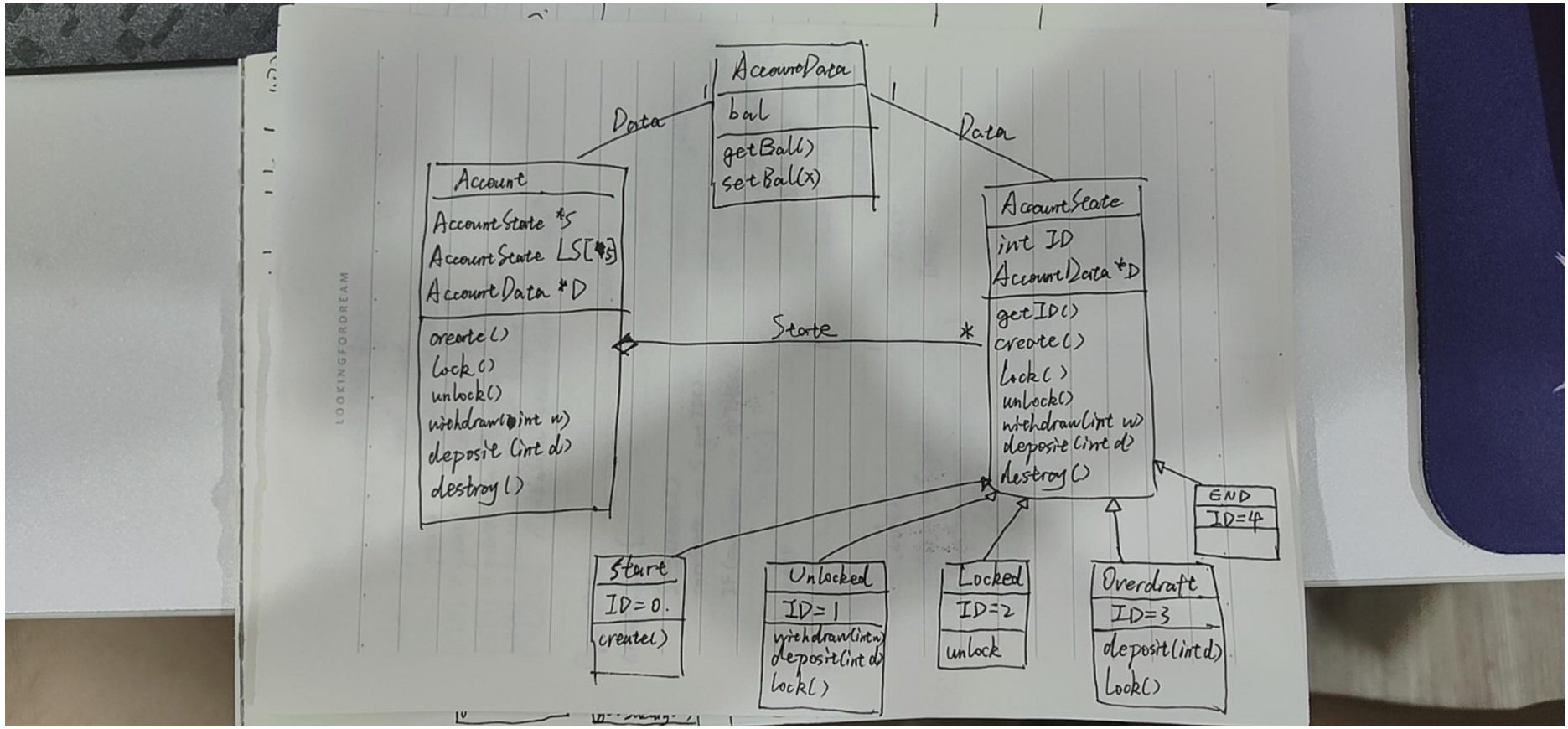
1. A:


AccountData
bal
getBal()
setBal(x)
Data
Data
Account
AccountState *S
AccountState LS[#S]
AccountData *D
create()
lock()
unlock()
withdraw(int w)
deposit(int d)
destroy()
State
*
AccountState
int ID
AccountData *D
getID()
create()
lock()
unlock()
withdraw(int w)
deposit(int d)
destroy()
END
ID=4
Start
ID=0.
create()
Unlocked
ID=1
withdraw(int w)
deposit(int d)
lock()
Locked
ID=2
unlock
Overdraft
ID=3
deposit(int d)
Look()

Class "Account"

```
S      // points to current state object
LS[0]  // points to "Start" Object
LS[1]  // points to "Unlocked" Object
LS[2]  // points to "Locked" Object
LS[3]  // points to "Overdraft" Object
LS[4]  // points to "END" Object
S=LS[0]  // initialize state object to "Start"
```

Operations.

```
create() {
    S->create()
    currState = S.getID()
    IF (currState == 0) THEN
        S=LS[1]
    ENDIF
}
lock() {
    S->Lock()
    currState = S.getID()
    IF (currState == 1) THEN
        S=LS[2]
    ELSE IF (currState == 3) THEN
        S=LS[2]
    ENDIF
}
```

```
unlock() {
    S-> unlock()
    currState = S.getID()
    IF (currState == 2) THEN
        IF (D-> getBal() >= 0) THEN
            S=LS[1]
        ELSE IF (D->getBal() < 0) THEN
            S=LS[3]
        ENDIF
    ENDIF
}
```

```
destroy() {
    S-> destroy()
    currState = S.getID()
    IF (currState == 1) THEN
        S=LS[]
```

```
deposit(int d) {
    S -> deposit(d)
    currState = S.getID()
    IF (currState == 1) THEN
        // no state change
    ELSE IF (currState == 3) THEN
        IF (D -> getBal() + d >= 0) THEN
            S = LS[1]
        ELSE IF (D -> getBal() + d < 0) THEN
            // no state change
        ENDIF
    ENDIF
}

withdraw(int w) {
    S -> withdraw(w)
    currState = S.getID()
    IF (currState == 1) THEN
        IF (D -> getBal() - w > 0) THEN
            // no state change
        ELSE IF (D -> getBal() - w < 0) THEN
            S = LS[3]
        ENDIF
    ENDIF
}
```

Class "Account Create"
Operations
getID() {
    return ID
}

create(), lock(), unlock(), withdraw(int w), deposit(int d),
destroy() are abstract operations.

Class "Account Data"
bal
getBal() {
    return bal
}
setBal(int x) {
    bal = x
}

Class "Start"
ID = 0
→ Operations: create() {
    ID→setBal(0)
}

Class "END"
ID = 4

Class "Unlocked"

ID = 1

Operations

withdraw(int w){

IF (D→getBal() − w > 0) THEN.

D→setBal(D→getBal() − w)

ELSE IF (D→getBal() − w < 0) THEN

D→setBal (D→getBal() − w)

ENDIF

}

deposite(int d) {

D→setBal (D→getBal() + d)

}

lock() {

// nothing to do

}

destroy() {

// nothing to do

}

Class "Locked"

ID = 2

Operations

unlock() {

IF (D→getBal() >= 0) THEN

// nothing to do

ELSE IF (D→getBal() < 0) THEN

// nothing to do

ENDIF

}

Class "Overdraft"

ID=3

operations

deposit (int d) {

IF (D->getBal() + d >= 0) THEN

D-> setBal(D->getBal() + d)

ELSE IF (D->getBal() + d < 0) THEN

D-> setBal (D->getBal() + d)

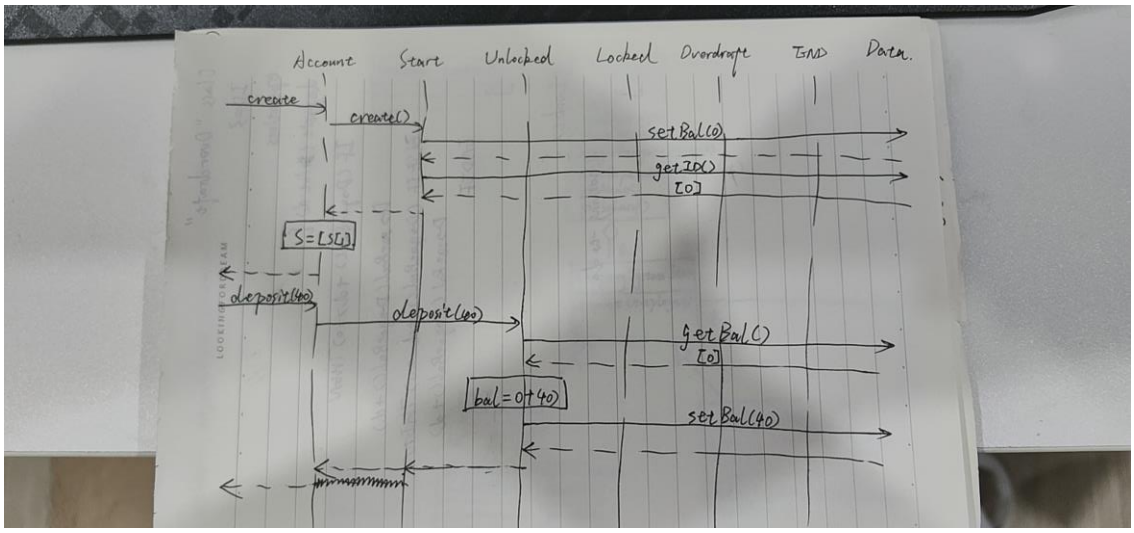
ENDIF

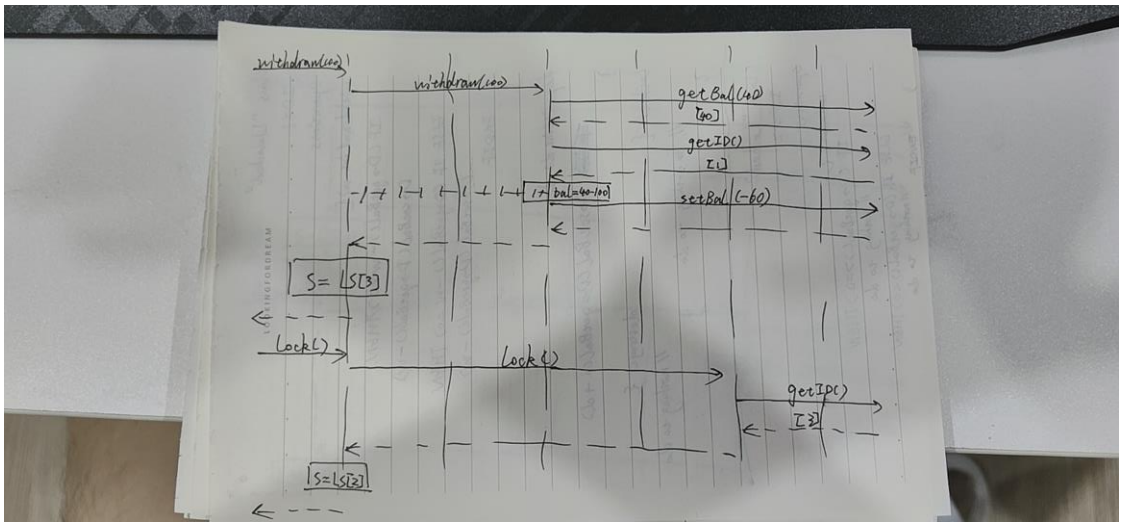
}

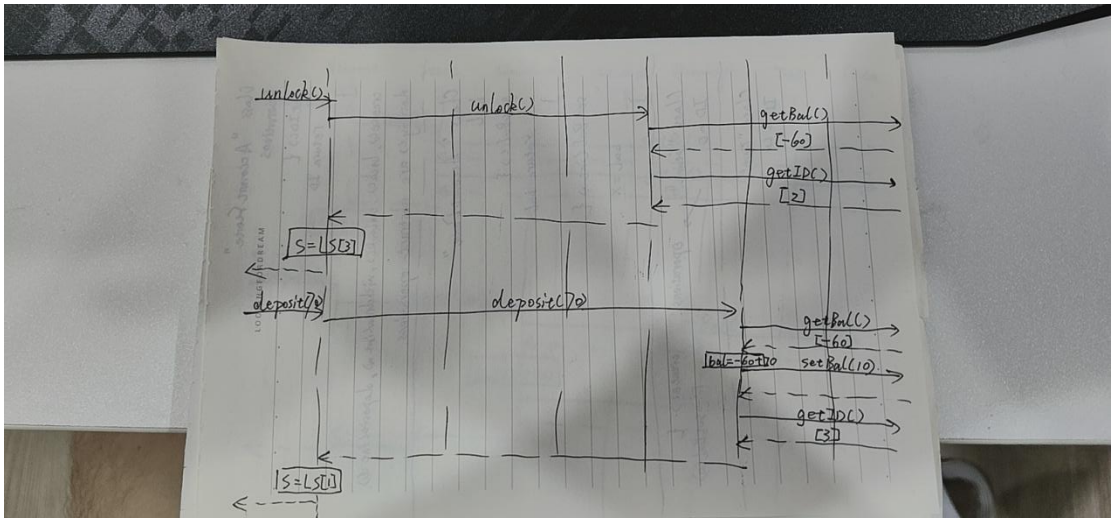
lock() {

// nothing to do

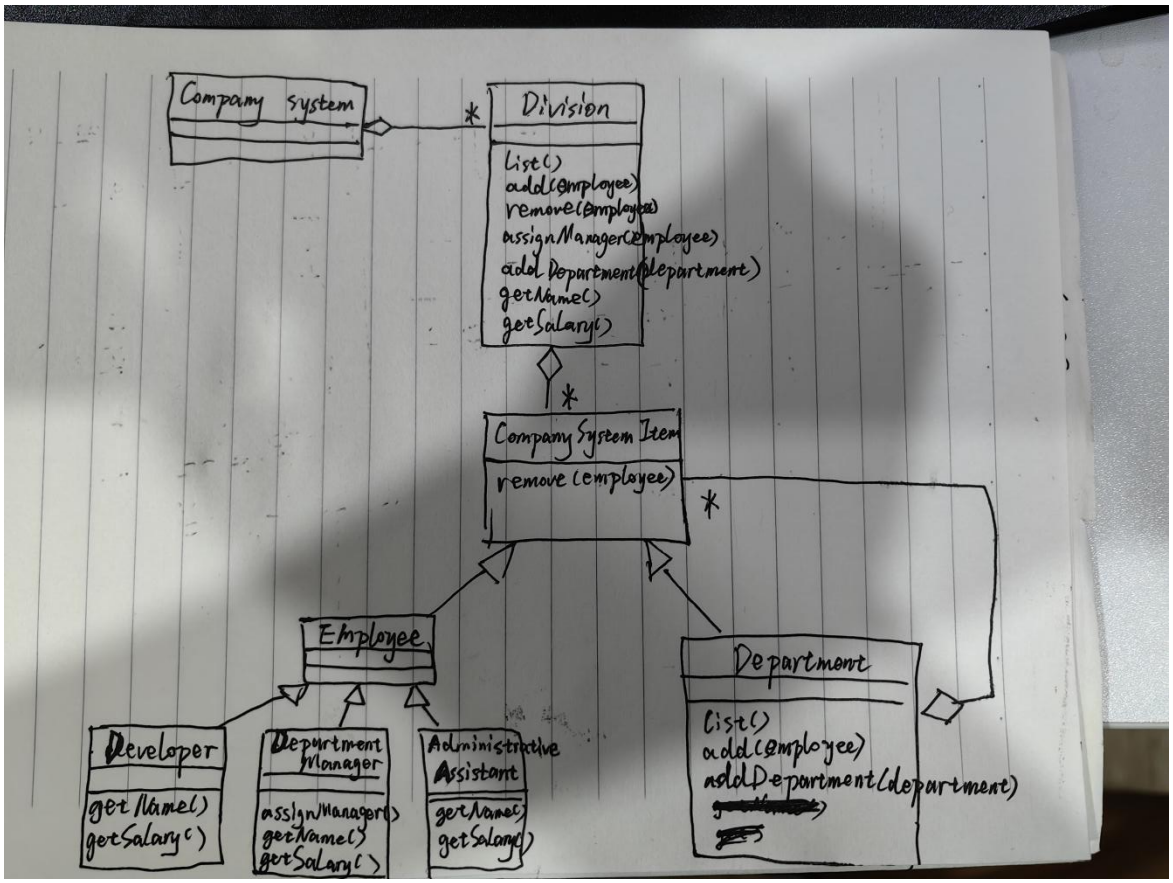
}


Account
Start
Unlocked
Locked
Overdraft
END
Data
create
create()
setBal(0)
getID()
[0]
S=LS[1]
deposit(40)
deposit(40)
getBal()
[0]
bal=0+40
setBal(40)


withdraw(60)
withdraw(60)
getBal(40)
[40]
getID()
[1]
bal=40-100
setBal(-60)
S=LS[3]
Lock()
Lock()
getID()
[3]
S=LS[2]


unlock()
unlock()
getBal()
[-60]
getID()
[2]
S=LS[3]
deposit(70)
deposit(70)
getBal()
[-60]
bal=-60+70
setBal(10)
getID()
[3]
S=LS[1]

2. A:


Company system
Division
list()
addE(employee)
remove(employee)
assignManager(employee)
addDepartment(department)
getName()
getSalary()
Company System Item
remove (employee)
Employee
Department
list()
addE(employee)
addDepartment(department)
Developer
getName()
getSalary()
Department Manager
assignManager()
getName()
getSalary()
Administrative Assistant
getName()
getSalary()