Software Systems Architecture (CS-586)

Homework #1

- *Sivasenthil Namachivayan*

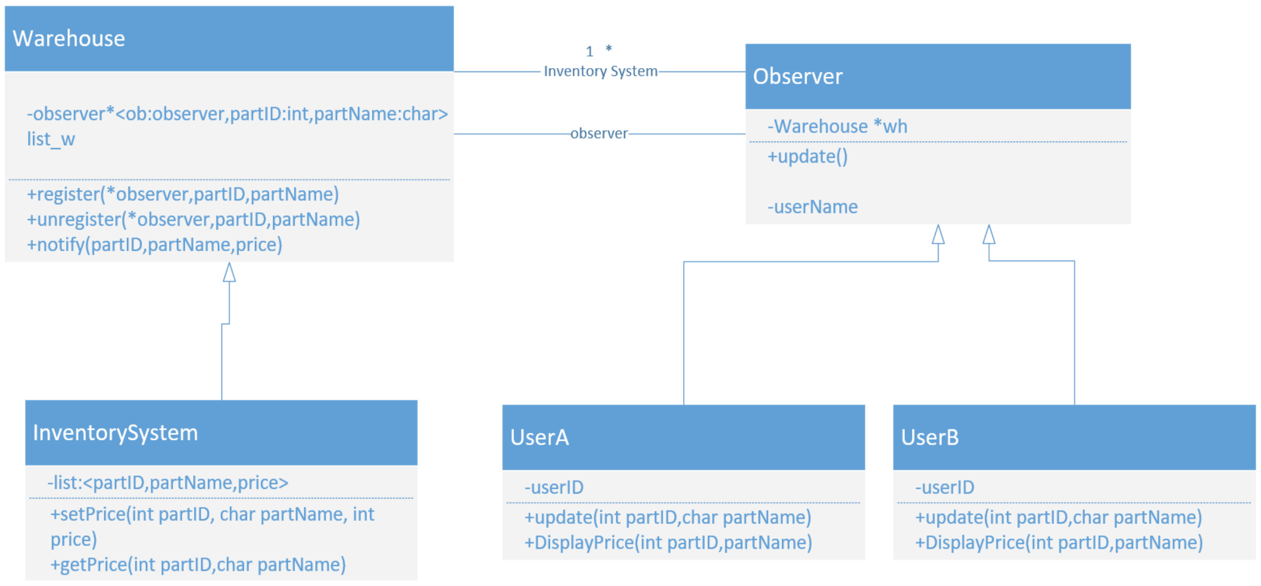
- *A20391478*

Table of Contents

Problem #1

Observer Design pattern

Class Diagram


Warehouse
-observer*<ob:observer,partID:int,partName:char> list_w
+register(*observer,partID,partName)
+unregister(*observer,partID,partName)
+notify(partID,partName,price)
1 *
Inventory System
observer
Observer
-Warehouse *wh
+update()
-userName
InventorySystem
-list:<partID,partName,price>
+setPrice(int partID, char partName, int price)
+getPrice(int partID,char partName)
UserA
-userID
+update(int partID,char partName)
+DisplayPrice(int partID,partName)
UserB
-userID
+update(int partID,char partName)
+DisplayPrice(int partID,partName)

Pseudo-code

Class InventorySystem

//The product is assigned with the respective inventory price and the price is shown

```
setPrice(partID,partName,price)
{
        list<partID,partName>→price=price;
        notify(partID,partName,price);
}
getPrice(partID,partName)
{
        x=list<partID,partName>→price;
        return x;
}
```

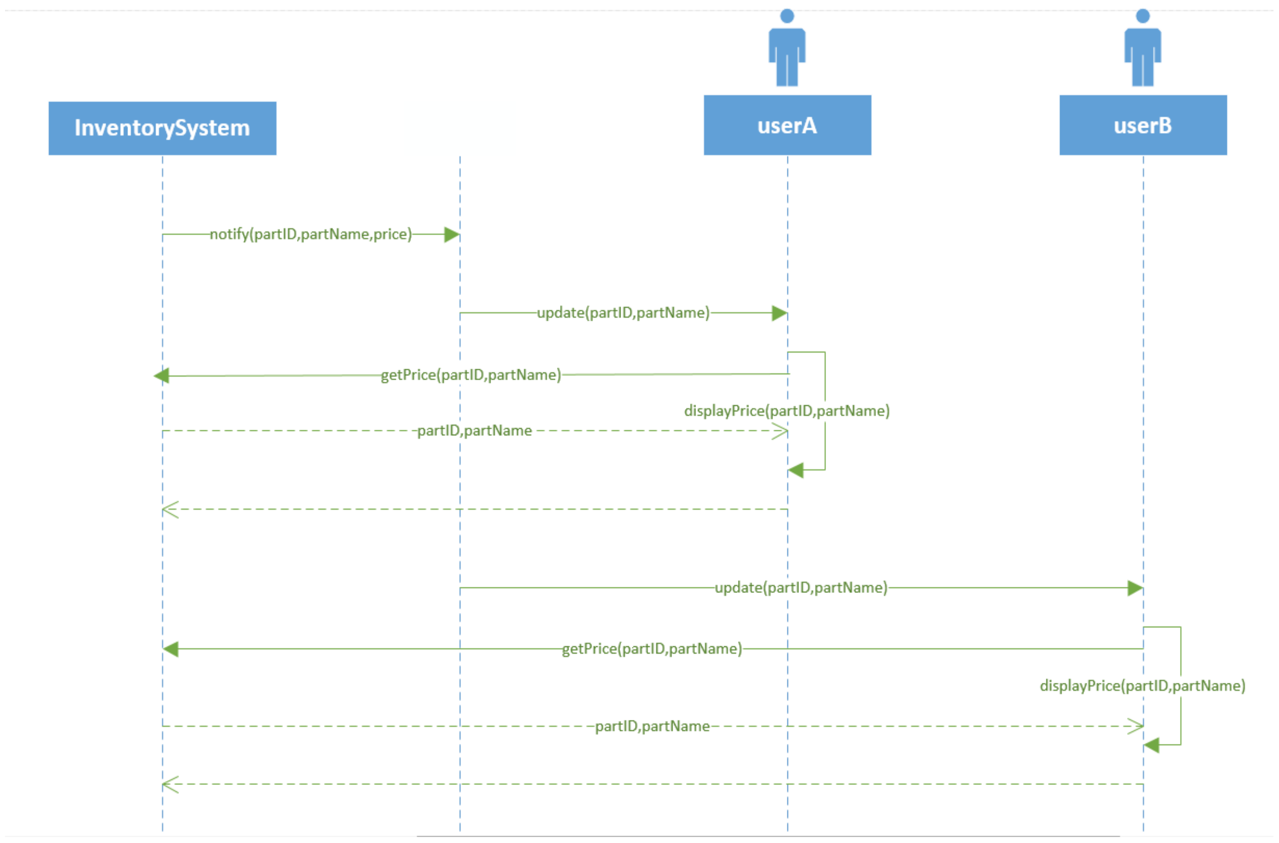
Class UserA & Class UserB
//The observer(user) is provided with the userID and the machine parts which he is interested in knowing the change in price

```
update(int partID, char partName)
{
        displayPrice(partID,partName)
}

//To observer(user) is provided with the inventory system price as base price
displayPrice(int partID,chat partName)
{
        int price=wh→getPrice(partID,partName);
        print price;
}
```

Class Warehouse

```
//The observer(user) is registered with the warehouse inventory system

register(*observer,partID,partName)
{
        list_w→add(ob,partID,partName)
}

//The observer(user) is unregistered from the warehouse inventory system

unregister(*observer,partID,partName)
{
        if(list_w find(ob,partID,partName)!=Null)
        list_w→remove(observer,partID,partName)
}

//The observer(user) is notified about the price change in the machine parts

notify(partID,partName,price)
{
        For each observer ob (user) in list_w
        {
                observer *ob
                if(partID,partName=list_w→partID,partName)
                &&
                price<getPrice || price>getPrice
        }
        ob→update(partID,partName)
}
```

Sequence Diagram

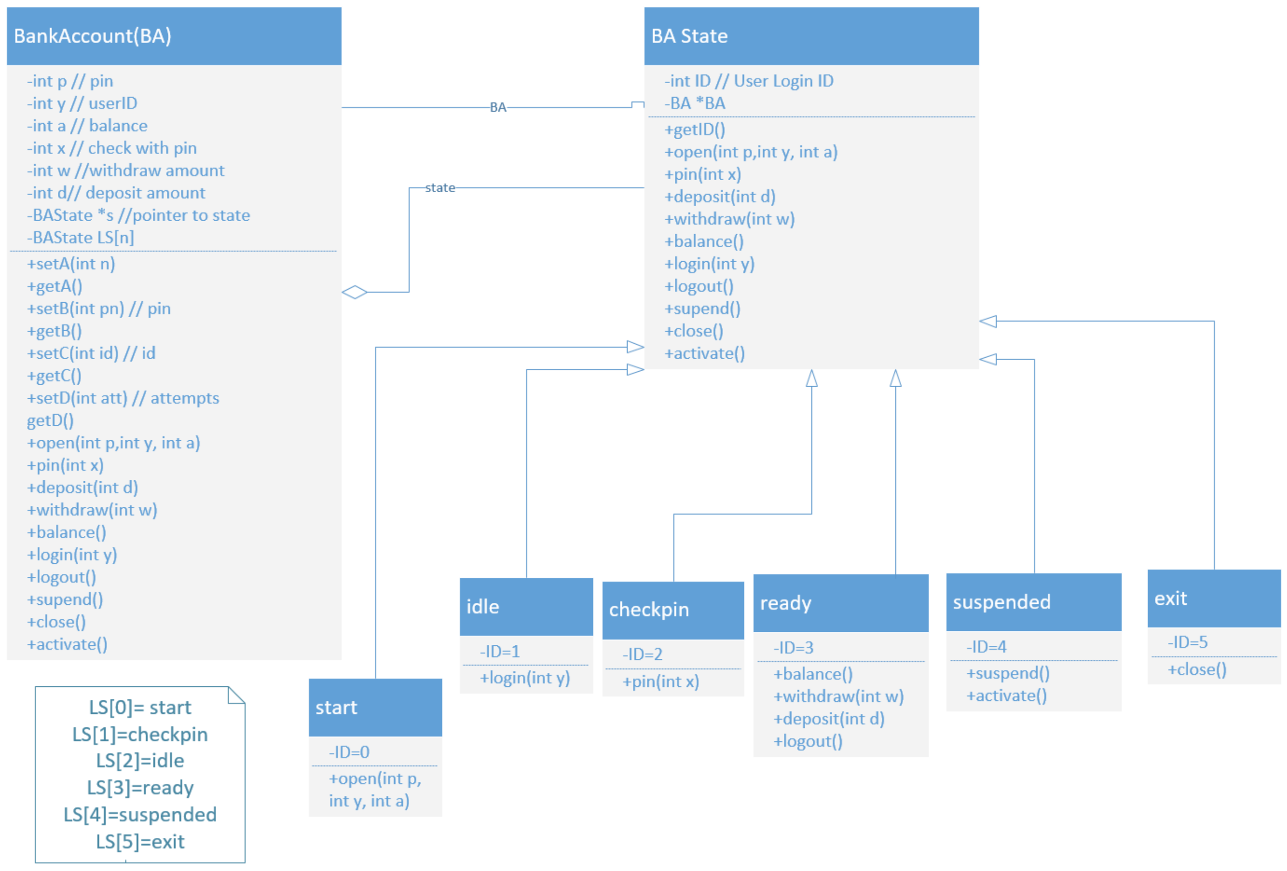

InventorySystem
userA
userB
notify(partID,partName,price)
update(partID,partName)
getPrice(partID,partName)
displayPrice(partID,partName)
partID,partName
update(partID,partName)
getPrice(partID,partName)
displayPrice(partID,partName)
partID,partName

Problem #2

State Design Pattern

Centralized version

Class Diagram



Pseudo-code

Class BackAccount(BA)

```
setA(int n)
{
        state=n;
}
getA()
{
        return n;
}
setB(int pn) //pin verification
{
        pn=p;
}
getB()
{
        return p;
}
setC(int id) //id verification
{
        id=y;
}
getC()
{
        return y;
}
setD(int att) //attempts count verification
{
        att<2;
}
getD()
{
        return att;
}
open(int p, int y, int a)
{
        s→ open(p,y,a);
        if(pn=p&&id=y&&a=b)
        n=s.getID;
        if(x=start)
        {
                s= LS[0];
        }
}
login(int y)
```

```
{
        s→ login(y);
        n=s.getID();
        if(x=start)
        {
                s=LS[1];
        }
}
checkpin(int p)
{
        s→ pin(x);
        n=s.getID();
        if(x=p)
        {
                s=LS[1];
        }
        if(x!=p)&&(att<2)
        {
                getD();
                att=att+1;
                display incorrect pin
        }
}
deposit(int d)
{
        s→deposit(d);
        n=s.getID();
        if(n=checkpin)
        {
                s=LS[3];
        }
}
withdraw(int w)
{
        s→ withdraw(w);
        n=s.getID();
        if(n=ready)
        {
                s=LS[3];
        }
}
balance(b)
{
        s→balance(b);
        n=s.getID();
        if(n=ready)
        {s=LS[3];}
}
logout()
{
        s→ logout();
        n=s.getID();
        if(n=idle)
        {
                s=LS[1];
        }
}
suspend()
{
        s→suspend();
        n=s.getID();
        if(n=ready)
        {
                s=LS[4];
        }
}
activate()
{
        s→ activate();
        n=s.getID();
        if(n=suspended)
        {
                s=LS[3];
        }
}
close()
{
        s→ close();
        n=s.getID();
        if(n=supended)
        {
                s=LS[0];
        }}
```

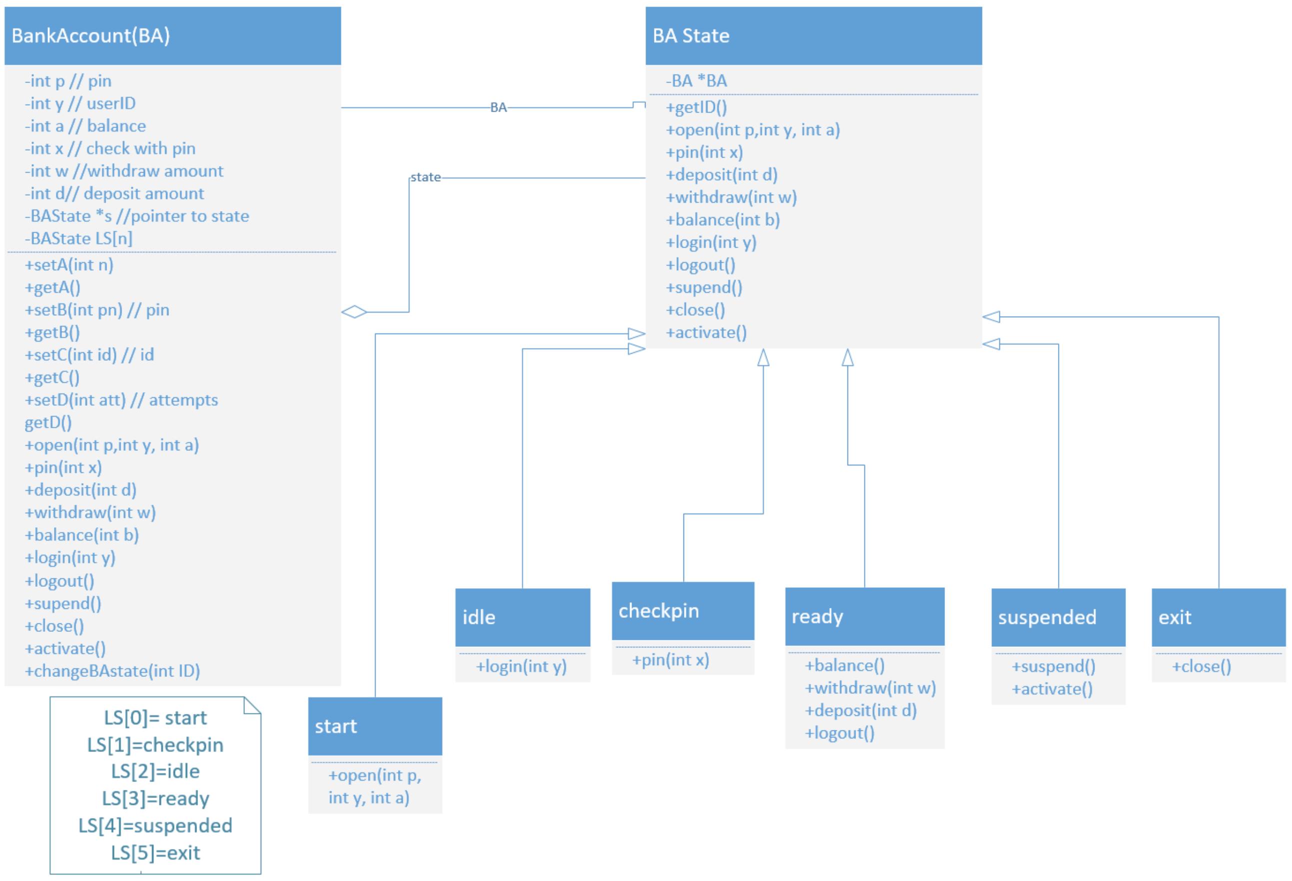
Class start()

```
open(int p, int y, int a)
{
        s→ open(p,y,a)
        If ( pn=p && b=a && id=y )
        {
                BA.getID=1;
                //state is changed to idle class
        }
```

```
class idle()
login(int y)
{
    s→ login(y)
    if(login(y)[y=id])
    {
        getC();
        BA.getID=2;
    }else
    {
        display incorrect ID message
        BA.getID=1;
    }
}
class checkpin()
pin(int x)
{
    If pin(x)[x!=pn)&&(attempts<2)]= true
            {
                attempts++;
                display Login again
                BA.getID=2;
            }
        elseif
            {
                pin(x)[(x!=pn)&&(attempts==2)];
                incorrect pin message; too many wrong attempts message
            }
        else
            {
                BA.getID=2;
                moved to the ready state after login was successful
            }
    }
}
class ready()
{
    s→checkpin
    if(pin(x)[x==pn]
    {display menu}
    if(b<0)
    {
        If the action deposit is performed
        b=b+d && b>0
        BA.getID=3;
        The account is moved from suspended to ready state and its activated
        }
}
withdraw(int w)
{
        if(b>0) && (b=b-w)
        {
                The withdrawal amount is successful
                return b;       // display the balance value in the account
        }
        elseif([b<=0])
        {       display the message
                There is no enough funds to perform the transaction
        }
        else
        {
                suspend the account
                display the balance
                return b;
                BA.getID=3;
        }
}
deposit(int d)
{
        perform the operation
        b= b+d
        return b        // display the balance value in the account
        BA.getID=3;
}
balance()
{
        display the balance value
        BA.getID=3;
}
class suspended()
{
        getID();
        BA.getID=4;
}
class exit()
{       close();
        BA.getID=0;
}
Class BAState()
getID()
{return ID;}
All the other functions in this class are abstract functions
```

Decentralized Version

Class diagram



Pseudo-code

Class BackAccount(BA)

```
setA(int n)
{
        state=n;
}
getA()
{
        return n;
}
setB(int pn) //pin verification
{
        pn=p;
}
getB()
{
        return p;
}
setC(int id) //id verification
{
        id=y;
}
getC()
{
        return y;
}
setD(int att) //attempts count verification
{
        att<2;
}
getD()
{
        return att;
}
open(int p, int y, int a)
{
        s→ open(p,y,a);
}
pin(int x)
{
        s→ pin(x);
}
login(int y)
{
        s→ login(y);
}
deposit(int d)
{
        s→deposit(d);
}
withdraw(int w)
{
        s→ withdraw(w);
}
balance()
{
        s→balance();
}
logout()
{
        s→ logout();
}
suspend()
{
        s→suspend();
}
activate()
{
        s→ activate();
}
close()
{
        s→ close();
}
changeBAstate(int ID)
{
        s= LS[ID]
}
```

Class Start()

```
{
        BA.setA(a);
        BA.changeBAstate(0);
}
open(int p, int y, int a)
{
        s→ open(p,y,a)
        If ( pn=p && b=a && id=y )
        {
                BA.changeBAstate(1);
        }
```

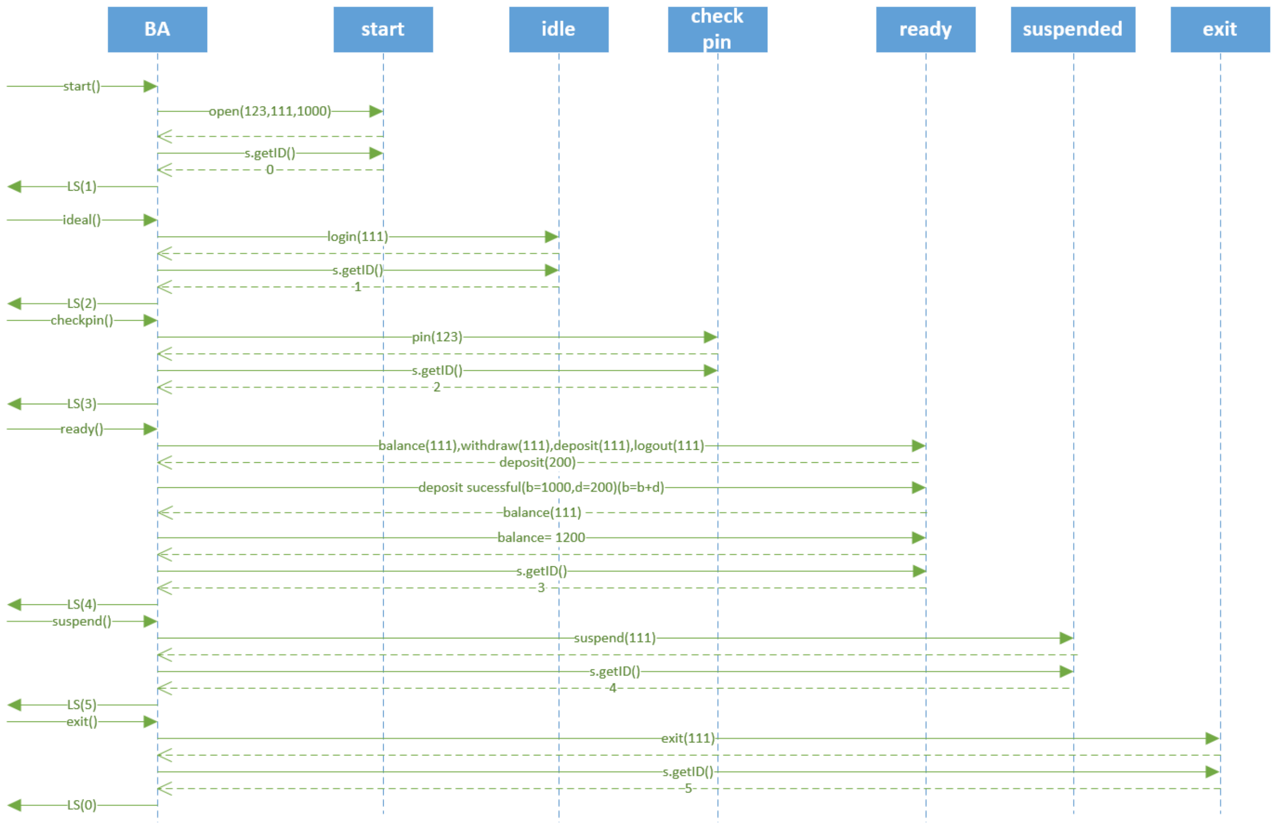
```
class idle()
login (int y)
{
        s→login(y)
        if(login(y)[y=id])

        {
                 BA.changeBAstate(2);
        }else
        {
                display incorrect ID message
                 BA.changeBAstate(1);
        }
}
class checkpin()
pin(int x)
        {
                If pin(x)[x!=pn)&&(attempts<2)]= true
                        {
                                attempts++
                                display Login again
                                 BA.changeBAstate(0);
                        }
                elseif
                        {
                                pin(x)[(x!=pn)&&(attempts==2)]
                                incorrect pin message; too many wrong attempts message
                        }
                else
                        {
                                 BA.changeBAstate(1);
                                moved to the ready state after login was successful
                        }
        }
}
class ready()
{
        s→idle
        if(pin(x)[x==pn]
        {
                display menu
        }
        if(b<0)
        {
                If the action deposit is performed
                b=b+d && b>0
```

```
                 BA.changeBAstate(3);
                The account is moved from suspended to ready state and its activated
        }
}
withdraw(int w)
{
        if(b>0) && (b=b-w)
        {
                The withdrawal amount is successful
                return b            // display the balance value in the account
        }
        elseif([b<=0])
        {
                display the message
                There is no enough funds to perform the transaction
        }
        else
        {
                suspend the account
                display the balance
                return b
                 BA.changeBAstate(3);
        }
}
deposit(int d)
{
        perform the operation
        b= b+d
        return b            // display the balance value in the account
         BA.changeBAstate(3);
}
balance()
{
        display the balance value
         BA.changeBAstate(3);
}
class suspended()
{
        BA.changeBAstate(4);
}
class exit()
{       close();
         BA.changeBAstate(0);}
Class BAState()
All the other functions in this class are abstract functions
```

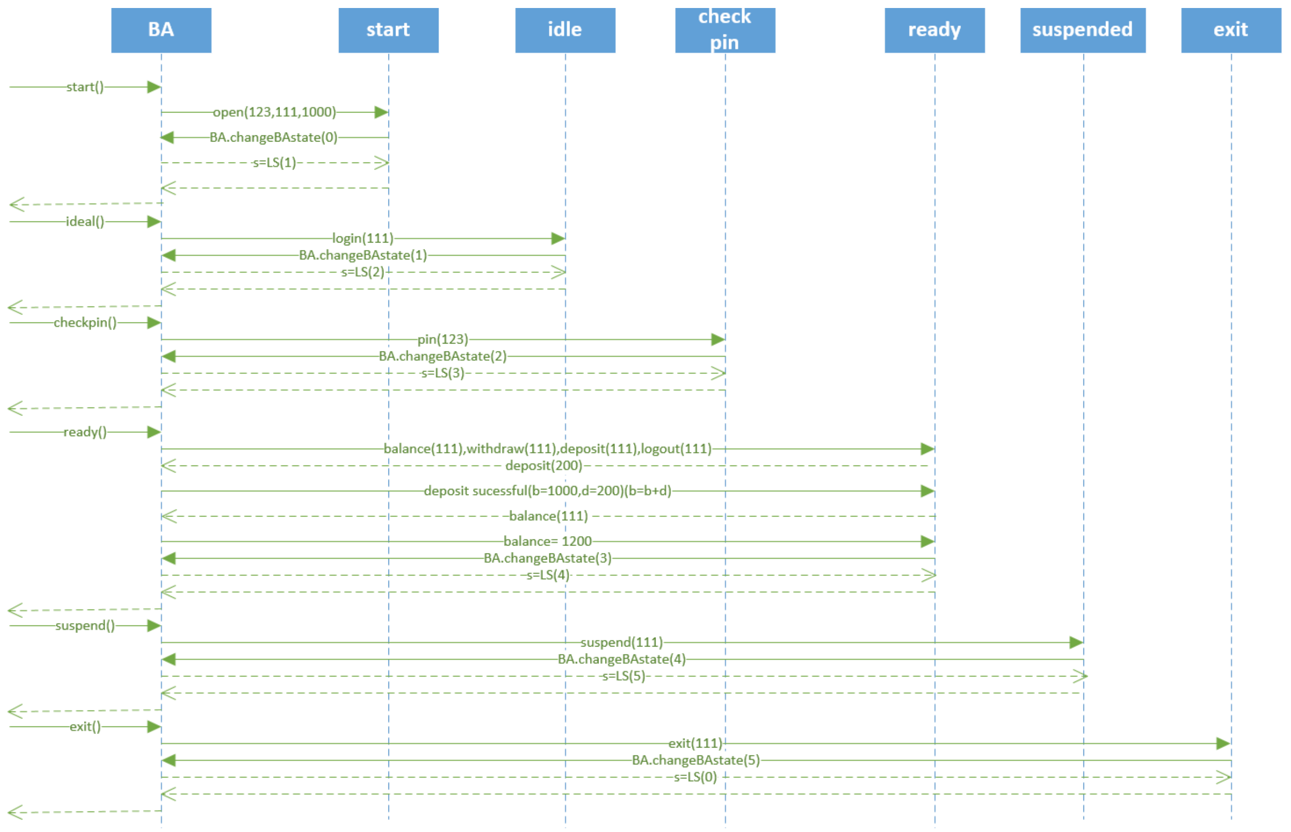
Sequence Diagram for the given case

OPEN(123,111,1000), LOGIN(111), PIN(123), DEPOSIT(200), BALANCE(), suspend(), close()

Centralized Version



Decentralized Version


BA
start
idle
check pin
ready
suspended
exit
start()
open(123,111,1000)
BA.changeBAstate(0)
s=LS(1)
ideal()
login(111)
BA.changeBAstate(1)
s=LS(2)
checkpin()
pin(123)
BA.changeBAstate(2)
s=LS(3)
ready()
balance(111),withdraw(111),deposit(111),logout(111)
deposit(200)
deposit sucessful(b=1000,d=200)(b=b+d)
balance(111)
balance= 1200
BA.changeBAstate(3)
s=LS(4)
suspend()
suspend(111)
BA.changeBAstate(4)
s=LS(5)
exit()
exit(111)
BA.changeBAstate(5)
s=LS(0)