

Retrieve Holdings

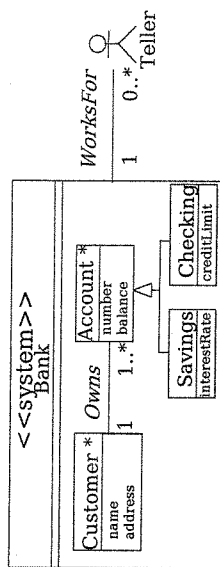
retrieveHoldings

Diagram illustrating the interaction between a Teller and a Bank:

```
sequenceDiagram
    participant Teller
    participant Bank
    Teller->>Bank: retrieveHoldings
    Bank-->>Teller: balance
```

The diagram shows a sequence of messages between a Teller and a Bank. The Teller sends a message labeled "retrieveHoldings" to the Bank, and the Bank returns a message labeled "balance" to the Teller.

<<system>>
Bank



Retrieve Holdings

```
retrieveHoldings (name:String, id:Terminal)
```

Contestant	1: c := find (name)
Contestant 1	
Contestant 2	
Contestant 3	
Contestant 4	
Contestant 5	
Contestant 6	
Contestant 7	
Contestant 8	
Contestant 9	
Contestant 10	
Contestant 11	
Contestant 12	
Contestant 13	
Contestant 14	
Contestant 15	
Contestant 16	
Contestant 17	
Contestant 18	
Contestant 19	
Contestant 20	
Contestant 21	
Contestant 22	
Contestant 23	
Contestant 24	
Contestant 25	
Contestant 26	
Contestant 27	
Contestant 28	
Contestant 29	
Contestant 30	
Contestant 31	
Contestant 32	
Contestant 33	
Contestant 34	
Contestant 35	
Contestant 36	
Contestant 37	
Contestant 38	
Contestant 39	
Contestant 40	
Contestant 41	
Contestant 42	
Contestant 43	
Contestant 44	
Contestant 45	
Contestant 46	
Contestant 47	
Contestant 48	
Contestant 49	
Contestant 50	
Contestant 51	
Contestant 52	
Contestant 53	
Contestant 54	
Contestant 55	
Contestant 56	
Contestant 57	
Contestant 58	
Contestant 59	
Contestant 60	
Contestant 61	
Contestant 62	
Contestant 63	
Contestant 64	
Contestant 65	
Contestant 66	
Contestant 67	
Contestant 68	
Contestant 69	
Contestant 70	
Contestant 71	
Contestant 72	
Contestant 73	
Contestant 74	
Contestant 75	
Contestant 76	
Contestant 77	
Contestant 78	
Contestant 79	
Contestant 80	
Contestant 81	
Contestant 82	
Contestant 83	
Contestant 84	
Contestant 85	
Contestant 86	
Contestant 87	
Contestant 88	
Contestant 89	
Contestant 90	
Contestant 91	
Contestant 92	
Contestant 93	
Contestant 94	
Contestant 95	
Contestant 96	
Contestant 97	
Contestant 98	
Contestant 99	
Contestant 100	

```

1: c := find (name)
2: retrieveHoldings (id:Terminal)

```

```

Z.2: display
operation CustomerManager::retrieveHoldings (name)
    find the customer having the given name (1);
    retrieve holdings for the found customer (2);

method Customer::retrieveHoldings (id:Terminal)
    foreach account of the customer
        get the balance (2.1);
        add it to the total holdings;
    end foreach;

```

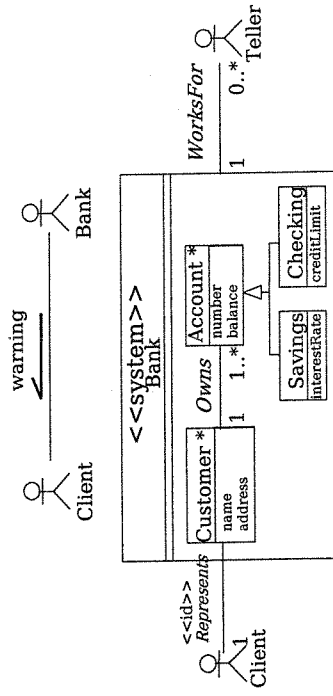
Monthly Verification

Statement

"When monthly verification is performed, the owners of all accounts having a negative balance are warned."

Develop a collaboration diagram for the verify operation

Monthly Verification



Monthly Verification

Message declaration: warning (balance: Integer);

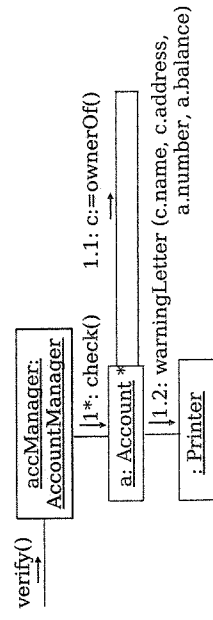
Operation:
Description: Bank::verify ();
Monthly verification of all bank accounts; a warning message is sent to the client if the balance is negative;

Messages:
Aliases: Client::{warning};
overdrawnAccounts: Set (Account) is
self.account->select(a | a.balance < 0);

Pre: true;

Post: overdrawnAccounts ->
forall (a | a.customer.client^warning (a.balance));

Monthly Verification



operation AccountManager::verify()
foreach a: Account
check a;
end foreach;

method Account::check()
if self.balance < 0 then
get the owner of the account (1.1);
print a warning letter for the owner (1.2);
end if;

Retrieve Balance

Develop Collaboration Diagrams for the different versions of the retrieveBalance operation (given as two Operation Schemas)

Show the impact on the Design Class Model.

Retrieve Balance, V1

Operation: Bank::retrieveBalanceV1 (accNum: Integer);
Description: Retrieves the balance of an account with a given account number;

Aliases: accounts: Set (Account) Is
 self.account -> select (a | a.number = accNum);

acc: Account Is accounts -> any ();

Messages: Teller::balance; incorrectNumber_e;

Pre: true;

Post: if accounts -> size () <> 1 then

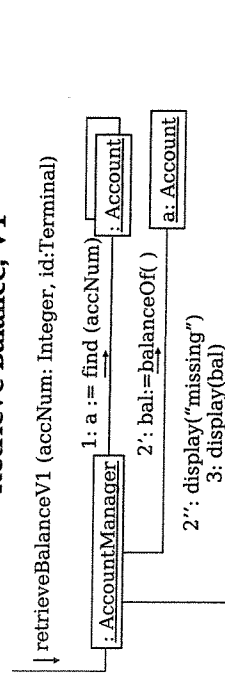
sender\incorrectNumber_e (accNum)

else

sender\balance (acc.balance)

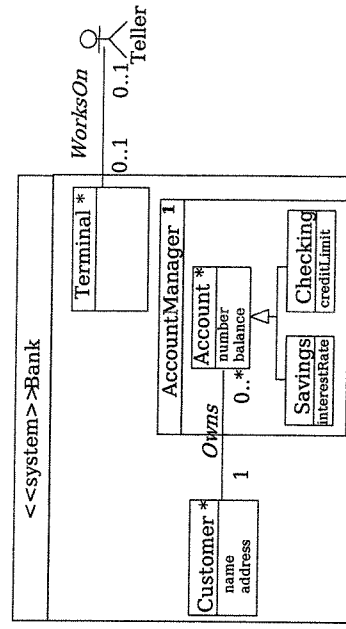
endif;

Retrieve Balance, V1



operation AccountManager:: retrieveBalanceV1 (accNum: Integer, id:Terminal)
 find the account having the given account number (1);
 if there is such an account then
 get its balance (2');
 display the balance on the teller's terminal (3);
 else notify the teller that the account is missing (2'');
 end if.

Retrieve Balance: Draft DCM



Retrieve Balance, V2

Message declaration: balanceOfAccount (accNum:Integer, amount:Integer);

Operation: Bank::retrieveBalanceV2 (name:String);

Description: Retrieves the balances of the accounts of a given customer.

This operation will work correctly with any number of accounts for a customer.

Aliases: cust:Customer Is self.customer -> any(c|c.name=name);

allAccsOfCust: Set (Account) Is cust.account;

Messages: Teller::balanceOfAccount;;

Pre: true;

Post:

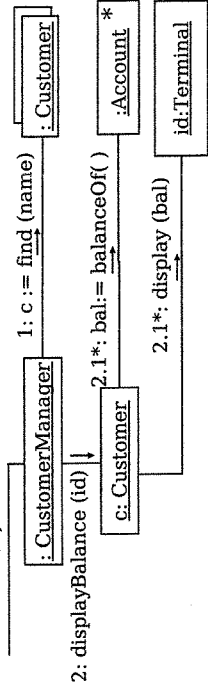
-- For every account of the customer, s/he gets a statement:

allAccsOfCust ->

forAll (a | sender^balanceOfAccount (a.number, a.balance));

Retrieve Balance, V2_1

retrieveBalanceV2 (name:String, id:Terminal)



operation CustomerManager::retrieveBalanceV2 (name:String, id:Terminal)
 find the customer having the given name (1);
 display balance for the found customer (2);

method Customer::displayBalance(id:Terminal)

foreach account of the customer

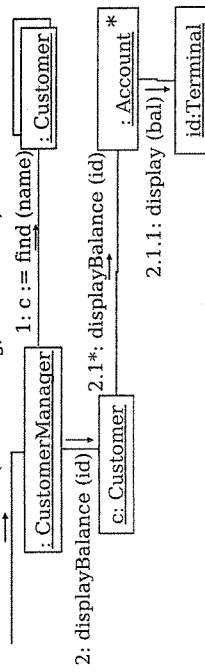
get the balance (2.1);

display the balance on the teller's terminal (2.1);

end foreach.

Retrieve Balance, V2_2

retrieveBalanceV2 (name:String, id:Terminal)



operation Customer::retrieveBalanceV2 (name:String, id:Terminal)

find the customer having the given name (1);

display balance for the found customer (2);

method Customer::displayBalance(id:Terminal)

foreach account of the customer

request the account to display its balance (2.1);

end foreach.

method Account::displayBalance (id:Terminal)

Display the balance on the given terminal (2.1.1);