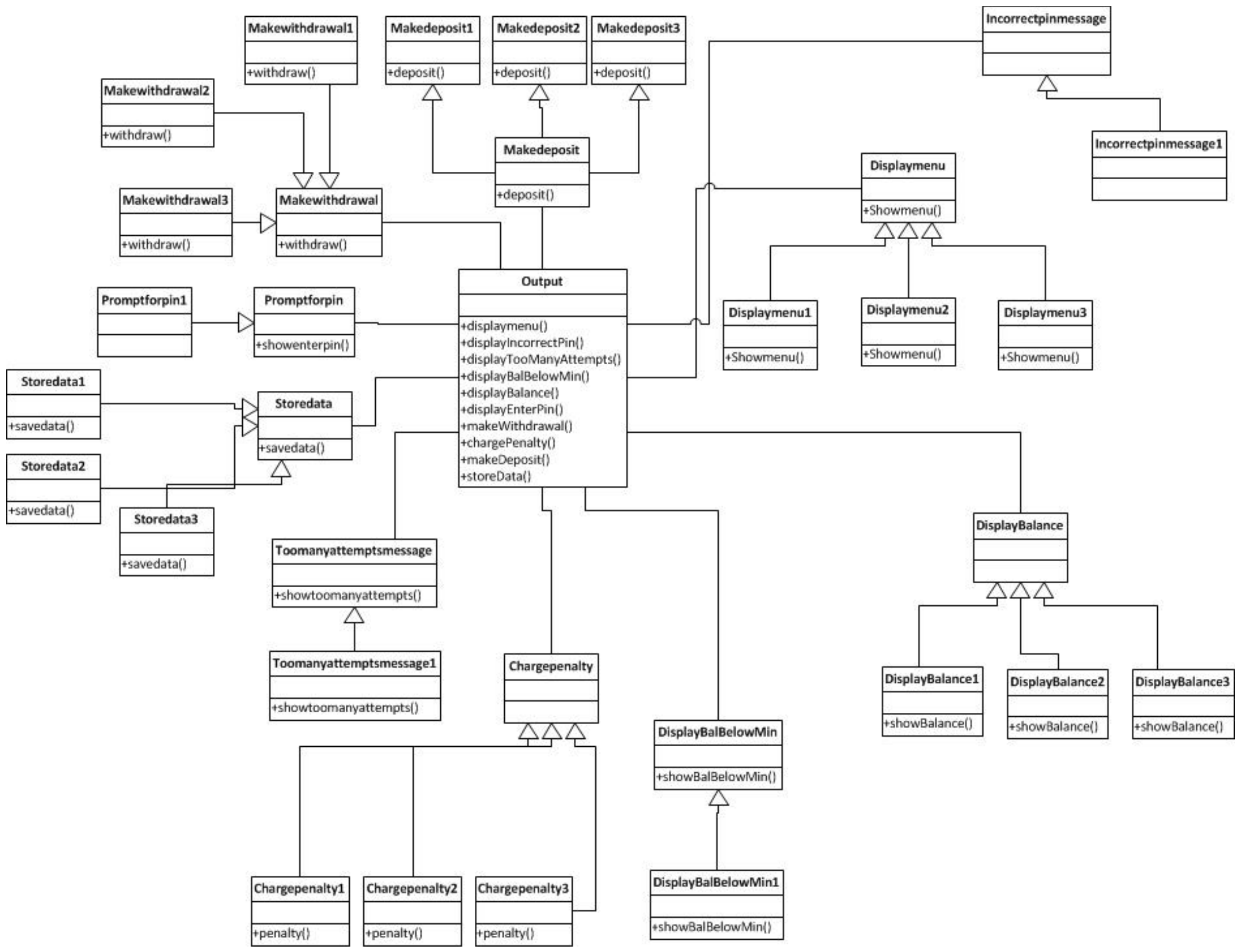
Provides a generalized finite state machine for the three ATMs. It takes events from ATM 1, ATM 2 and ATM 3 and generate the corresponding action through output class.

Pseudo Code:

```
public class MDA_efsm {
        state s;

                sList[0] = new idle(this);
                sList[1] = new checkpin(this);
                sList[2] = new balanceChecker(this);
                sList[3] = new ready(this);
                sList[4] = new overdrawn(this);
                sList[5] = new locked(this);
                s = sList[0];
        }

        public void card()
        {
                s -> card();
        }

        public void correctPin()
        {
                s -> correctPin();
        }

        public void incorrectPin(int max)
        {
                s -> incorrectPin(max);
}

public void aboveMinBalance()
{
	s -> aboveMinBalance();
}

public void belowMinBalance()
{
	s -> belowMinBalance();
}


public void exit()
{
	s -> exit();
}

public void withdraw()
{
	s -> withdraw();
}

public void belowMinBalanceWithPanelty()
{
	s -> belowMinBalanceWithPanelty();
}

public void deposit()
{
	s -> deposit();
}

public void balance()
{
	s -> balance();
}

public void lock()
{
	s -> lock();
}

public void incorrectLock()
{
	s -> incorrectLock();
}

public void unLock()
{
	s -> unlock();
}

public void incorrectUnLock()
{
	s -> incorrectUnlock();
}
```

```
public void changeState(int state)
{
        s = sList[state];
}
```

Output and Actions Class Diagram:



Output Class:

The output class receives actions to be executed from the MDA-EFSM class and then through abstract factory calls the action classes.

Pseudo Code:

```
public class output {

      public ATM_Factory af; //initialize ATM_Factory Interface as af


      public void displayMenu()
      {
      Shows the transaction menu for particular ATM
      }


      public void displayIncorrectPin()
      {
      display error message if wrong pin is entered for ATM
      }


      public void displayTooManyAttempts()
      {
      message shows if certain no of tries attempt for correct pin.
      }


      public void dispalyBalBelowMin()
      {
      error message if there is minimum balance and withdrawn action is made
      }


      public void displayBalance()
      {
      display the current balance after transaction
      }


      public void displayEnterPin()
      {
      display message if pin is not entered after card is applied.
      }


      public void makeWithdrawal()
      {
      action for withdraw amount from current balance
      }


      public void chargePenalty()
      {
      charge the penalty if current balance is below minimum amount
      }


      public void makeDeposit()
      {
```

```
      add or deposit amount in current balance
      }


      public void storeData()
      {
      store pin and opening balance for ATM
      }
}
```

Action Classes:

Performs the actions required by the output class.

Pseudo Code:

```
class ChargePenalty1
      {
      public void penalty()
      {
      subtracts the penalty amount from current balance for ATM 1
      }
}

public class ChargePenalty2
      {
            public void penalty()
            {
      subtracts the penalty amount from current balance for ATM 2
            }
      }

public class ChargePenalty3
{
      public void penalty()
      {
      //subtracts the penalty amount from current balance for ATM 3
      }
}


public class DispalyBalBelowMin1
{
      public void showBalBelowMin()
      {
      Error message for withdraw amount if current balance is below minimum
balance
      }

}

public class DisplayBalance1
{
```

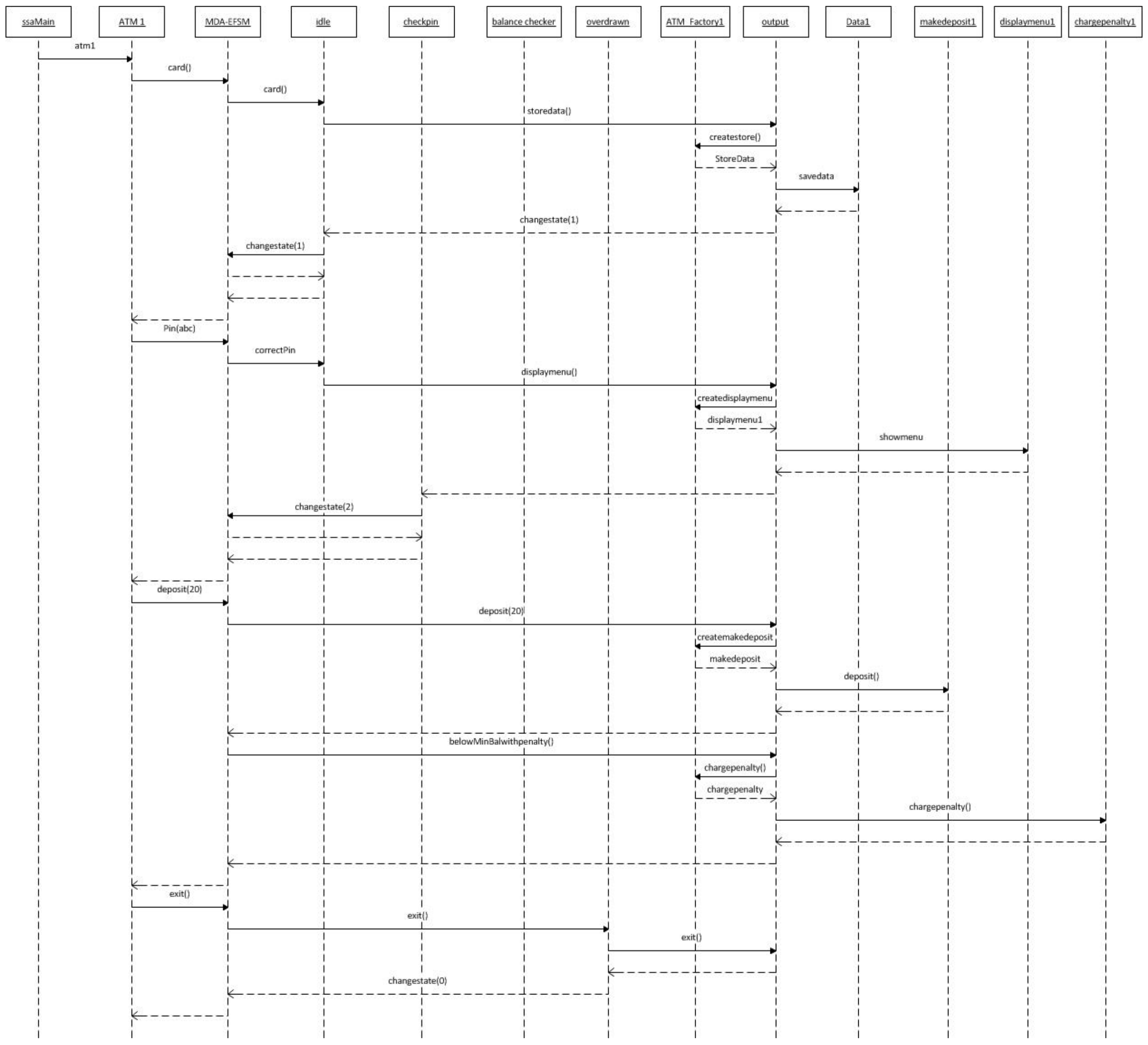
```
            public void showBalance()
            {
            Show the current balance for ATM 1
            }
}

public class DisplayBalance2
{
            public void showBalance()
            {
            Show the current balance for ATM 2
            }
}

public class DisplayBalance3
{
            public void showBalance()
            {
            Show the current balance for ATM 3
            }
}

public  class DisplayMenu1 {

            public void showMenu()
            {
            show the transaction menu for ATM 1 and 3
            }
      }

public class DisplayMenu2{

      public void showMenu()
      {
      show the transaction menu for ATM 2
      }
}

public class Incorrectpinmessage1
{

      public void ShowIncorrectPin()
      {
      show message if incorrect message entered.
      }

}

      public class MakeDeposit1
      {
            public void deposit()
                  {
            add or deposit amount in current balance for ATM 1
                  }

      }
```

```
	public class MakeDeposit2
	{
		public void deposit()
			{
		add or deposit amount in current balance for ATM 2
			}

	}
	public class MakeDeposit3
	{
		public void deposit()
			{
		add or deposit amount in current balance for ATM 3
			}

	}

public class MakeWithdrawal1
{

	public void withdraw()
	{
		withdraw amount from current balance for ATM 1
	}

}

public class MakeWithdrawal2
{

	public void withdraw()
	{
		withdraw amount from current balance for ATM 2
	}

}
public class MakeWithdrawal3
{

	public void withdraw()
	{
		withdraw amount from current balance for ATM 3
	}

}

public class Promptforpin1
{
	public void showEnterPin()
	{
		show message for operation enter pin number
	}

}

public class StoreData1
{
      public void saveData()
      {
      store pin and opening balance for ATM 1
      }

}

public class StoreData2
{

      public void saveData()
      {
      store pin and opening balance for ATM 2
      }

}
public class StoreData3
{

      public void saveData()
      {
      store pin and opening balance for ATM 3
      }

}

public class Toomanyattemptsmessage1
 {
      public void showTooManyAttempts()
      {
      message shows if certain no of tries attempt for correct pin.
      }
}
```

Dynamics

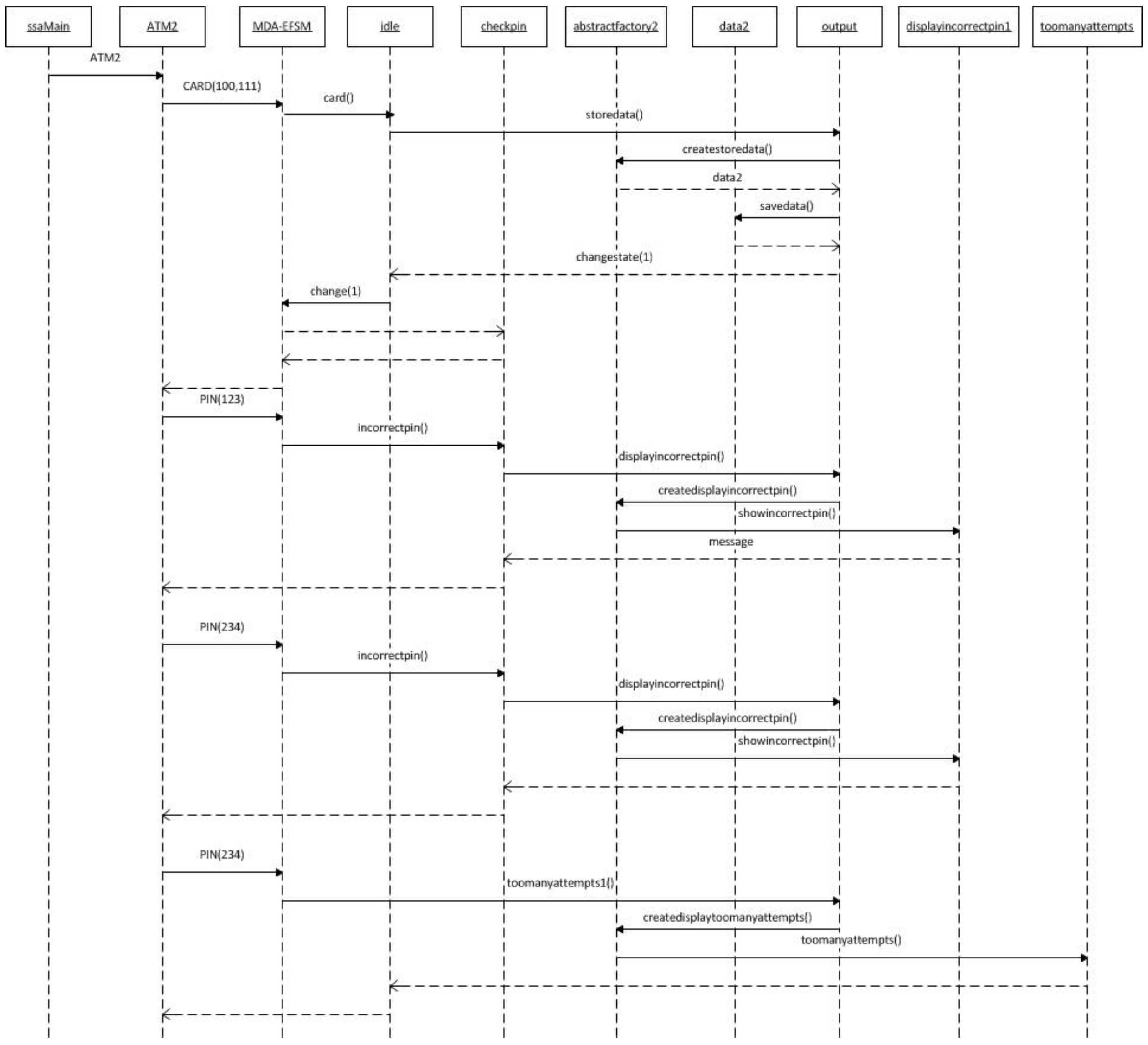
Scenario I:

card(100,"abc"), pin("abc"), deposit(20), exit()



Scenario II:

CARD(100,111), PIN(123), PIN(234), PIN(345)



Source Code and Patterns:

The following is the list of classes with the pattern they follow.

State Pattern:

- State class
- idle
- ready
- check pin
- locked
- overdrawn
- balance checker

Abstract Factory Pattern:

- ATM_Factory
- ATM_Factory1
- ATM_Factory2
- ATM_Factory3

Strategy Pattern:

The action classes are designed in strategy pattern.

- ChargePenalty
- ChargePenalty1
- ChargePenalty2
- ChargePenalty3
- DisplayBalance
- DisplayBalance1
- DisplayBalance2
- DisplayBalance3
- MakeDeposit
- MakeDeposit1
- MakeDeposit2
- MakeDeposit3
- MakeWithdrawal
- MakeWithdrawal1
- MakeWithdrawal2
- MakeWithdrawal3
- StoreData
- StoreData1
- StoreData2
- StoreData3

Source Code

```
package Kapeel586;

import Kapeel586.State.*;

public class MDA_efsm {
	public state s;
	public state[] sList = new state[6];
	public int attempts	= 0;
	public output op = new output();

	public MDA_efsm()
	{
		sList[0] = new idle(this);
		sList[1] = new checkpin(this);
		sList[2] = new balanceChecker(this);
		sList[3] = new ready(this);
		sList[4] = new overdrawn(this);
		sList[5] = new locked(this);
		s = sList[0];
	}

	public void card()
	{
		s.card();
	}

	public void correctPin()
	{
		s.correctPin();
	}

	public void incorrectPin(int max)
	{
		s.incorrectPin(max);
	}

	public void aboveMinBalance()
	{
		s.aboveMinBalance();
	}

	public void belowMinBalance()
	{
		s.belowMinBalance();
	}


	public void exit()
	{
		s.exit();
	}

	public void withdraw()
	{
		s.withdraw();
	}

	public void belowMinBalanceWithPanelty()
	{
		s.belowMinBalanceWithPanelty();
	}

	public void deposit()
	{
		s.deposit();
	}

	public void balance()
	{
		s.balance();
	}

	public void lock()
	{
		s.lock();
	}

	public void incorrectLock()
	{
		s.incorrectLock();
	}

	public void unLock()
	{
		s.unlock();
	}

	public void incorrectUnLock()
	{
		s.incorrectUnlock();
	}

	public void changeState(int state)
	{
		s = sList[state];
	}

}

package Kapeel586;

import Kapeel586.AbstractFactory.ATM_Factory;

public class output {
	//initialize ATM_Factory Interface as af
	public ATM_Factory af;

	//Shows the transaction menu for perticular ATM
	public void displayMenu()
	{
		af = ssaMain.af;
		af.createDisplayMenu().showMenu();
	}

	//display error message if wrong pin is entered for ATM
	public void displayIncorrectPin()
	{
		af = ssaMain.af;
		af.createDisplayIncorrectPin().ShowIncorrectPin();
	}

	//message shows if certain no of tries attempt for correct pin.
	public void displayTooManyAttempts()
	{
		af = ssaMain.af;
		af.createDisplayTooManyAttempts().showTooManyAttempts();
	}

	//error message if there is minimum balance and withdrawn action is
made
	public void dispalyBalBelowMin()
	{
		af = ssaMain.af;
		af.createDispalyBalBelowMin().showBalBelowMin();
	}

	//display the current balance after transaction
	public void displayBalance()
	{
		af = ssaMain.af;
		af.createDispalyBalance().showBalance();
	}

	//display message if pin is not entered after card is applied.
	public void displayEnterPin()
	{
		af = ssaMain.af;
		af.createDisplayEnterPin().showEnterPin();
	}

	// action for withdraw amount from current balance
	public void makeWithdrawal()
	{
		af = ssaMain.af;
		af.createMakeWithdraw().withdraw();
	}

	//charge the penalty if current balance is below minimum amount
	public void chargePenalty()
	{
		af = ssaMain.af;
		af.createChargePenalty().penalty();
	}

	//add or deposit amount in current balance
	public void makeDeposit()
	{
		af = ssaMain.af;
		af.createMakeDeposit().deposit();
	}

	//store pin and opening balance for ATM
	public void storeData()
	{
		af = ssaMain.af;
		af.createStoreData().saveData();
	}
}
```

```
package Kapeel586;

import java.util.Scanner;
import Kapeel586.AbstractFactory.ATM_Factory;
import Kapeel586.AbstractFactory.ATM_Factory1;
import Kapeel586.AbstractFactory.ATM_Factory2;
import Kapeel586.AbstractFactory.ATM_Factory3;
import Kapeel586.ATMS.atm1;
import Kapeel586.ATMS.atm2;
import Kapeel586.ATMS.atm3;
import Kapeel586.Data.*;

public class ssaMain {

	public static ATM_Factory af;
	public static Data1				d1;
	public static Data2				d2;
	public static Data3				d3;
	public static  String			tmp_pin1;//atm1

	public static  int			tmp_balance1;
	public static int			tmp_deposit1;
	public static int			tmp_withdraw1;
	public static int		tmp_pin2;//atm2
	public static float			tmp_balance2;
	public static float			tmp_deposit2;
	public static float			tmp_withdraw2;
	public static int		tmp_pin3;//atm3
	public static int			tmp_balance3;
	public static int			tmp_deposit3;
	public static int			tmp_withdraw3;
	public static int			ch;
	public static int		flag;



	public static void main(String[] args)
```

```
{
        System.out.println("Software System Architecture - CS586 ");
        System.out.println("                              Project");
        System.out.println();
        System.out.println("Submitted By: Kapeel Daryani - A20313146");
        System.out.println();
        Scanner scan = new Scanner(System.in);
        while (true)
        {
                System.out.println("Select from ATM-1 or ATM-2 or ATM-3 to be used");
                System.out.println();
                System.out.println("Press 1 for ATM-1");
                System.out.println("Press 2 for ATM-2");
                System.out.println("Press 3 for ATM-3");
                System.out.println("Press 4 to quit from the program");
                System.out.println();
                System.out.println("Enter the value:");

                ch = scan.nextInt();
                if (ch == 4)
                {
                        System.out.print("Operation: Quit ");
                        System.exit(0);
                        break;
                }
                else
                {
                        if (ch == 1)
                        {
                                flag=1;
                                atm1 atm1 = new atm1();
                                af = new ATM_Factory1();
                                d1 = new Data1();

                                String pin;
                                int balance;
                                int deposit;
                                int withdraw;

                                System.out.println();
                                System.out.println("      ATM-1");
                                System.out.println("  MENU of Operations");
                                System.out.println();
                                System.out.println("   0. card(int,string)");
                                System.out.println("   1. pin(string)");
                                System.out.println("   2. deposit(int)");
                                System.out.println("   3. withdraw(int)");
                                System.out.println("   4. balance()");
```

```
					System.out.println("   5. lock(String)");
					System.out.println("   6. unlock(String)");
					System.out.println("   7. exit()");
					System.out.println("   8. closeSystem()");
					System.out.println();

					while (ch != 8)
					{
						System.out.println("      ATM-1 Execution    ");
						System.out.println();
						System.out.println("  Select Operation:");
						System.out
								.println("0-card,1-pin,2-deposit,3-
withdraw,4-balance,5-lock,6-unlock,7-exit");

						ch = scan.nextInt();
						System.out.println();
						switch (ch)
						{
						case 0:
							System.out.print("Operation:  card (int x, string
y) ");
							System.out.print("Enter value of the balance x:
");
							balance = scan.nextInt();
							System.out.print("Enter value of the pin p: ");
							pin = scan.next();
							atm1.card(balance,pin);
							scan.nextLine();
							System.out.println();
							break;


						case 1:
							System.out.print("Operation:  pin(String x) ");
							System.out.print("Enter value of the pin x ");
							pin = scan.next();
							atm1.pin(pin);
							scan.nextLine();
							System.out.println();
							break;

						case 2:
							System.out.print("Operation:  deposit(int d)");
							System.out.print("Enter value of the money you
want to deposit d: ");
							deposit = scan.nextInt();
							atm1.deposit(deposit);
```

```
                        scan.nextLine();
                        System.out.println();
                        break;

                case 3:
                        System.out.print("Operation:  withdraw(int w)
");

                        System.out.print("Enter value of the money you
want to withdraw w: ");

                        withdraw = scan.nextInt();
                        atm1.withdraw(withdraw);
                        scan.nextLine();
                        System.out.println();
                        break;

                case 4:
                        System.out.print("Operation:  balance() ");
                        atm1.balance();
                        scan.nextLine();
                        System.out.println();
                        break;


                case 5:
                        System.out.print("Operation:  lock(String x) ");
                        System.out.print(" Enter value of the pin x: ");
                        pin = scan.next();
                        atm1.lock(pin);
                        scan.nextLine();
                        System.out.println();
                        break;

                case 6:
                        System.out.print("Operation:  unlock(String x)
");

                        System.out.print(" Enter value of the pin x: ");
                        pin = scan.next();
                        atm1.unlock(pin);
                        scan.nextLine();
                        System.out.println();
                        break;

                case 7:
                        System.out.print("Operation: EXIT, Card
Ejected, Press 0 to insert card again.");

                        atm1.exit();
                        System.out.println();
                        break;
```

```
					case 8:
						System.out.print("Operation: System Closed");
						System.exit(0);
						break;



					}
				}
			}
			else if (ch == 2)
			{
				flag=2;
				atm2 atm2 = new atm2();
				af = new ATM_Factory2();
				d2 = new Data2();

				System.out.println();
				System.out.println("      ATM-2");
				System.out.println();
				System.out.println("  MENU of Operations");
				System.out.println();
				System.out.println("   0. CARD(float,int)");
				System.out.println("   1. PIN(int)");
				System.out.println("   2. DEPOSIT(float)");
				System.out.println("   3. WITHDRAW(float)");
				System.out.println("   4. BALANCE()");
				System.out.println("   5. EXIT()");
				System.out.println();

				int pin;
				float balance;
				float deposit;
				float withdraw;

				while (ch != 5)
				{
					System.out.println("      ATM-2 Execution");
					System.out.println();
					System.out.println("     Select Operation:");
					System.out.println("0-Card,1-PIN,2-DEPOSIT,3-
WITHDRAW,4-BALANCE,5-EXIT");

					ch = scan.nextInt();
					System.out.println();
					switch (ch)
					{
```

```
                              case 0:
                                   System.out.print("Operation:  card (float x, int
y) ");
                                   System.out.print("Enter value of the balance x:
");
                                   balance = scan.nextFloat();
                                   System.out.print("Enter value of the pin p: ");
                                   pin = scan.nextInt();
                                   atm2.CARD(balance,pin);
                                   scan.nextLine();
                                   System.out.println();
                                   break;

                              case 1:
                                   System.out.print("Operation:  PIN(int x) ");
                                   System.out.print("Enter value of the PIN x ");
                                   pin = scan.nextInt();
                                   atm2.PIN(pin);
                                   System.out.println();
                                   break;

                              case 2:
                                   System.out.print("Operation:  DEPOSIT(float
d)");
                                   System.out.print("Enter value of the money you
want to deposit d: ");
                                   deposit = scan.nextFloat();
                                   atm2.DEPOSIT(deposit);
                                   scan.nextLine();
                                   System.out.println();
                                   break;

                              case 3:
                                   System.out.print("Operation:  WITHDRAW(float
w) ");
                                   System.out.print("Enter value of the money you
want to withdraw w: ");
                                   withdraw = scan.nextFloat();
                                   atm2.WITHDRAW(withdraw);
                                   scan.nextLine();
                                   System.out.println();
                                   break;

                              case 4:
                                   System.out.print("Operation:  BALANCE() ");
                                   atm2.BALANCE();
                                   System.out.println();
                                   break;
```

```
                        case 5:
                                System.out.print("Operation: EXIT, Card
Ejected");

                                atm2.exit();
                                break;



                        }
                }
        }
        if (ch == 3)
        {
                flag=3;
                atm3 atm3 = new atm3();
                af = new ATM_Factory3();
                d3 = new Data3();

                int pin;
                int balance;
                int deposit;
                int withdraw;

                System.out.println();
                System.out.println("      ATM-3");
                System.out.println("  MENU of Operations");
                System.out.println();
                System.out.println("   0. card(int,int)");
                System.out.println("   1. pin(int)");
                System.out.println("   2. deposit(int)");
                System.out.println("   3. withdraw(int)");
                System.out.println("   4. balance()");
                System.out.println("   5. lock()");
                System.out.println("   6. unlock()");
                System.out.println("   7. exit()");
                System.out.println();

                while (ch != 8)
                {
                        System.out.println("      ATM-3 Execution    ");
                        System.out.println();
                        System.out.println("  Select Operation:");
                        System.out
                                        .println("0-card,1-pin,2-deposit,3-
withdraw,4-balance,5-lock,6-unlock,7-exit");

                        ch = scan.nextInt();
```

```
                              System.out.println();
                              switch (ch)
                              {
                              case 0:
                                  System.out.print("Operation:  card (int x, int y)
");

                                  System.out.print("Enter value of the balance x:
");

                                  balance = scan.nextInt();
                                  System.out.print("Enter value of the pin p: ");
                                  pin = scan.nextInt();
                                  atm3.card(balance,pin);
                                  scan.nextLine();
                                  System.out.println();
                                  break;



                              case 1:
                                  System.out.print("Operation:  pin(int x) ");
                                  System.out.print("Enter value of the pin x ");
                                  pin = scan.nextInt();
                                  atm3.pin(pin);
                                  scan.nextLine();
                                  System.out.println();
                                  break;

                              case 2:
                                  System.out.print("Operation:  deposit(int d)");
                                  System.out.print("Enter value of the money you
want to deposit d: ");

                                  deposit = scan.nextInt();
                                  atm3.deposit(deposit);
                                  scan.nextLine();
                                  System.out.println();
                                  break;

                              case 3:
                                  System.out.print("Operation:  withdraw(int w)
");

                                  System.out.print("Enter value of the money you
want to withdraw w: ");

                                  withdraw = scan.nextInt();
                                  atm3.withdraw(withdraw);
                                  scan.nextLine();
                                  System.out.println();
                                  break;

                              case 4:
```

```
                                                System.out.print("Operation:  balance() ");
                                                atm3.balance();
                                                scan.nextLine();
                                                System.out.println();
                                                break;


                                        case 5:
                                                System.out.print("Operation:  lock() ");
                                                atm3.lock();
                                                scan.nextLine();
                                                System.out.println();
                                                break;

                                        case 6:
                                                System.out.print("Operation:  unlock() ");
                                                atm3.unlock();
                                                scan.nextLine();
                                                System.out.println();
                                                break;

                                        case 7:
                                                System.out.print("Operation: EXIT, Card
Ejected, Press 0 to insert card again. ");

                                                atm3.exit();
                                                break;

                                        case 8:
                                                System.out.print("Operation: Close System");
                                                System.exit(0);
                                                break;
                                        }
                                }
                        }
                }
        }
    }
}

package Kapeel586.AbstractFactory;

import Kapeel586.Output.Chargepenalty.ChargePenalty;
import Kapeel586.Output.Displaybalance.DisplayBalance;
import Kapeel586.Output.Displaymenu.DisplayMenu;
import Kapeel586.Output.Incorrectpinmessage.Incorrectpinmessage;
import Kapeel586.Output.Makedeposit.MakeDeposit;
import Kapeel586.Output.Makewithdrawal.MakeWithdrawal;
import Kapeel586.Output.Promptforpin.Promptforpin;
import Kapeel586.Output.Toomanyattemptsmessage.Toomanyattemptsmessage;
import Kapeel586.Output.dispalybalbelowmin.DispalyBalBelowMin;
import Kapeel586.Output.storedata.StoreData;


public interface ATM_Factory {
	public DisplayMenu createDisplayMenu();

	public Incorrectpinmessage createDisplayIncorrectPin();

	public Toomanyattemptsmessage createDisplayTooManyAttempts();

	public DispalyBalBelowMin createDispalyBalBelowMin();

	public DisplayBalance createDispalyBalance();

	public Promptforpin createDisplayEnterPin();

	public MakeWithdrawal createMakeWithdraw();

	public ChargePenalty createChargePenalty();

	public MakeDeposit createMakeDeposit();

	public StoreData createStoreData();
}
package Kapeel586.AbstractFactory;

import Kapeel586.Output.Chargepenalty.ChargePenalty;
import Kapeel586.Output.Chargepenalty.ChargePenalty1;
import Kapeel586.Output.Displaybalance.DisplayBalance;
import Kapeel586.Output.Displaybalance.DisplayBalance1;
import Kapeel586.Output.Displaymenu.DisplayMenu;
import Kapeel586.Output.Displaymenu.DisplayMenu1;
import Kapeel586.Output.Incorrectpinmessage.Incorrectpinmessage;
import Kapeel586.Output.Incorrectpinmessage.Incorrectpinmessage1;
import Kapeel586.Output.Makedeposit.MakeDeposit;
import Kapeel586.Output.Makedeposit.MakeDeposit1;
import Kapeel586.Output.Makewithdrawal.MakeWithdrawal;
import Kapeel586.Output.Makewithdrawal.MakeWithdrawal1;
import Kapeel586.Output.Promptforpin.Promptforpin;
import Kapeel586.Output.Promptforpin.Promptforpin1;
import Kapeel586.Output.Toomanyattemptsmessage.Toomanyattemptsmessage;
import Kapeel586.Output.Toomanyattemptsmessage.Toomanyattemptsmessage1;
import Kapeel586.Output.dispalybalbelowmin.DispalyBalBelowMin;
import Kapeel586.Output.dispalybalbelowmin.DispalyBalBelowMin1;
import Kapeel586.Output.storedata.StoreData;
import Kapeel586.Output.storedata.StoreData1;
```

```
public class ATM_Factory1 implements ATM_Factory {

	public DisplayMenu createDisplayMenu()
	{
		return new DisplayMenu1();
	}

	//return method for error message if wrong pin is entered for ATM 1
	public Incorrectpinmessage createDisplayIncorrectPin()
	{
		return new Incorrectpinmessage1();
	}

	//return method for message if certain no of tries attempt for pin against ATM 1
	public Toomanyattemptsmessage createDisplayTooManyAttempts()
	{
		return new Toomanyattemptsmessage1();
	}

	//return Method for error message if there is minimum balance and withdrawn action is made
in ATM 1
	public DispalyBalBelowMin createDispalyBalBelowMin()
	{
		return new DispalyBalBelowMin1();
	}

	//return Method for display the current balance after transaction in ATM 1
	public DisplayBalance createDispalyBalance()
	{
		return new DisplayBalance1();
	}

	//return method that display message if pin is not entered after card is applied in ATM 1
	public Promptforpin createDisplayEnterPin()
	{
		return new Promptforpin1();
	}

	// return method for action to withdraw amount from current balance against ATM 1
	public MakeWithdrawal createMakeWithdraw()
	{
		return new MakeWithdrawal1();
	}

	//return method for charge the penalty if current balance is below minimum amount in ATM 1
	public ChargePenalty createChargePenalty()
	{
		return new ChargePenalty1();
	}

	//return method for add or deposit amount in current balance from ATM 1
	public MakeDeposit createMakeDeposit()
	{
		return new MakeDeposit1();
	}

	//return method, for store pin and opening balance for ATM 1
	public StoreData createStoreData()
	{
		return new StoreData1();
	}
} package Kapeel586.AbstractFactory;

import Kapeel586.Output.Chargepenalty.ChargePenalty;
import Kapeel586.Output.Chargepenalty.ChargePenalty2;
import Kapeel586.Output.Displaybalance.DisplayBalance;
import Kapeel586.Output.Displaybalance.DisplayBalance2;
import Kapeel586.Output.Displaymenu.DisplayMenu;
import Kapeel586.Output.Displaymenu.DisplayMenu2;
import Kapeel586.Output.Incorrectpinmessage.Incorrectpinmessage;
import Kapeel586.Output.Incorrectpinmessage.Incorrectpinmessage1;
import Kapeel586.Output.Makedeposit.MakeDeposit;
import Kapeel586.Output.Makedeposit.MakeDeposit2;
import Kapeel586.Output.Makewithdrawal.MakeWithdrawal;
import Kapeel586.Output.Makewithdrawal.MakeWithdrawal2;
import Kapeel586.Output.Promptforpin.Promptforpin;
import Kapeel586.Output.Promptforpin.Promptforpin1;
import Kapeel586.Output.Toomanyattemptsmessage.Toomanyattemptsmessage;
import Kapeel586.Output.Toomanyattemptsmessage.Toomanyattemptsmessage1;
import Kapeel586.Output.dispalybalbelowmin.DispalyBalBelowMin;
import Kapeel586.Output.dispalybalbelowmin.DispalyBalBelowMin1;
import Kapeel586.Output.storedata.StoreData;
import Kapeel586.Output.storedata.StoreData2;

public class ATM_Factory2  implements ATM_Factory{

	@Override
	public DisplayMenu createDisplayMenu() {
		return new DisplayMenu2();
	}

	@Override
	public Incorrectpinmessage createDisplayIncorrectPin() {
		return new Incorrectpinmessage1();
	}

	@Override
	public Toomanyattemptsmessage createDisplayTooManyAttempts() {
		return new Toomanyattemptsmessage1();
	}

	@Override
	public DispalyBalBelowMin createDispalyBalBelowMin() {
		return new DispalyBalBelowMin1();
	}

	@Override
	public DisplayBalance createDispalyBalance() {
		return new DisplayBalance2();
	}

	@Override
	public Promptforpin createDisplayEnterPin() {
		return new Promptforpin1();
	}

	@Override
	public MakeWithdrawal createMakeWithdraw() {
		return new MakeWithdrawal2();
	}

	@Override
	public ChargePenalty createChargePenalty() {
		return new ChargePenalty2();
	}

	@Override
	public MakeDeposit createMakeDeposit() {
		return new MakeDeposit2();
	}

	@Override
	public StoreData createStoreData() {
		// TODO Auto-generated method stub
		return new StoreData2();
	}

}
package Kapeel586.AbstractFactory;

import Kapeel586.Output.Chargepenalty.ChargePenalty;
import Kapeel586.Output.Chargepenalty.ChargePenalty3;
import Kapeel586.Output.Displaybalance.DisplayBalance;
import Kapeel586.Output.Displaybalance.DisplayBalance3;
import Kapeel586.Output.Displaymenu.DisplayMenu;
import Kapeel586.Output.Displaymenu.DisplayMenu1;
import Kapeel586.Output.Incorrectpinmessage.Incorrectpinmessage;
import Kapeel586.Output.Incorrectpinmessage.Incorrectpinmessage1;
import Kapeel586.Output.Makedeposit.MakeDeposit;
import Kapeel586.Output.Makedeposit.MakeDeposit3;
import Kapeel586.Output.Makewithdrawal.MakeWithdrawal;
import Kapeel586.Output.Makewithdrawal.MakeWithdrawal3;
import Kapeel586.Output.Promptforpin.Promptforpin;
import Kapeel586.Output.Promptforpin.Promptforpin1;
import Kapeel586.Output.Toomanyattemptsmessage.Toomanyattemptsmessage;
import Kapeel586.Output.Toomanyattemptsmessage.Toomanyattemptsmessage1;
import Kapeel586.Output.dispalybalbelowmin.DispalyBalBelowMin;
import Kapeel586.Output.dispalybalbelowmin.DispalyBalBelowMin1;
import Kapeel586.Output.storedata.StoreData;
import Kapeel586.Output.storedata.StoreData3;

public class ATM_Factory3 implements ATM_Factory{

	@Override
	public DisplayMenu createDisplayMenu() {
		return new DisplayMenu1();
	}

	@Override
	public Incorrectpinmessage createDisplayIncorrectPin() {
		return new Incorrectpinmessage1();
	}

	@Override
	public Toomanyattemptsmessage createDisplayTooManyAttempts() {
		return new Toomanyattemptsmessage1();
	}

	@Override
	public DispalyBalBelowMin createDispalyBalBelowMin() {
		return new DispalyBalBelowMin1();
	}

	@Override
	public DisplayBalance createDispalyBalance() {
		return new DisplayBalance3();
	}

	@Override
	public Promptforpin createDisplayEnterPin() {
		return new Promptforpin1();
	}

	@Override
	public MakeWithdrawal createMakeWithdraw() {
		return new MakeWithdrawal3();
	}

	@Override
	public ChargePenalty createChargePenalty() {
		return new ChargePenalty3();
	}

	@Override
	public MakeDeposit createMakeDeposit() {
		return new MakeDeposit3();
	}

	@Override
	public StoreData createStoreData() {
		// TODO Auto-generated method stub
		return new StoreData3();
	}

}
package Kapeel586.ATMS;

import Kapeel586.ssaMain;
import Kapeel586.MDA_efsm;
import Kapeel586.Data.Data1;

public class atm1 {
	MDA_efsm m = new MDA_efsm();
	Data1 d1;
	public int tmp_balance;
	public String tmp_pin;
	public int tmp_deposit;
	public static final int 	MIN_BALANCE  = 1000;
	private static final int MAX_ATTEMPTS = 2;//from 0;
	public static final int 	PENALTY 		= 10;

	public void card(int x, String y)
	{
		ssaMain.tmp_pin1 = y;
		ssaMain.tmp_balance1 = x;

		tmp_balance = x;
		tmp_pin = y;
		m.card();
	}

	public void pin(String x)
	{
		d1 = ssaMain.d1;
		if (x.equals(d1.pin))
		{
			m.correctPin();
			if (d1.balance > MIN_BALANCE)
				m.aboveMinBalance();
			else
				m.belowMinBalance();
		}
		else
			m.incorrectPin(MAX_ATTEMPTS);
	}

	public void exit()
	{
		m.exit();
	}

	public void deposit(int d)
	{
		d1 = ssaMain.d1;
		ssaMain.tmp_deposit1 = d;

		tmp_deposit = d;
		m.deposit();
		if (d1.balance > MIN_BALANCE)
		{
			m.aboveMinBalance();
		}
		else
		{
			m.belowMinBalanceWithPanelty();
		}
	}

	public void withdraw(int w)
	{
		d1 = ssaMain.d1;
		ssaMain.tmp_withdraw1 = w;
		m.withdraw();
		if (d1.balance > MIN_BALANCE)
		{
```

```
                    m.aboveMinBalance();
                }
                else
                {
                    m.belowMinBalanceWithPanelty();
                }
        }

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

        public void lock(String x)
        {
                d1 = ssaMain.d1;
                if (x.equals( ssaMain.tmp_pin1))
                {
                        m.lock();
                }
                else
                {
                        m.incorrectLock();
                }
        }

        public void unlock(String x)
        {
                d1 = ssaMain.d1;
                if (x.equals( ssaMain.tmp_pin1))
                {
                        m.unLock();
                        if (d1.balance > MIN_BALANCE)
                        {
                                m.aboveMinBalance();
                        }
                        else
                        {
                                m.belowMinBalance();
                        }
                }
                else
                {
                        m.incorrectUnLock();
                }
        }
}
package Kapeel586.ATMS;
```

```
import Kapeel586.ssaMain;
import Kapeel586.MDA_efsm;
import Kapeel586.Data.Data2;

public class atm2 {
        MDA_efsm m = new MDA_efsm();
        Data2 d2;
        public static final int      MIN_BALANCE            = 500;
        private static final int     MAX_ATTEMPTS           = 1;//from 0;
        public static final int      PENALTY                        = 20;

        public void CARD(float x, int y)
        {
                ssaMain.tmp_pin2 = y;
                ssaMain.tmp_balance2 = x;
                m.card();
        }

        public void PIN(int x)
        {
                d2 = ssaMain.d2;
                if (x == d2.pin)
                {
                        m.correctPin();
                        if (d2.balance > MIN_BALANCE)
                                m.aboveMinBalance();
                        else
                                m.belowMinBalance();
                }
                else
                        m.incorrectPin(MAX_ATTEMPTS);
        }

        public void exit()
        {
                m.exit();
        }

        public void DEPOSIT(float d)
        {
                d2 = ssaMain.d2;
                ssaMain.tmp_deposit2 = d;
                m.deposit();
                if (d2.balance > MIN_BALANCE)
                {
                        m.aboveMinBalance();
                }
```

```
		else
		{
			m.belowMinBalanceWithPanelty();
		}
	}

	public void WITHDRAW(float w)
	{
		d2 = ssaMain.d2;
		ssaMain.tmp_withdraw2 = w;
		m.withdraw();
		if (d2.balance > MIN_BALANCE)
		{
			m.aboveMinBalance();
		}
		else
		{
			m.belowMinBalanceWithPanelty();
		}
	}

	public void BALANCE()
	{
		m.balance();
	}
}
package Kapeel586.ATMS;

import Kapeel586.MDA_efsm;
import Kapeel586.ssaMain;
import Kapeel586.Data.Data3;

public class atm3 {
	MDA_efsm m = new MDA_efsm();
	Data3 d3;
	public static final int	MIN_BALANCE		= 100;
	private static final int	MAX_ATTEMPTS		= 1;//from 0;
	public static final int	PENALTY				= 0;

	public void card(int x, int y)
	{
		ssaMain.tmp_pin3 = y;
		ssaMain.tmp_balance3 = x;
		m.card();
	}

	public void pin(int x)
	{
		d3 = ssaMain.d3;
		if (x==d3.pin)
		{
			m.correctPin();
			if (d3.balance > MIN_BALANCE)
				m.aboveMinBalance();
			else
				m.belowMinBalance();
		}
		else
			m.incorrectPin(MAX_ATTEMPTS);
	}

	public void exit()
	{
		m.exit();
	}

	public void deposit(int d)
	{
		d3 = ssaMain.d3;
		ssaMain.tmp_deposit3 = d;
		m.deposit();
		if (d3.balance > MIN_BALANCE)
		{
			m.aboveMinBalance();
		}
		else
		{
			m.belowMinBalanceWithPanelty();
		}
	}

	public void withdraw(int w)
	{
		d3 = ssaMain.d3;
		ssaMain.tmp_withdraw3 = w;
		m.withdraw();
		if (d3.balance > MIN_BALANCE)
		{
			m.aboveMinBalance();
		}
		else
		{
			m.belowMinBalanceWithPanelty();
		}
	}
	public void balance()
	{
		m.balance();
	}

	public void lock()
	{
		d3 = ssaMain.d3;

		m.lock();
	}

	public void unlock()
	{
		d3 = ssaMain.d3;

			m.unLock();
			if (d3.balance > MIN_BALANCE)
			{
				m.aboveMinBalance();
			}
			else
			{
				m.belowMinBalance();
			}

	}
}
package Kapeel586.Data;

public abstract class data {

    /*maximum invalid pin attempts*/
    protected int maxInvalidAttempts;

    /*Minimum balance Amount to be maintained in the account*/
    protected int minBalance;

    /*Penalty if account is overdrawn*/
    protected int penalty;

    /*Locked Status*/
    protected boolean locked;

    public data() {
    }

    public boolean isLocked() {
        return locked;
    }

    public void setLocked(boolean locked) {
        this.locked = locked;
    }

    public int getMinBalance() {
        return minBalance;
    }

    public void setMinBalance(int minBalance) {
        this.minBalance = minBalance;
    }

    public int getMaxInvalidAttempts() {
        return maxInvalidAttempts;
    }

    public void setMaxInvalidAttempts(int maxInvalidAttempts) {
        this.maxInvalidAttempts = maxInvalidAttempts;
    }

    public int getPenalty() {
        return penalty;
    }

    public void setPenalty(int penalty) {
        this.penalty = penalty;
    }
}

package Kapeel586.Data;


public class Data1 extends data {

    /* pin for the account*/
    public String pin;


    /*balance in the account */
    public int balance;

    /*Input deposit amount*/
    public int deposit;

    /*Input withdraw amount*/
    public int withdraw;

    /* Input pin for the account*/
    public String pinInput;


    /*Input balance in the account */
    public int balanceInput;

    public Data1() {
        super();
        maxInvalidAttempts = 2;
        minBalance = 1000;
        penalty =10;
    }

}
package Kapeel586.Data;

public class Data2 extends data{

    /* pin for the account*/
    public int pin;

    /*balance in the account */
    public float balance;

    /*Input deposit amount*/
    public float deposit;

    /*Input withdraw amount*/
    public float withdraw;

    public Data2() {
        super();
        maxInvalidAttempts = 2;
        minBalance = 1000;
        penalty =10;
    }

}


package Kapeel586.Data;

public class Data3 extends data{  /* pin for the account*/
            public int pin;

            /*balance in the account */
            public int balance;

            /*Input deposit amount*/
            public int deposit;

            /*Input withdraw amount*/
            public int withdraw;

    public Data3() {
        super();
        maxInvalidAttempts = 0;
        minBalance = 100;
        penalty =0;
    }}


package Kapeel586.Data;

public class Snippet {
      public static void main(String[] args) {
        }
}

package Kapeel586.Output.Chargepenalty;

public interface ChargePenalty {

        public void penalty();

}

package Kapeel586.Output.Chargepenalty;
import Kapeel586.ssaMain;
import Kapeel586.ATMS.atm1;

public class ChargePenalty1 implements ChargePenalty{

        //subtracts the penalty amount from current balance for ATM 1
        @Override
        public void penalty()
        {
                ssaMain.d1.balance =ssaMain.d1.balance- atm1.PENALTY;
                System.out.println("BALANCE BELOW MINIMUM. $"+atm1.PENALTY+" PENALTY
APPLIED");

        }
}
package Kapeel586.Output.Chargepenalty;
import Kapeel586.ssaMain;
import Kapeel586.ATMS.atm2;

public class ChargePenalty2  implements ChargePenalty{

        //subtracts the penalty amount from current balance for ATM 2
                @Override
                public void penalty()
                {
                        if(ssaMain.flag==2){
                        ssaMain.d2.balance = ssaMain.d2.balance-atm2.PENALTY;
                        System.out.println("BALANCE BELOW MINIMUM. $"+atm2.PENALTY+" PENALTY
APPLIED");
                        }
                }
        }
```

```
package Kapeel586.Output.Chargepenalty;
import Kapeel586.ssaMain;
import Kapeel586.ATMS.atm3;

public class ChargePenalty3 implements ChargePenalty
{
	//subtracts the penalty amount from current balance for ATM 3
	@Override
	public void penalty()
	{
		{
		ssaMain.d3.balance = ssaMain.d3.balance-atm3.PENALTY;
		System.out.println("BALANCE BELOW MINIMUM. $"+atm3.PENALTY+" PENALTY 
APPLIED");
		}
	}
}

package Kapeel586.Output.dispalybalbelowmin;

public interface DispalyBalBelowMin
{
	public void showBalBelowMin();
}
package Kapeel586.Output.dispalybalbelowmin;

public class DispalyBalBelowMin1 implements DispalyBalBelowMin
{

	//Error message for withdraw amount if current balance is below minimum 
balance
	@Override
	public void showBalBelowMin()
	{
		System.out.println("Sorry the amount could not be withdrawn due 
to insufficient funds or below minimum balance");
	}

}


package Kapeel586.Output.Displaybalance;

public interface DisplayBalance {

	public void showBalance();

}

package Kapeel586.Output.Displaybalance;
import Kapeel586.ssaMain;

public class DisplayBalance1 implements DisplayBalance
```

```
{
	//Show the current balance for ATM 1
		@Override
		public void showBalance()
		{
			System.out.println("Your Current Balance: 
$"+ssaMain.d1.balance);
		}
}
package Kapeel586.Output.Displaybalance;
import Kapeel586.ssaMain;

public class DisplayBalance2 implements DisplayBalance
{
	//Show the current balance for ATM 2
	@Override
	public void showBalance()
	{
		System.out.println("Your Current Balance: $"+ssaMain.d2.balance);

	}
}


package Kapeel586.Output.Displaybalance;

import Kapeel586.ssaMain;

public class DisplayBalance3 implements DisplayBalance{
	//Show the current balance for ATM 3
	@Override
	public void showBalance()
	{
		System.out.println("Your Current Balance: $"+ssaMain.d3.balance);
	}

}


package Kapeel586.Output.Displaymenu;

public interface DisplayMenu {

	public void showMenu();

}

package Kapeel586.Output.Displaymenu;

public  class DisplayMenu1 implements DisplayMenu {

	//show the transaction menu for ATM 1 and 2
		@Override
		public void showMenu()
		{
```

```
                System.out.println();
                  System.out.println("   TRANSACTION MENU: ");
              System.out.println("      deposit");
              System.out.println("      withdraw");
              System.out.println("      balance");
              System.out.println("      lock");
              System.out.println("      unlock");
              System.out.println();


           }
      }




package Kapeel586.Output.Displaymenu;

public class DisplayMenu2 implements DisplayMenu{

      //show the transaction menu for ATM 2
      @Override
      public void showMenu() {

          System.out.println();
            System.out.println("   TRANSACTION MENU: ");
        System.out.println("      DEPOSIT");
        System.out.println("      WITHDRAW");
        System.out.println("      BALANCE");
        System.out.println();


      }

}

package Kapeel586.Output.Incorrectpinmessage;

public interface Incorrectpinmessage {

      public void ShowIncorrectPin();

}

package Kapeel586.Output.Incorrectpinmessage;

public class Incorrectpinmessage1 implements Incorrectpinmessage
{
      //show message if incorrect message entered.
      @Override
      public void ShowIncorrectPin()
      {
                  System.out.println("Incorrect pin.");
      }
```

```
}

package Kapeel586.Output.Makedeposit;

public interface MakeDeposit {

public void deposit();

//public void StoreData1();
      // TODO Auto-generated method stub
}

package Kapeel586.Output.Makedeposit;

import Kapeel586.ssaMain;

      public class MakeDeposit1 implements MakeDeposit
      {
            //add or deposit amount in current balance for ATM 1
            @Override
            public void deposit()
                  {
            ssaMain.d1.balance += ssaMain.tmp_deposit1;
            System.out.println("Your deposit has been successfully added to
your balance");
                  }

      }


package Kapeel586.Output.Makedeposit;
import Kapeel586.ssaMain;

public class MakeDeposit2 implements MakeDeposit
{
      //add amount in current balance for ATM 2
      @Override
      public void deposit()
            {
      ssaMain.d2.balance += ssaMain.tmp_deposit2;
      System.out.println("Your deposit has been successfully added to your
balance");
            }

}


package Kapeel586.Output.Makedeposit;
import Kapeel586.ssaMain;

public class MakeDeposit3 implements MakeDeposit
{
      //add amount in current balance for ATM 3
      @Override
      public void deposit()
            {
```

```
      ssaMain.d3.balance += ssaMain.tmp_deposit3;
      System.out.println("Your deposit has been successfully added to your
balance");
            }

}

package Kapeel586.Output.Makewithdrawal;

public interface MakeWithdrawal {

      public void withdraw();

      }

package Kapeel586.Output.Makewithdrawal;
import Kapeel586.ssaMain;

public class MakeWithdrawal1 implements MakeWithdrawal
{

      //withdraw amount from current balance for ATM 1
      @Override
      public void withdraw()
      {
            ssaMain.d1.balance = ssaMain.d1.balance - ssaMain.tmp_withdraw1;
      }

}
package Kapeel586.Output.Makewithdrawal;
import Kapeel586.ssaMain;

public class MakeWithdrawal2 implements MakeWithdrawal
{
      //withdraw amount from current balance for ATM 2
      @Override
      public void withdraw()
      {
            ssaMain.d2.balance = ssaMain.d2.balance - ssaMain.tmp_withdraw2;
      }

}

package Kapeel586.Output.Makewithdrawal;
import Kapeel586.ssaMain;

public class MakeWithdrawal3 implements MakeWithdrawal
{
      //withdraw amount from current balance for ATM 3
      @Override
      public void withdraw()
      {
            ssaMain.d3.balance = ssaMain.d3.balance - ssaMain.tmp_withdraw3;
      }

}
```

```
package Kapeel586.Output.Promptforpin;

public interface Promptforpin
{

      public void showEnterPin();

}

package Kapeel586.Output.Promptforpin;

public class Promptforpin1  implements Promptforpin
{
      //show message for operation enter pin number
      @Override
      public void showEnterPin()
      {
            System.out.println("Select operation 2 to enter your pin");
      }

}

package Kapeel586.Output.storedata;


public interface StoreData {

      public void saveData();

}

package Kapeel586.Output.storedata;

import Kapeel586.ssaMain;

public class StoreData1 implements StoreData
{

      //store pin and opening balance for ATM 1
      @Override
      public void saveData()
      {
         ssaMain.d1.pin = ssaMain.tmp_pin1;
         ssaMain.d1.balance = ssaMain.tmp_balance1;
         System.out.println("Your account has been established
successfully.");
      }

}



package Kapeel586.Output.storedata;
```

```
import Kapeel586.ssaMain;

public class StoreData2 implements StoreData
{
	//store pin and opening balance for ATM 2
	@Override
	public void saveData()
	{
	   ssaMain.d2.pin = ssaMain.tmp_pin2;
	   ssaMain.d2.balance = ssaMain.tmp_balance2;
	   System.out.println("Your account has been established
successfully.");
	}

}

package Kapeel586.Output.storedata;

import Kapeel586.ssaMain;

public class StoreData3 implements StoreData
{
	//store pin and opening balance for ATM 3
	@Override
	public void saveData()
	{
	   ssaMain.d3.pin = ssaMain.tmp_pin3;
	   ssaMain.d3.balance = ssaMain.tmp_balance3;
	   System.out.println();
	   System.out.println("Your account has been established
successfully.");
	   System.out.println();
	}

}

package Kapeel586.Output.Toomanyattemptsmessage;

public interface Toomanyattemptsmessage {

	public void showTooManyAttempts();

}

package Kapeel586.Output.Toomanyattemptsmessage;

public class Toomanyattemptsmessage1 implements Toomanyattemptsmessage {


	//message shows if certain no of tries attempt for correct pin.
	public void showTooManyAttempts()
	{
		System.out.println("Too many attempts,Card ejected");
	}
}
```

```
package Kapeel586.State;

import Kapeel586.MDA_efsm;


public class balanceChecker extends state{

	public balanceChecker(MDA_efsm m)
	{
		mda = m;
	}

	public void S1(MDA_efsm m)
	{
		mda = m;
	}

	@Override
	public void card()
	{
		System.out.println("Operation open is not available. You already 
opened an account.");
		System.out.println();
	}

	@Override
	public void deposit()
	{
		System.out.println("Operation deposit is not available.");
		System.out.println();

	}

	@Override
	public void withdraw()
	{
		System.out.println("Operation withdraw is not available.");
		System.out.println();

	}

	@Override
	public void balance()
	{
		System.out.println("Operation is not available.");
		System.out.println();
	}

	@Override
	public void incorrectPin(int max)
	{
		System.out.println("Operation is not available.");
		System.out.println();
	}

	@Override
	public void correctPin()
```

```
{
      System.out.println("Operation is not available.");
      System.out.println();
}

@Override
public void aboveMinBalance()
{
      mda.changeState(3);
}

@Override
public void belowMinBalance()
{
      mda.changeState(4);
}


@Override
public void incorrectLock()
{
      System.out.println("Operation is not available.");
      System.out.println();
}

@Override
public void incorrectUnlock()
{
      System.out.println("Operation is not available.");
      System.out.println();
}

@Override
public void lock()
{
      System.out.println("Operation lock is not available.");
      System.out.println();
}

@Override
public void unlock()
{
      System.out.println("Operation unlock is not available.");
      System.out.println();
}

@Override
public void exit()
{
      System.out.println("Operation logout is not available.");
      System.out.println();
}
public void belowMinBalanceWithPanelty()
{
      mda.changeState(4);
      mda.op.chargePenalty();
}
```

```
}

package Kapeel586.State;

import Kapeel586.MDA_efsm;

public class checkpin extends state {
	public  checkpin(MDA_efsm m)
	{
		mda = m;
	}

	@Override
	public void card()
	{
		System.out.println("Operation open is not available. You already 
opened an card.");
		System.out.println();
	}


	@Override
	public void incorrectPin(int max)
	{
		if (mda.attempts < max)
		{
			mda.attempts++;
			mda.op.displayIncorrectPin();
		}
		else
		{
			mda.changeState(0);
			mda.op.displayTooManyAttempts();
			mda.attempts = 0;
		}
	}

	@Override
	public void correctPin()
	{
		mda.changeState(2);
		mda.op.displayMenu();
	}

	@Override
	public void incorrectLock()
	{
		System.out.println("PIN is not entered. Operation deposit is not 
available");
		System.out.println();
	}

	@Override
	public void incorrectUnlock()
	{
```

```
            System.out.println("PIN is not entered. Operation deposit is not 
available");
            System.out.println();
      }

      @Override
      public void deposit()
      {
            System.out.println("PIN is not entered. Operation deposit is not 
available");
            System.out.println();
      }

      @Override
      public void withdraw()
      {
            System.out.println("PIN is not entered. Operation withdraw is not 
available");
            System.out.println();
      }

      @Override
      public void balance()
      {
            System.out.println("PIN is not entered. Operation balance is not 
available");
            System.out.println();
      }

      @Override
      public void lock()
      {
            System.out.println("PIN is not entered. Operation lock is not 
available");
            System.out.println();
      }

      @Override
      public void unlock()
      {
            System.out.println("PIN is not entered. Operation unlock is not 
available");
            System.out.println();
      }

      @Override
      public void exit()
      {
            //mda.changeState(0);
            System.out.println("No Exit operation available");
            System.out.println();
      }

      public void belowMinBalanceWithPanelty()
      {
            System.out.println("PIN is not entered. Operation unlock is not 
available");
```

```
            System.out.println();
      }
}

package Kapeel586.State;
import Kapeel586.MDA_efsm;


public class idle extends state{

      public  idle(MDA_efsm m)
      {
            mda = m;
      }

      @Override
      public void card()
      {
            mda.changeState(1); //Change the state to CheckPin.
            mda.op.storeData(); //Point to data class
      }

      @Override
      public void incorrectPin(int max)
      {
            //mda.op.displayIncorrectPin();
            System.out.println("Please open the card first.");
            System.out.println();
      }

      @Override
      public void incorrectLock()
      {
            System.out.println("Please login first");
            System.out.println();
      }

      @Override
      public void incorrectUnlock()
      {
            mda.op.displayIncorrectPin();
      }

      @Override
      public void correctPin()
      {
            System.out.println("Please open the card first.");
            System.out.println();
      }

      @Override
      public void deposit()
      {
            System.out.println("Please open the card first.");
            System.out.println();
      }
```

```
	@Override
	public void withdraw()
	{
		System.out.println("Please open the card first.");
		System.out.println();
	}

	@Override
	public void balance()
	{
		System.out.println("Please open the card first.");
		System.out.println();
	}

	@Override
	public void lock()
	{
		System.out.println("Please open the card first.");
		System.out.println();
	}

	@Override
	public void unlock()
	{
		System.out.println("Please open the card first.");
		System.out.println();
	}

	@Override
	public void exit()
	{
		System.out.println("Please open the card first.");
		System.out.println();
	}
}

package Kapeel586.State;

import Kapeel586.MDA_efsm;


public class locked extends state{
	public  locked(MDA_efsm m)
	{
		mda = m;
	}

	@Override
	public void card()
	{
		System.out.println("Operation open is not available. You already 
opened an account.");
		System.out.println();
	}

	@Override
	public void deposit()
	{
		System.out.println("You account is locked, please unlock it 
first.");
		System.out.println();

	}

	@Override
	public void withdraw()
	{
		System.out.println("You account is locked, please unlock it 
first.");
		System.out.println();

	}

	@Override
	public void balance()
	{
		System.out.println("You account is locked, please unlock it 
first.");
		System.out.println();
	}

	@Override
	public void lock()
	{
		System.out.println("Operation lock is not available because your 
account had already been locked.");
		System.out.println();
	}

	@Override
	public void incorrectLock()
	{
		mda.op.displayIncorrectPin();
	}

	@Override
	public void unlock()
	{
		mda.changeState(2);
		System.out.println("You unlocked your account.");
	}

	@Override
	public void incorrectUnlock()
	{
		mda.op.displayIncorrectPin();
	}

	@Override
	public void exit()
	{
		System.out.println("Your account is locked, please unlock it 
first.");
		mda.changeState(1);
```

```
            System.out.println();
      }
}


package Kapeel586.State;

import Kapeel586.MDA_efsm;


public class overdrawn extends state{
      public  overdrawn(MDA_efsm m)
      {
            mda = m;
      }

      @Override
      public void card()
      {
            System.out.println("Operation open is not available. You already 
opened an account.");
            System.out.println();
      }

      @Override
      public void deposit()
      {
            mda.changeState(2);
            mda.op.makeDeposit();
      }

      @Override
      public void withdraw()
      {
            mda.op.dispalyBalBelowMin();
      }

      @Override
      public void balance()
      {
            mda.op.displayBalance();
      }

      @Override
      public void lock()
      {
            mda.changeState(5);
            System.out.println("You locked your account.");
      }

      @Override
      public void unlock()
      {
            System.out.println("Please lock first to use unlock");
            System.out.println();
      }
```

```
	@Override
	public void incorrectLock()
	{
		mda.op.displayIncorrectPin();
	}

	@Override
	public void exit()
	{
		mda.changeState(0);
		System.out.println("Card ejected");
		System.out.println();

	}

	@Override
	public void incorrectPin(int max)
	{
		mda.op.displayIncorrectPin();
	};

}

package Kapeel586.State;

import Kapeel586.MDA_efsm;


public class ready extends state{
	public  ready(MDA_efsm m)
	{
		mda = m;
	}

	@Override
	public void card()
	{
		System.out.println("Operation open is not available. You already 
opened an account.");
		System.out.println();
	}

	@Override
	public void deposit()
	{
		mda.op.makeDeposit();

	}

	@Override
	public void withdraw()
	{
		mda.op.makeWithdrawal();
		mda.changeState(2);

	}
	@Override
	public void balance()
	{
		mda.op.displayBalance();
	}

	@Override
	public void lock()
	{
		mda.changeState(5);
		System.out.println("You locked your account.");
	}

	@Override
	public void unlock()
	{
		System.out.println("Please lock first to use unlock");
		System.out.println();
	}

	@Override
	public void exit()
	{
		mda.changeState(0);
		System.out.println("Card ejected");
		System.out.println();
	}

	@Override
	public void incorrectLock()
	{
		mda.op.displayIncorrectPin();
	}
}

package Kapeel586.State;

import Kapeel586.MDA_efsm;

public abstract class state {
MDA_efsm	mda;

	public void card()
	{
	};

	public void incorrectPin(int max)
	{
	};

	public void correctPin()
	{
	};

	public void aboveMinBalance()
	{
	};

	public void belowMinBalance()
	{
	};

	public void lock()
	{
	};

	public void deposit()
	{
	};

	public void withdraw()
	{
	};

	public void balance()
	{
	};

	public void exit()
	{
	};

	public void incorrectLock()
	{
	};

	public void unlock()
	{
	};

	public void incorrectUnlock()
	{
	}

	public void belowMinBalanceWithPanelty()
	{

	}

}
```