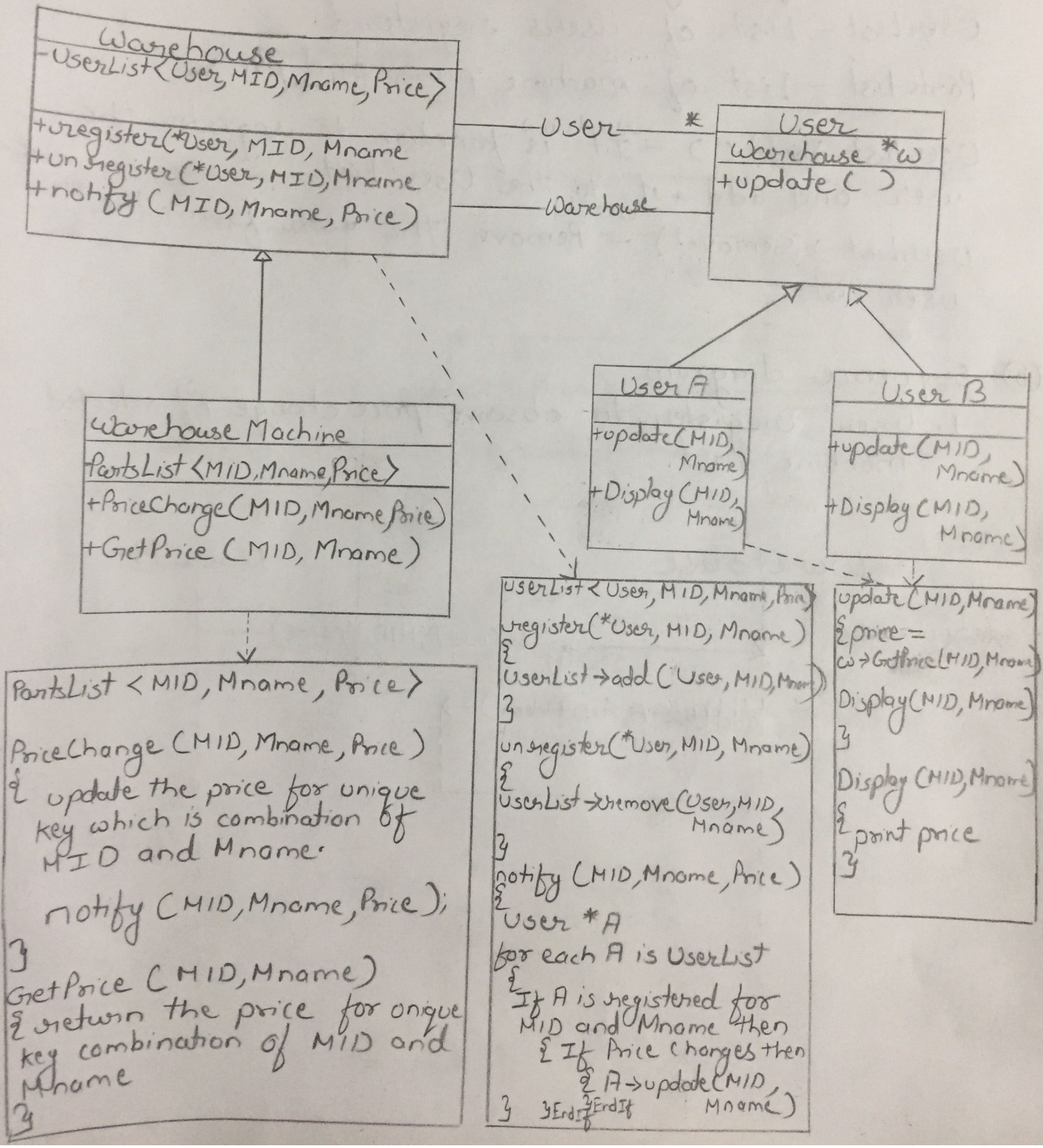
Name – Abhinav Pimpalgaonkar
Student Id- A20387324

Assignment - 1

Problem 1

(a) Observer Pattern for Warehouse - Class diagram



* Key Points

MID – Id number for machine Part

Mname – Machine Part Name

Price – Price for machines parts which is uniquely identified by MID and Mname

UserList – List of users registered

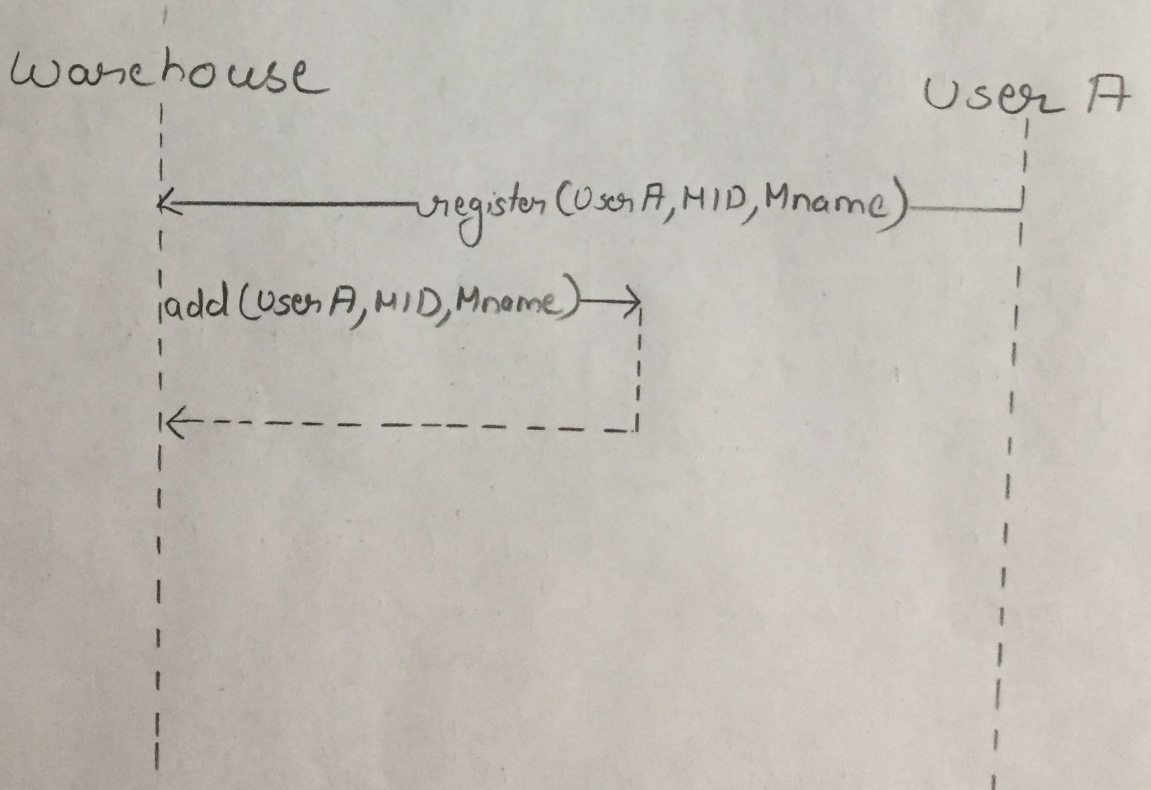
Parts list – List of machine parts and price

Userlist → add ( ) – It is function to register the user and add it to the User List

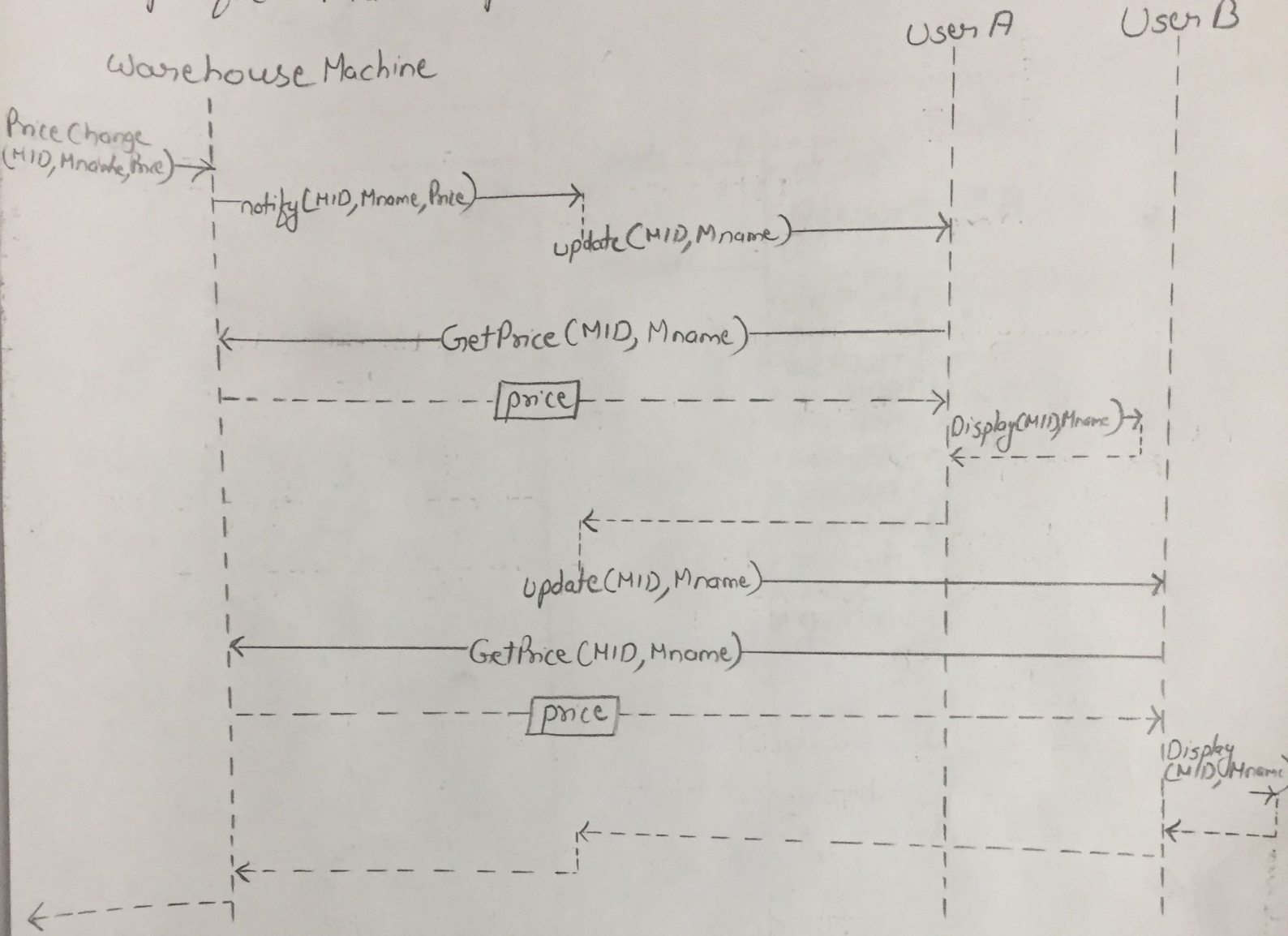
UserList → remove() – Remove the user from the user List

(b) Sequence diagram

1. User register to observe price change of selected machine part.



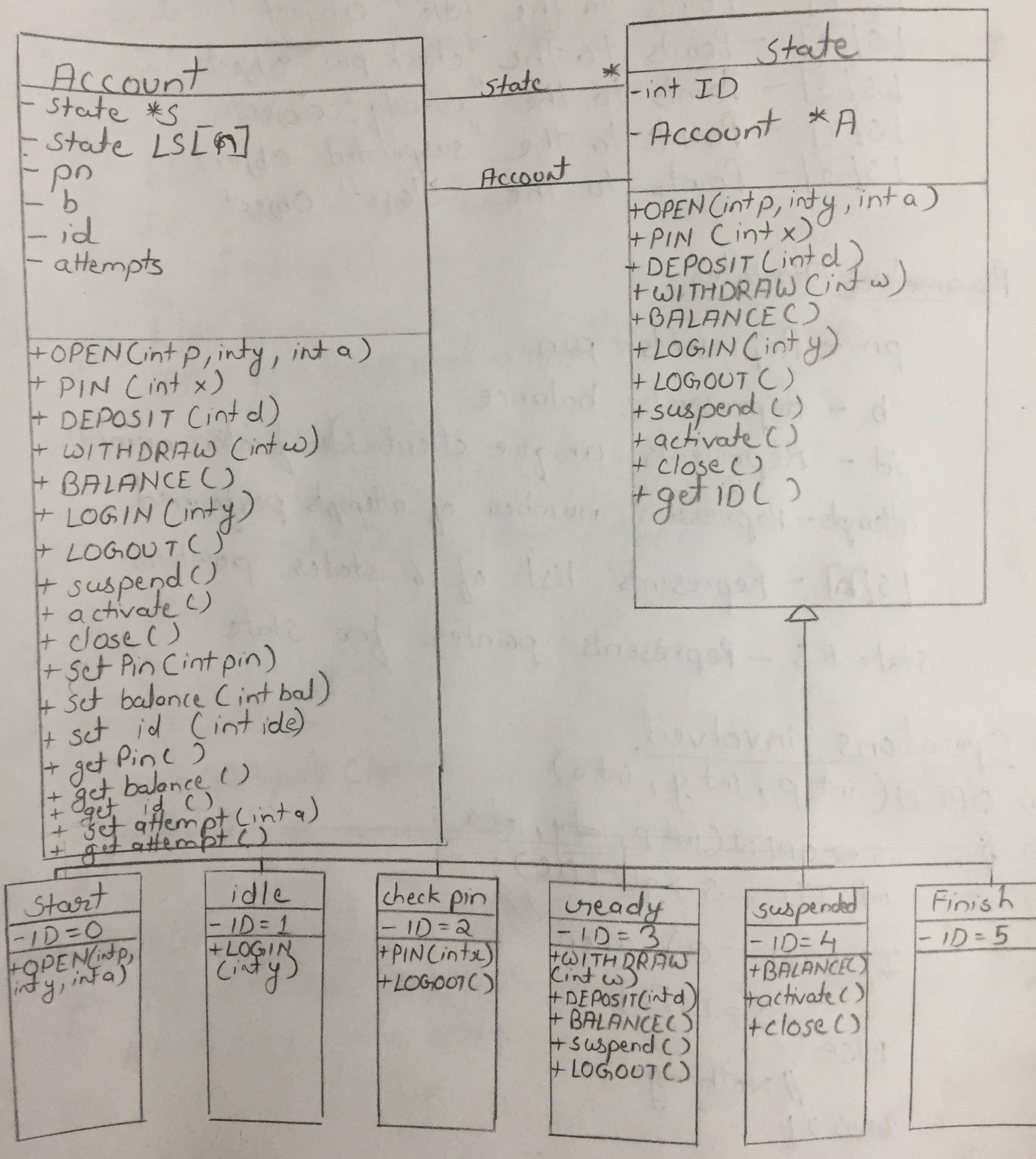
2. System notification to user about price change of specific machine part



Problem 2

(a) Design the system using State Design Pattern.

1. Centralized Version



Account Class

List of states

LS[0] - Points to the "start" object
LS[1] - Points to the "idle" object
LS[2] - Points to the "check pin" object
LS[3] - Points to the "ready" object
LS[4] - Points to the "suspended" object
LS[5] - Points to the "close" object

~~Parameters~~ Attributes.

pn - Represents pin
b - Represents balance
id - Represents unique client identification number
attempts - Represents number of attempts performed
LS[6] - Represents list of 6 states pointers.
state *S - Represents pointer for state

Operations involved.

1) OPEN(int p, int y, int a)

```
{   S → OPEN(int p, int y, int a);
    int ID = S → getID();
    If (ID == 0) then
        S = LS[1]
    Else
        // Nothing
    END If
}
```

2) PIN (int x)

{ S → PIN (~~int~~ x); int temp = pn; int temp2 = attempts;

int ID = S → getID ();

If (ID == 2) Then

IF (x != temp and temp2 < 2) then

//Remain in the same state

Else

IF (x == temp) then

S = LS[3];

Else

IF (x != temp and attempts == 2) then

S = LS[1];

Else

S = LS[1]; //For LOGOUT

End IF

}

3) DEPOSIT (int d)

{ S → DEPOSIT (~~int~~ d);

//Remain in same state

}

4) WITHDRAW (int w)

{ S → WITHDRAW (~~int~~ w);

//Remain in same state

}

⑤ BALANCE ( )

```
{  S → BALANCE ();
   //Remain in same state
}
```

⑥ LOGIN (int y)

```
{  S → LOGIN (y) ; int a = y;
   int ID = S → getID();
   IF (ID == 1) Then
      IF (a != id)
         //Remain in same state
      Else
         S = LS[2];
   END IF
}
```

⑦ LOGOUT ( )

```
{  S → LOGOUT ();
   int ID = S → getID();
   IF ( ID == 2 OR ID == 3 )
      S = LS[1];
   END IF
}
```

(8) suspend ( )

```
{ S → suspend ( );
  int ID = S → getID ( )
  IF (ID == 3) then
      S = LS[4]
  END IF
}
```

(9) Activate ( )

```
{ S → activate ( )
  int ID = S → getID ( )
  IF (ID == 4) then
      S = LS [3]
  END IF
}
```

(10) close ( )

```
{ S → close ( )
  int ID = S → getID ( )
  IF (ID == 4)
      S = LS[5]
  END IF
}
```

(11) setPin(int pin)
{ pn = pin
}

(12) setBalance(int bal)
{ b = bal
}

(13) setid(int ide)
{ id = ide
}

(14) getPin()
{ return pn
}

(15) getbalance()
{ return b
}

(16) getid()
{ return id
}

(17) setattempts(int a)
{ attempts = a
}

(18) getattempt()
{ return attempts
}

Class State

List of states

ID = 0 Start
ID = 1 idle
ID = 2 check pin
ID = 3 ready
ID = 4 suspended
ID = 5 Finish

Attributes

id - indicates the state id

Note - There are two functions.
1) getid() - This in the class Account. Used for unique identity of Person
2) getID() - This is in the class state. Used for change of states.

Operations involved

1) ~~OPEN (int p, int y, int a)~~

1) get ID ()
{ return ID;
}

Class start
ID = 0

Operations

1) OPEN (int p, int y, int a)
{ A → setPin (int p)
A → set balance (a)
A → set id (y)
}

Class idle
ID = 1

Operations

1) Login (int y)
{ int b = A → get id ()
IF (b != y) then
print "incorrect user id"
Else
A → set attempt (0)
}

class check pin

ID = 2

Operations

1) PIN(int x)

```
{ int c = A→getPin()
  int d = A→getattempt()
  IF (c != x and attempt d<2) then
    { Print "Incorrect Pin"
        d = d+1
      A→ set attempt (d)
    }
  Else
   IF ( c == x) and then
       Display Menu

   Else
     int e = A→ get attempt()
    IF ( c != x and e==2) then
       Print "Incorrect Pin; too many attempts"
    End IF
}
```

2) LOGOUT ( )

```
{  int e = A→get attempt()
     e = e+4

}
```

Class ready

ID = 3

Operations

1) WITHDRAW (int w)

```
{ int f = A → getbalance ()
  IF ( f < -5000 ) then
  {Print "Cannot withdraw; Account needs to be suspended
     A → suspend ()
  }
  Else
  { f = f - w
    ~~A → setbalance (f)~~
    A → set balance (f)
  }
```

2) DEPOSIT (int d)

```
{ int g = A → getbalance ()
    g = g + d
    A → set balance (g)
}
```

3) BALANCE ()

```
{ int h = A → get balance ()
  Print h
}
```

4) suspend ( )
{ Print Message "Account suspended; Do you want to activate the account or close the account
Take two input from user - activate and close
1 - Activate
0 - Close
If (1) then
A → activate ( )
Else
A → close ( )
End IF
}

4) suspend ( )
{
}

5) Logout
{ Print Message "Enter 0 for logout"
IF (0) then
~~A → LOGIN~~ // Display Nothing
End IF
}

Class suspended
ID = 4
Operations.
1) BALANCE ( )
{ int i = A → get balance ( )
Display i
}

2) close ( )
{
}

③ activate ()
{

}

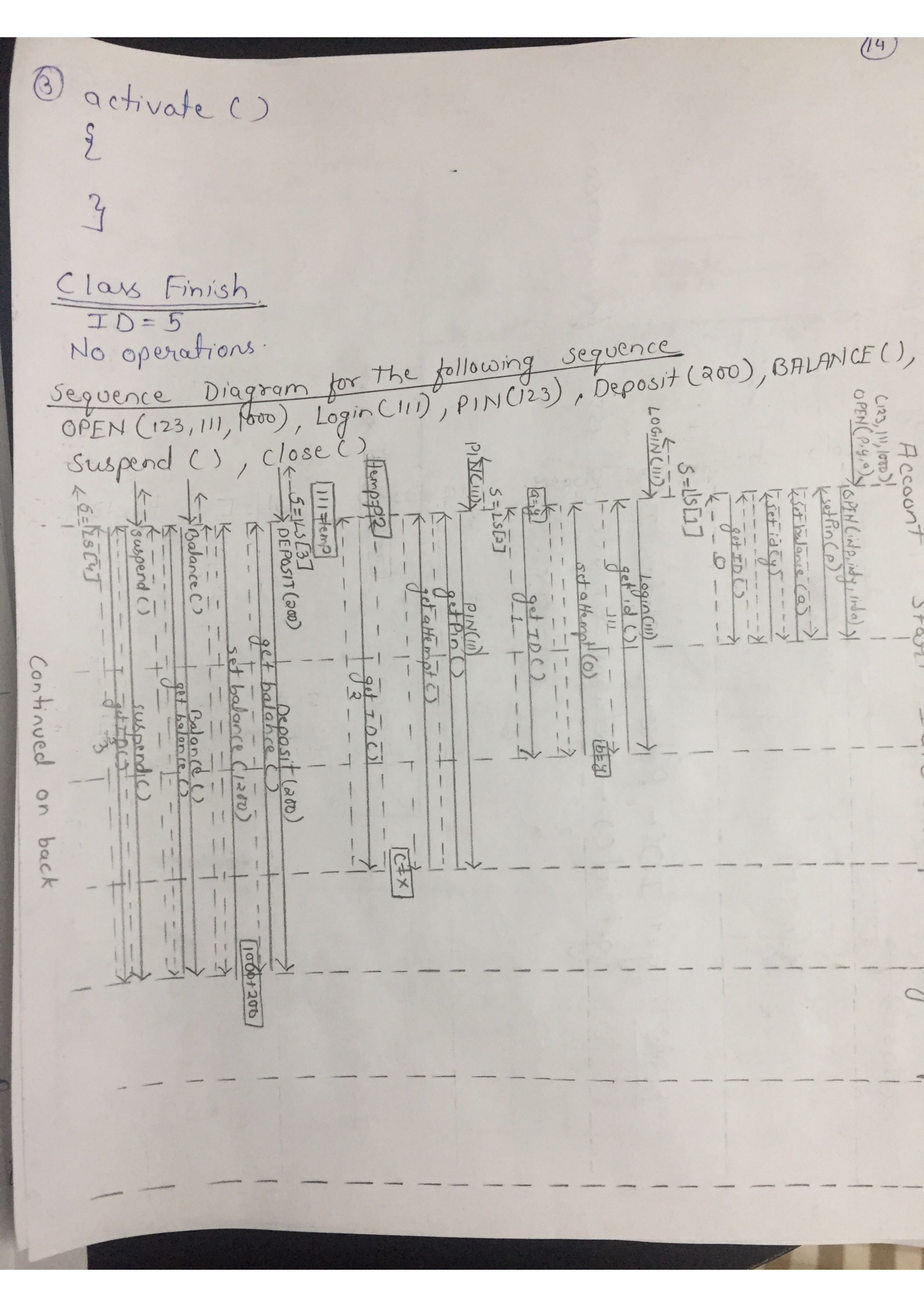
Class Finish

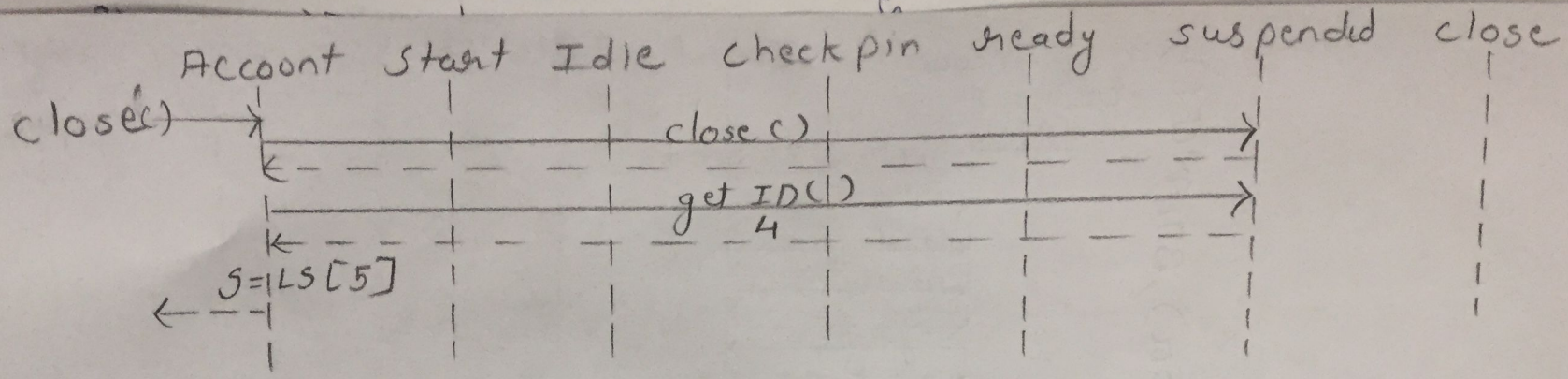
ID = 5

No operations.

Sequence Diagram for the following sequence

OPEN (123, 111, 1000), Login (111), PIN (123), Deposit (200), BALANCE (), Suspend (), close ()

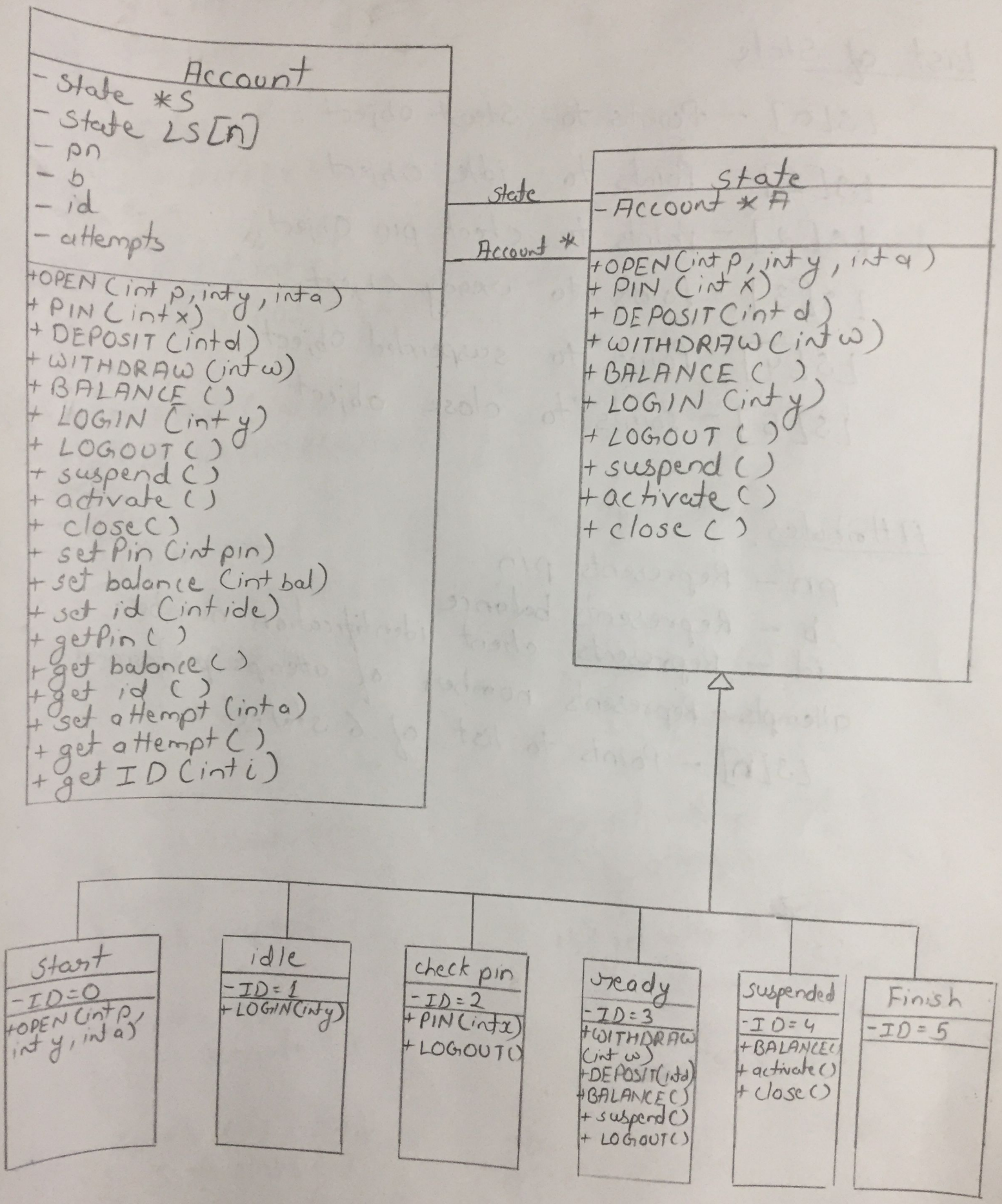




Note — get ID() → Returns states Id

get id() – Returns unique id issued for a person

Decentralised State Pattern



Class Account

List of State

LS[0] - Points to start object
LS[1] - Points to idle object
LS[2] - Points to check pin object
LS[3] - Points to ready object
LS[4] - Points to suspended object
LS[5] - Points to close object

Attributes

pn - Represents pin
b - Represents balance
id - Represents client identification number
attempts - Represents number of attempts performed
LS[n] - Points to list of 6 states.

Class Account Operations

1) OPEN (int p, int y, int a)
{ S → OPEN (P, y, a)
}

2) PIN (int x)
{ S → PIN (x)
}

3) DEPOSIT (int d)
{ S → DEPOSIT (d)
}

4) WITHDRAW (int w)
{ S → WITHDRAW (w)
}

5) BALANCE ( )
{ S → BALANCE ( )

6) LOGIN (int y)
{ S → LOGIN (y)
}

7) LOGOUT ( )
{ S → LOGOUT ( )
}

8) suspend ( )
{ S → suspend ( )
}

9) activate ( )
{ S → activate ( )
}

10) close ( )
{ S → close ( )
}

11) setPin (int pin)
{ pn = pin
}

12) ~~Get~~ getPin ( )
{ return pn
}

13) set balance (int bal)
{ b = bal
}

14) set id (int ide)
{ id = ide
}

15) get balance ( )
{ return b
}

16) get id ( )
{ return id
}

17) set attempt (int a)
{ attempts = a
}

18) get attempt ( )
{ return attempts
}

19) get ID (int i)
{ S = LS[i]
}

Class "State"

All the functions are abstract functions

Class "Start"

ID = 0

Operations.

1) OPEN (int p, int y, int a)

```
{   A->setPint(p)
    A->set id(y)
    A->setbalance(a)
    A->get ID[1]
}
```

Class idle

ID = 1

Operations.

1) LOGIN (int y)

```
{ int a = A->get id()
  IF (y != a) then
    Print "Invalid Id"
  else
    { A->set attempt(0)
      A-> get ID[2)
    }
```

}

Class check pin

ID = 2

Operations

1) PIN (int x)

```
{ int a = A → get Pin()
  int b = A → get attempt
  If ( x != a and b < 2) then
    { Print "Incorrect Pin"
      A → set attempt (b+1)
    }
  else
  If ( x != a and b == 2) then
    { Print "Incorrect pin and too many attempts"
      A → get ID (1)
      A → set attempt (0)
    }
  else
    { Display the menu of option
      A → get ID (3)
    } End IF
}
```

2) LOGOUT ()

```
{  A → get ID (1)
}
```