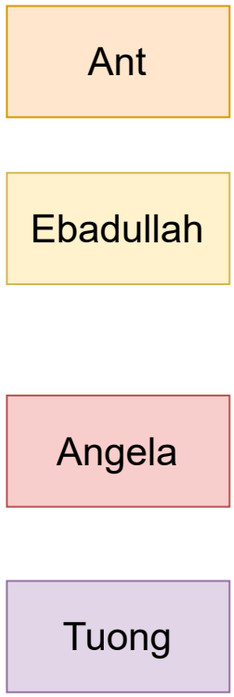
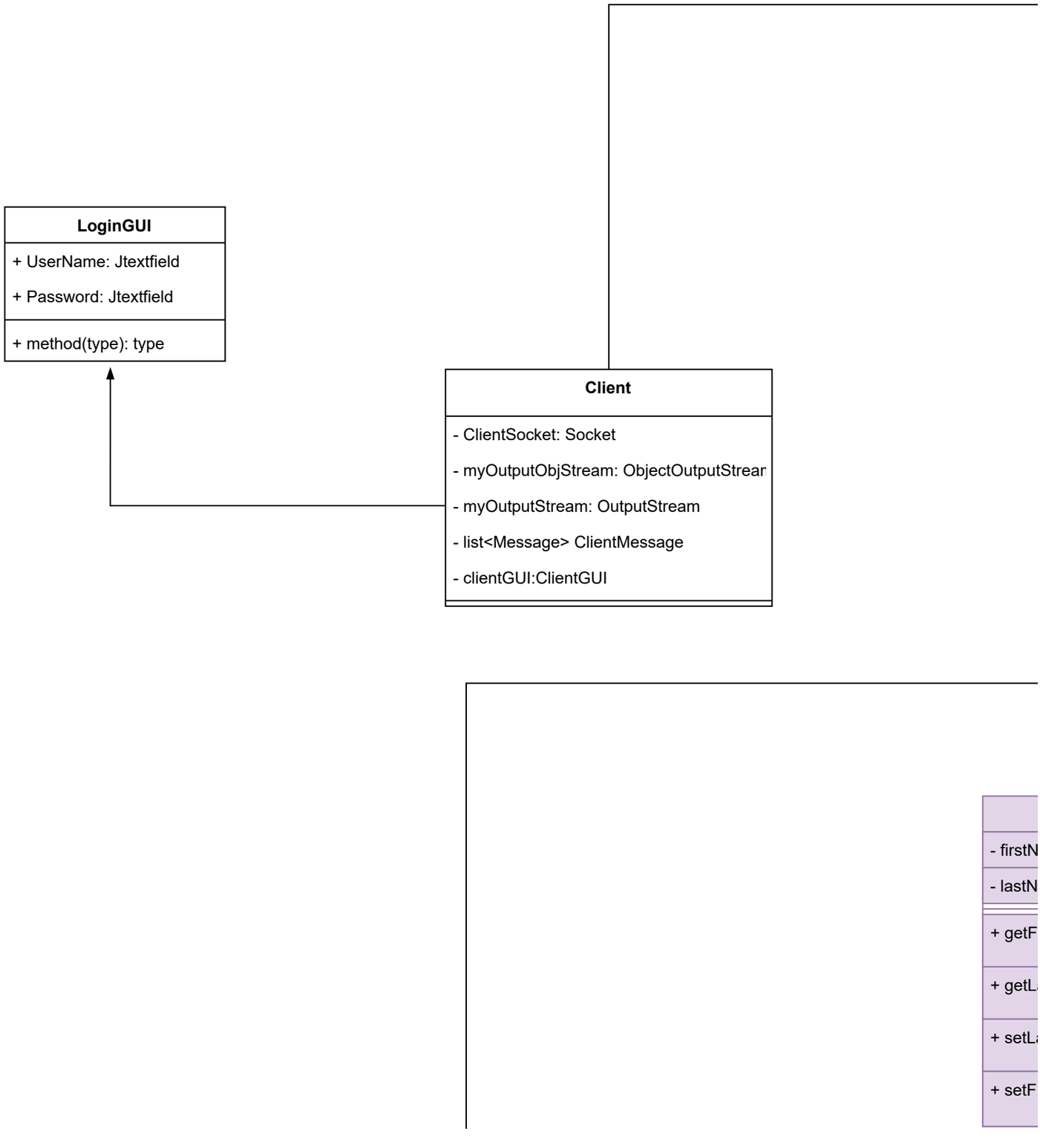
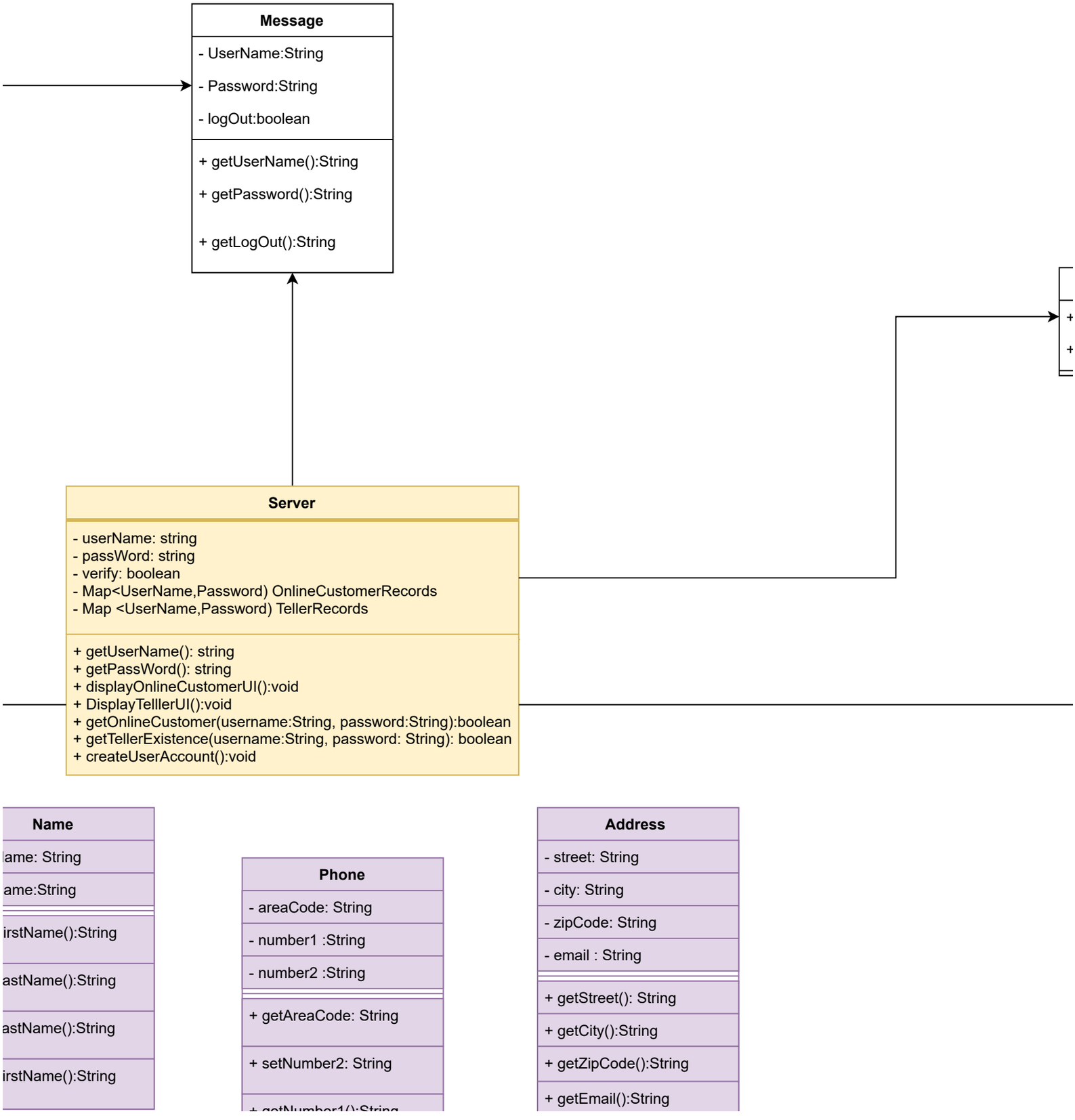
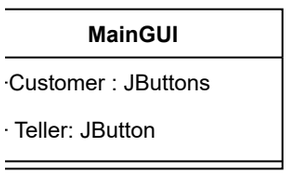
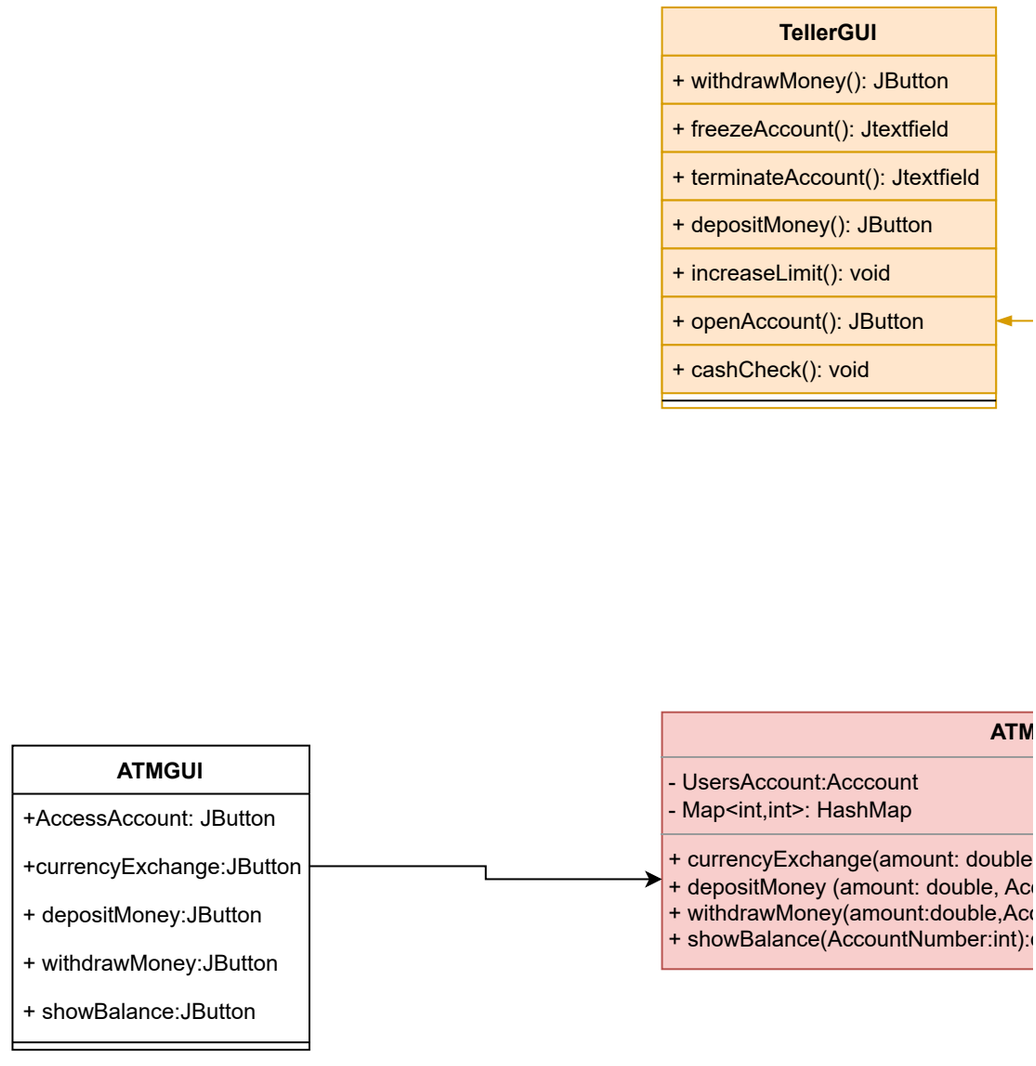
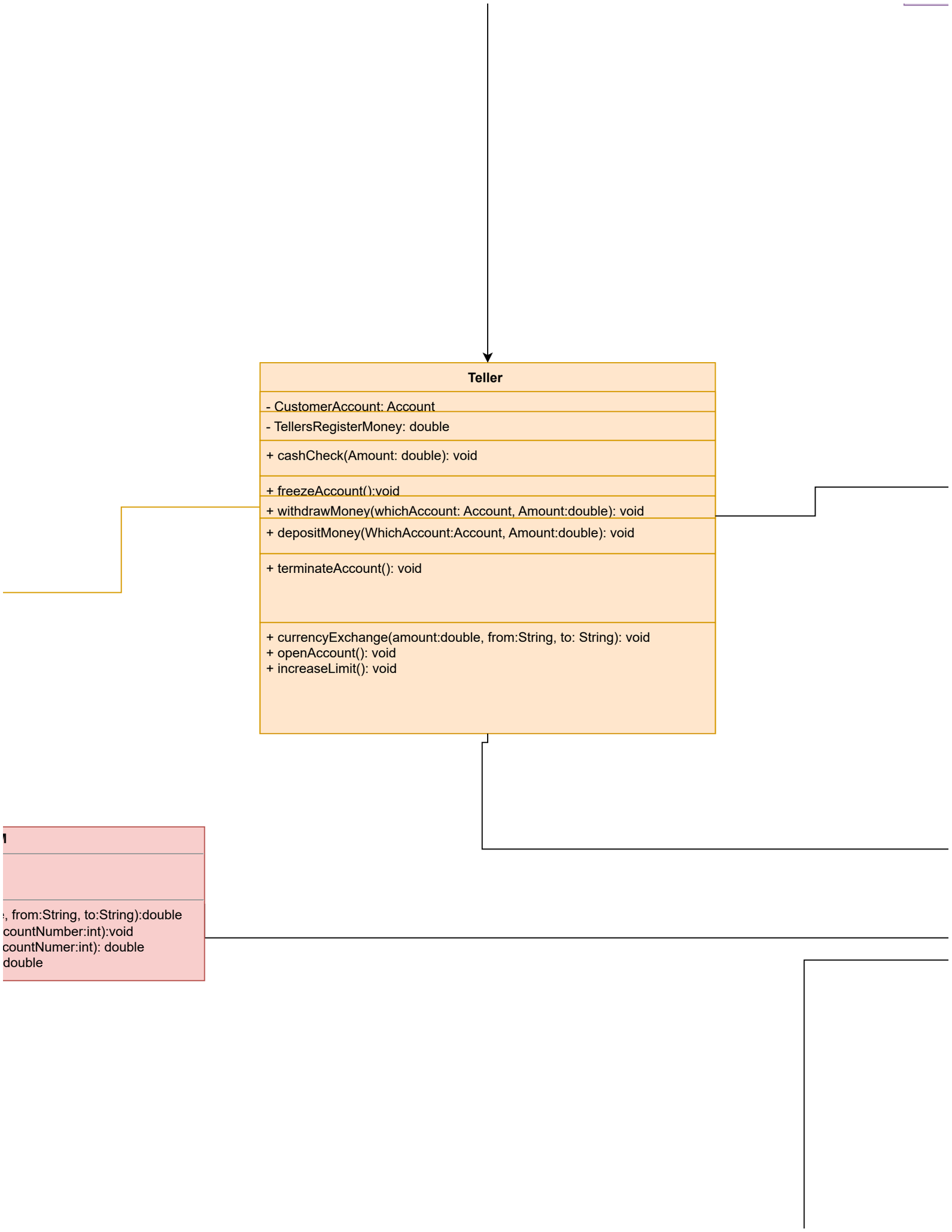

Ant
Ebadullah
Angela
Tuong

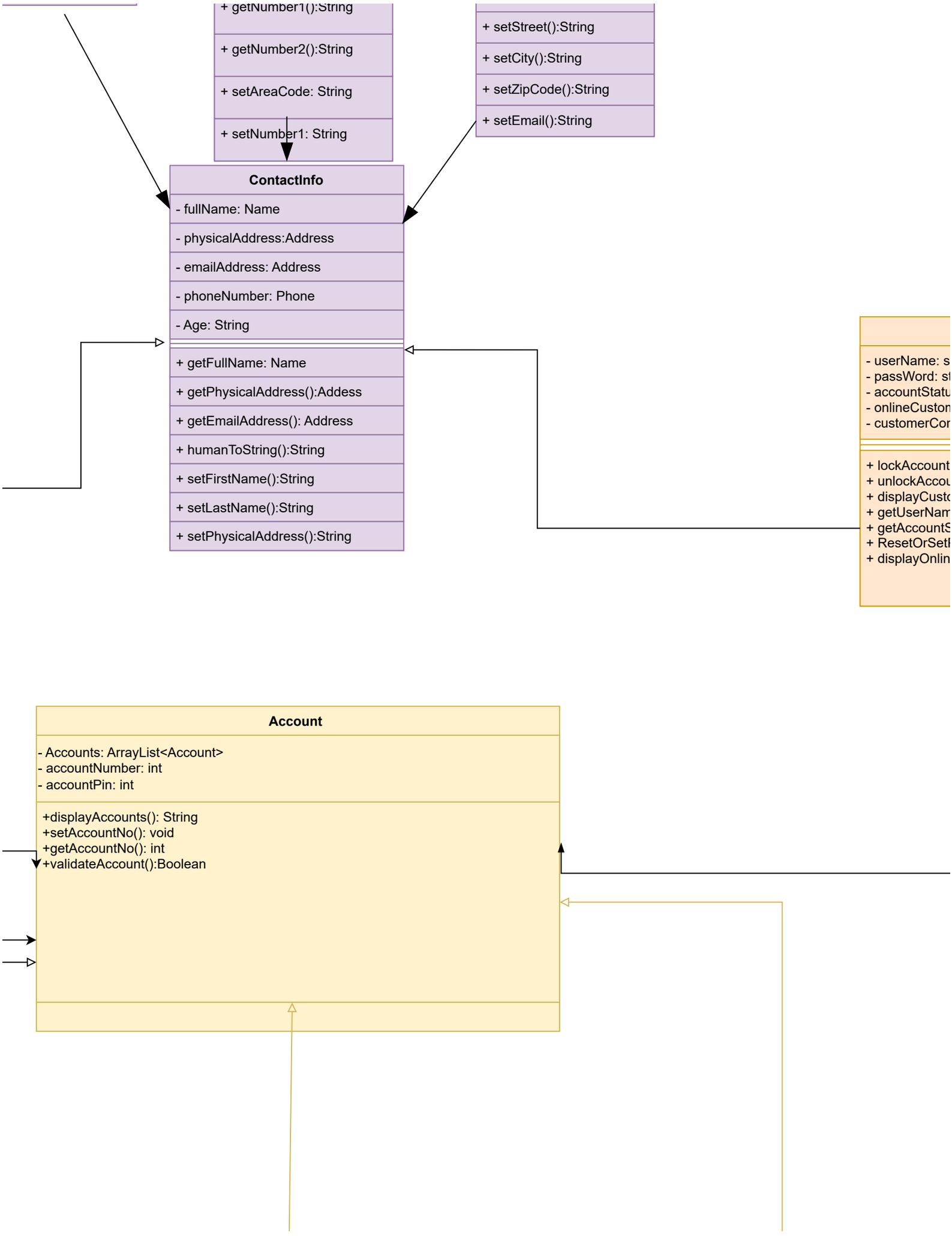

LoginGUI
+ UserName: Jtextfield
+ Password: Jtextfield
+ method(type): type
Client
- ClientSocket: Socket
- myOutputObjStream: ObjectOutputStrear
- myOutputStream: OutputStream
- list<Message> ClientMessage
- clientGUI:ClientGUI
- firstN
- lastN
+ getF
+ getL
+ setL
+ setF

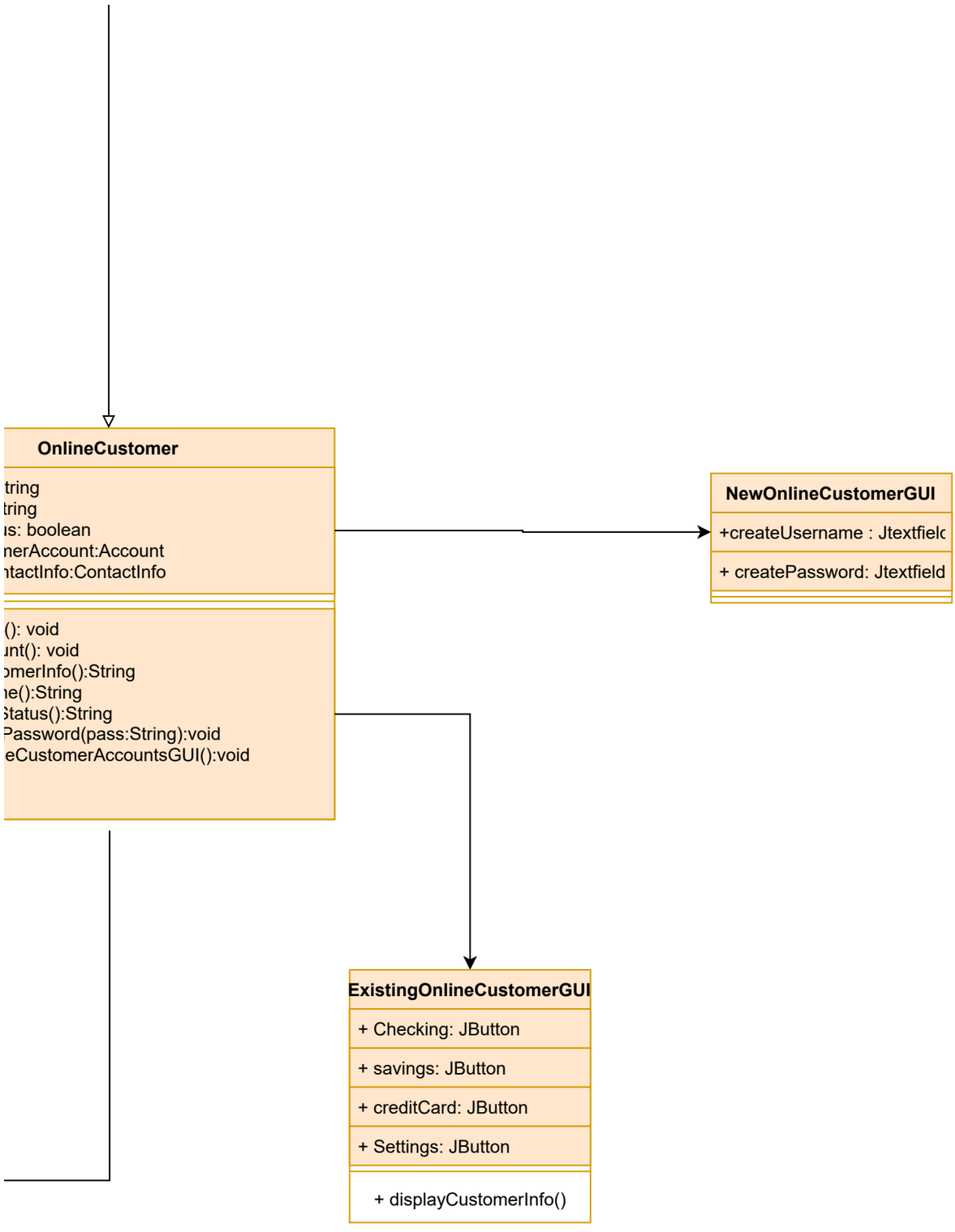

Message
- UserName:String
- Password:String
- logOut:boolean
+ getUserName():String
+ getPassword():String
+ getLogOut():String
Server
- userName: string
- passWord: string
- verify: boolean
- Map<UserName,Password) OnlineCustomerRecords
- Map <UserName,Password) TellerRecords
+ getUserName(): string
+ getPassWord(): string
+ displayOnlineCustomerUI():void
+ DisplayTelllerUI():void
+ getOnlineCustomer(username:String, password:String):boolean
+ getTellerExistence(username:String, password: String): boolean
+ createUserAccount():void
Name
ame: String
ame:String
irstName():String
astName():String
astName():String
irstName():String
Phone
- areaCode: String
- number1 :String
- number2 :String
+ getAreaCode: String
+ setNumber2: String
Address
- street: String
- city: String
- zipCode: String
- email : String
+ getStreet(): String
+ getCity():String
+ getZipCode():String
+ getEmail():String

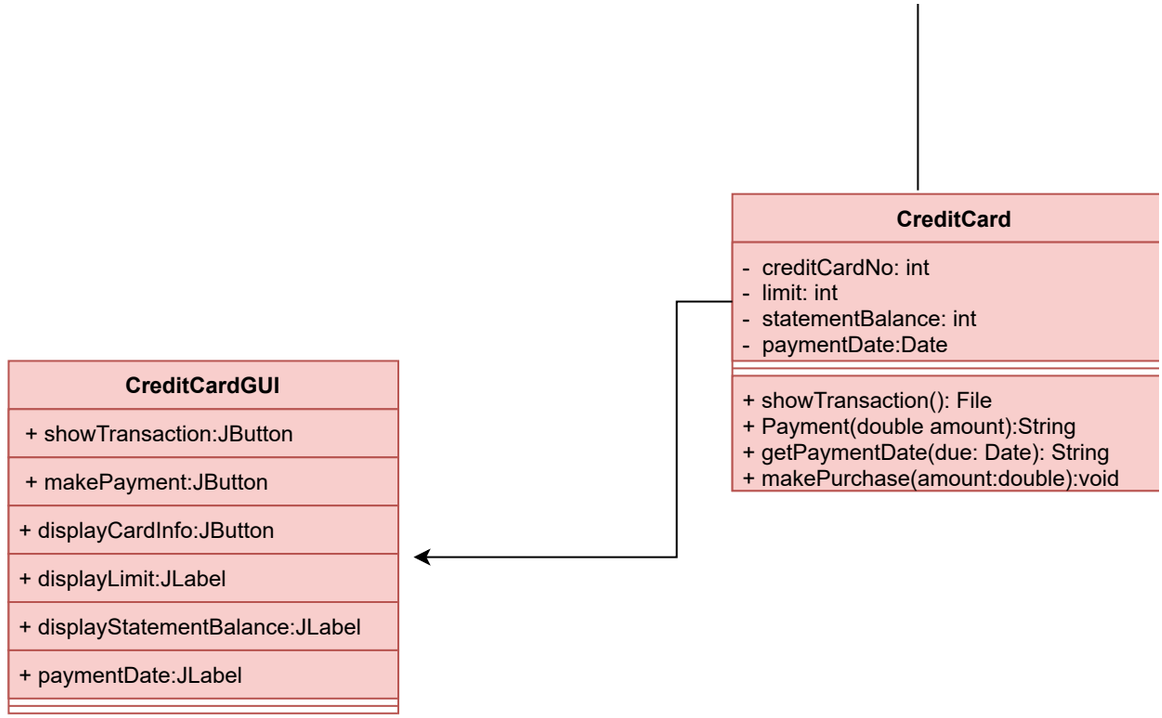

MainGUI
·Customer : JButtons
· Teller: JButton

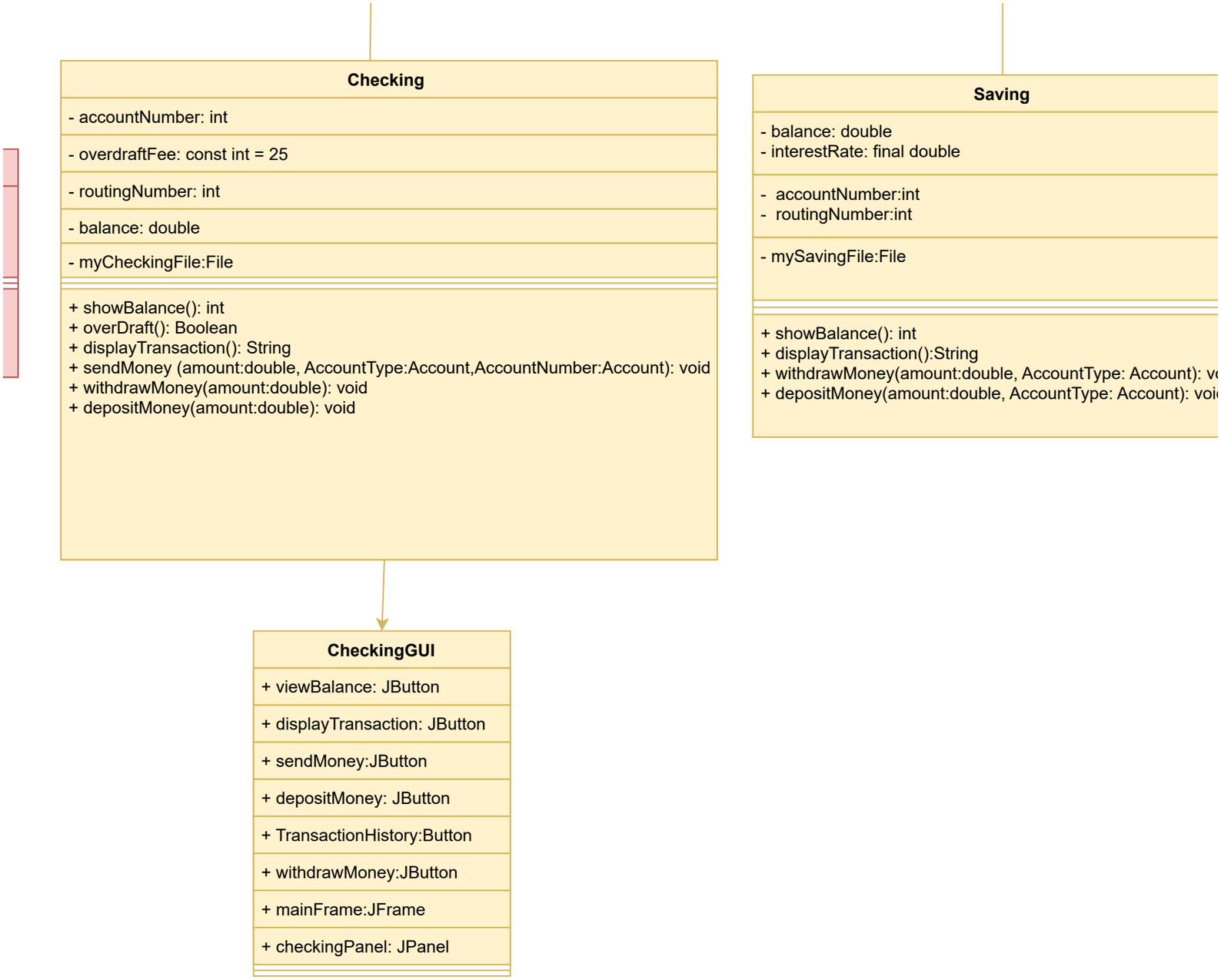

TellerGUI
+ withdrawMoney(): JButton
+ freezeAccount(): Jtextfield
+ terminateAccount(): Jtextfield
+ depositMoney(): JButton
+ increaseLimit(): void
+ openAccount(): JButton
+ cashCheck(): void
ATMGUI
+AccessAccount: JButton
+currencyExchange:JButton
+ depositMoney:JButton
+ withdrawMoney:JButton
+ showBalance:JButton
ATM
- UsersAccount:Acccount
- Map<int,int>: HashMap
+ currencyExchange(amount: double
+ depositMoney (amount: double, Ac
+ withdrawMoney(amount:double,Ac
+ showBalance(AccountNumber:int):

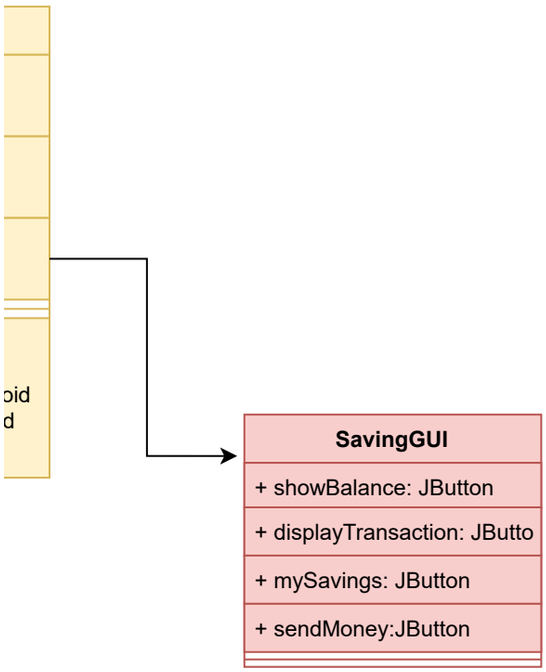

Teller
- CustomerAccount: Account
- TellersRegisterMoney: double
+ cashCheck(Amount: double): void
+ freezeAccount():void
+ withdrawMoney(whichAccount: Account, Amount:double): void
+ depositMoney(WhichAccount:Account, Amount:double): void
+ terminateAccount(): void
+ currencyExchange(amount:double, from:String, to: String): void
+ openAccount(): void
+ increaseLimit(): void
, from:String, to:String):double
countNumber:int):void
countNumer:int): double
double


+ getNumber1():String
+ getNumber2():String
+ setAreaCode: String
+ setNumber1: String
+ setStreet():String
+ setCity():String
+ setZipCode():String
+ setEmail():String
ContactInfo
- fullName: Name
- physicalAddress:Address
- emailAddress: Address
- phoneNumber: Phone
- Age: String
+ getFullName: Name
+ getPhysicalAddress():Addess
+ getEmailAddress(): Address
+ humanToString():String
+ setFirstName():String
+ setLastName():String
+ setPhysicalAddress():String
- userName: s
- passWord: st
- accountStatu
- onlineCustom
- customerCor
+ lockAccount
+ unlockAccou
+ displayCusto
+ getUserNam
+ getAccountS
+ ResetOrSetP
+ displayOnlin
Account
- Accounts: ArrayList<Account>
- accountNumber: int
- accountPin: int
+displayAccounts(): String
+setAccountNo(): void
+getAccountNo(): int
+validateAccount():Boolean


OnlineCustomer
tring
tring
ıs: boolean
nerAccount:Account
ntactInfo:ContactInfo
(): void
unt(): void
omerInfo():String
ne():String
Status():String
Password(pass:String):void
eCustomerAccountsGUI():void
NewOnlineCustomerGUI
+createUsername : Jtextfield
+ createPassword: Jtextfield
ExistingOnlineCustomerGUI
+ Checking: JButton
+ savings: JButton
+ creditCard: JButton
+ Settings: JButton
+ displayCustomerInfo()


CreditCard
- creditCardNo: int
- limit: int
- statementBalance: int
- paymentDate:Date
+ showTransaction(): File
+ Payment(double amount):String
+ getPaymentDate(due: Date): String
+ makePurchase(amount:double):void
CreditCardGUI
+ showTransaction:JButton
+ makePayment:JButton
+ displayCardInfo:JButton
+ displayLimit:JLabel
+ displayStatementBalance:JLabel
+ paymentDate:JLabel


Checking
- accountNumber: int
- overdraftFee: const int = 25
- routingNumber: int
- balance: double
- myCheckingFile:File
+ showBalance(): int
+ overDraft(): Boolean
+ displayTransaction(): String
+ sendMoney (amount:double, AccountType:Account,AccountNumber:Account): void
+ withdrawMoney(amount:double): void
+ depositMoney(amount:double): void
Saving
- balance: double
- interestRate: final double
- accountNumber:int
- routingNumber:int
- mySavingFile:File
+ showBalance(): int
+ displayTransaction():String
+ withdrawMoney(amount:double, AccountType: Account): v
+ depositMoney(amount:double, AccountType: Account): voi
CheckingGUI
+ viewBalance: JButton
+ displayTransaction: JButton
+ sendMoney:JButton
+ depositMoney: JButton
+ TransactionHistory:Button
+ withdrawMoney:JButton
+ mainFrame:JFrame
+ checkingPanel: JPanel


oid
d
SavingGUI
+ showBalance: JButton
+ displayTransaction: JButto
+ mySavings: JButton
+ sendMoney:JButton

Use case ID: 1

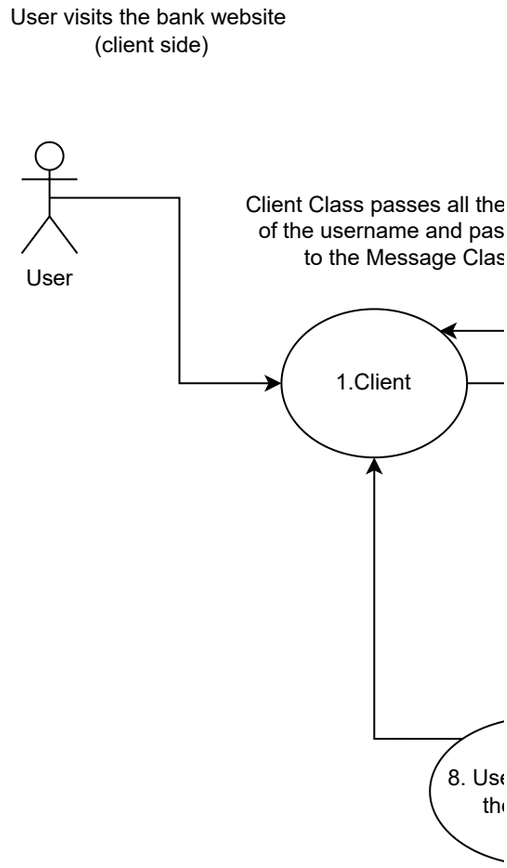
Use case Name: User interacts with the banking online system
Relevant requirements:
Primary Actor: User, Client, Server
Pre-condition: A valid username and password
Post-condition: A deposit made by user to user's checking account
Basic Flow or Main Scenario:

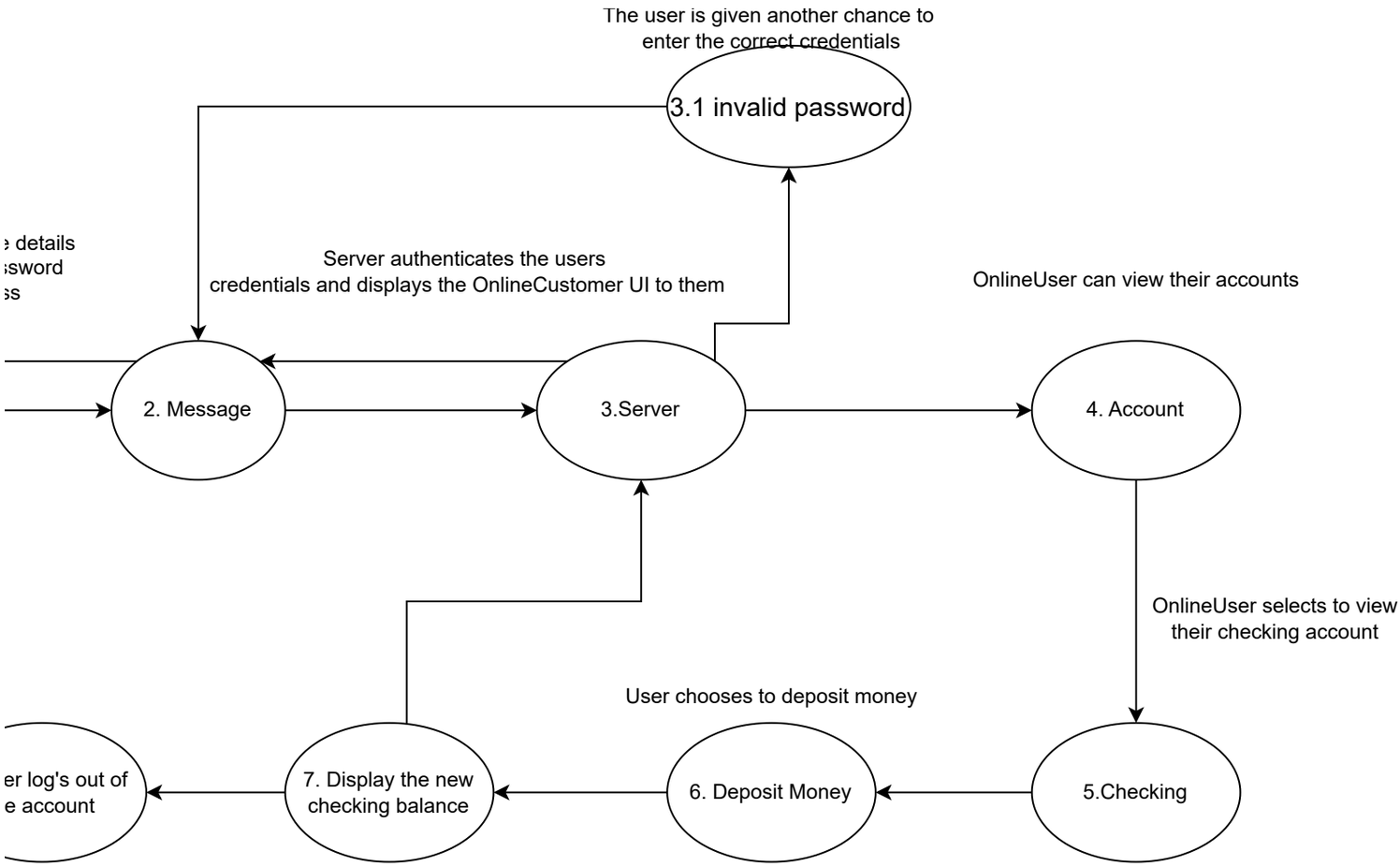
1. Client login to the bank system with username and password
2. The system authenticates the username and password. If it's correct then the system prompts the user to input the correct username and password again.
3. After validating the user, the system will display user's Account
4. User chooses to view the checking account (Alternative choice is 2)
5. User chooses to deposit money to checking account
6. System deposits money to designated checking account
7. System show the current balance after deposit.
8. User chooses to log out
9. System log out user's account

Extension or Alternative Flow:

1. User inputs incorrect username and password.
2. User can choose to view savings or credit card
3. User can Deposit, Withdraw, or view transaction after choosing to view the Checking

The user is given another chance to


User visits the bank website
(client side)
User
Client Class passes all the
of the username and pas
to the Message Clas
1.Client
8. Use
th

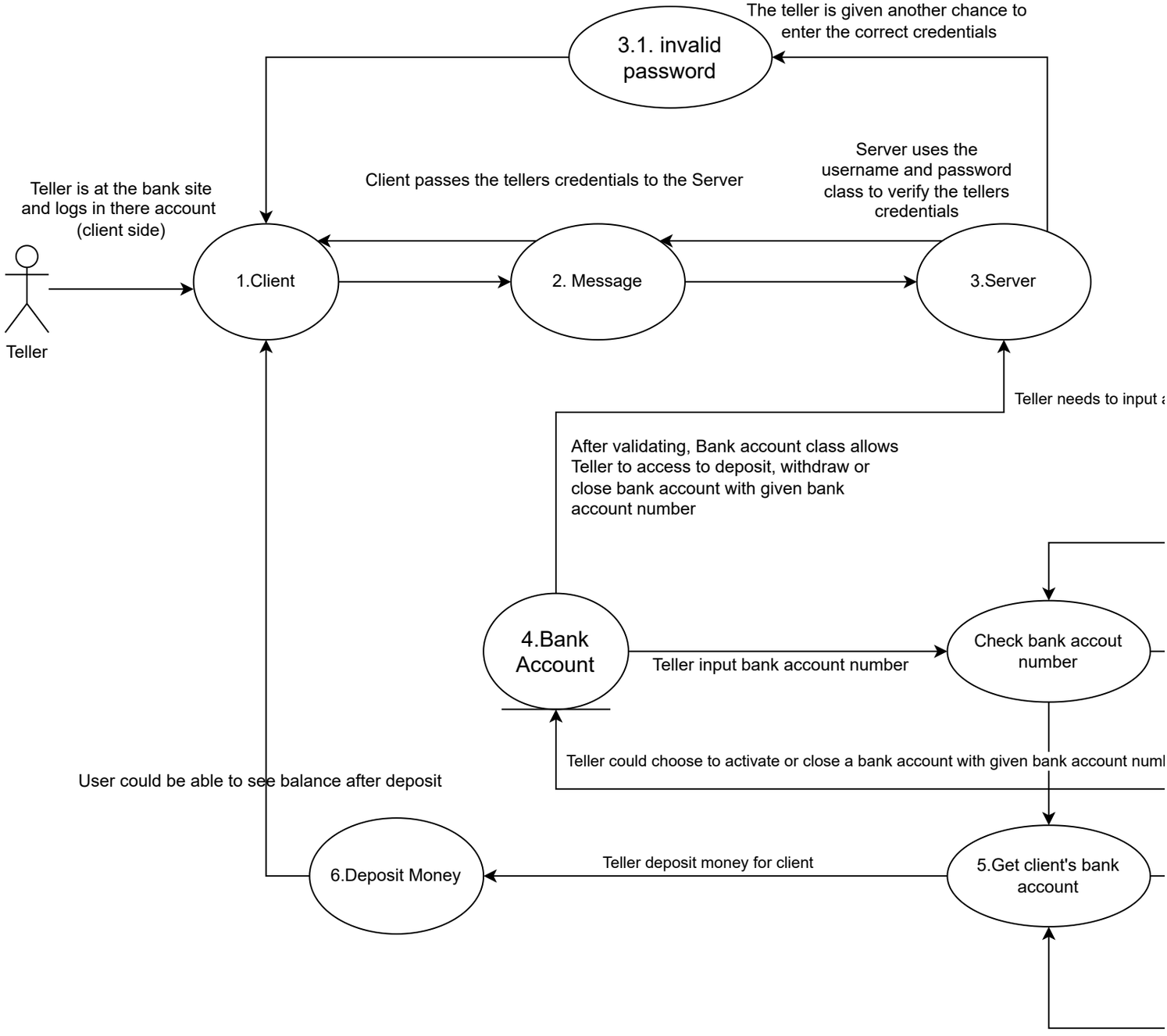

The user is given another chance to
enter the correct credentials
3.1 invalid password
e details
ssword
ss
Server authenticates the users
credentials and displays the OnlineCustomer UI to them
OnlineUser can view their accounts
2. Message
3.Server
4. Account
OnlineUser selects to view
their checking account
User chooses to deposit money
er log's out of
e account
7. Display the new
checking balance
6. Deposit Money
5.Checking

Use case ID: 2
Use case Name: Teller
Relevant requirements:
Primary Actor: Client
Pre-condition: Valid username and password
Post-condition: An deposit made by a teller to a client's bank account
Basic Flow or Main Scenario:

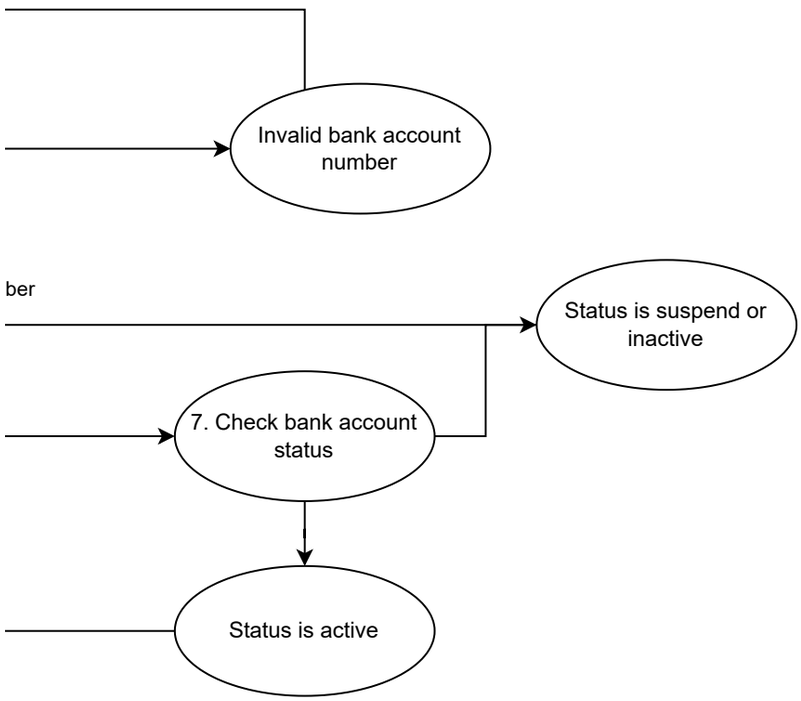
1. Teller login to the bank system with username and password
2. The system authenticates the username and password. If it's correct then the system prompts the user to input the correct username and password again.
3. After validating the user, the system will display teller's account
4. Teller helps a client to deposit money to bank account
5. System prompts bank account number
6. Teller inputs client's bank account number and chooses deposit
7. System deposits money to client's bank account
8. System show current balance after deposit
9. Teller log out
10. System log out teller's account

Extension or Alternative Flow:
1. User inputs incorrect username and password.
2. Bank account status is inactive or suspended
3. Teller chooses to activate it again or terminate it.
4. System reactivates the client's bank account
5. Teller chooses to cash check, withdraw money, exchange currency, increase limit

Exception: None


The teller is given another chance to enter the correct credentials
3.1. invalid password
Server uses the username and password class to verify the tellers credentials
Client passes the tellers credentials to the Server
Teller is at the bank site and logs in there account (client side)
1.Client
2. Message
3.Server
Teller
Teller needs to input a
After validating, Bank account class allows Teller to access to deposit, withdraw or close bank account with given bank account number
4.Bank Account
Teller input bank account number
Check bank accout number
Teller could choose to activate or close a bank account with given bank account numb
User could be able to see balance after deposit
6.Deposit Money
Teller deposit money for client
5.Get client's bank account

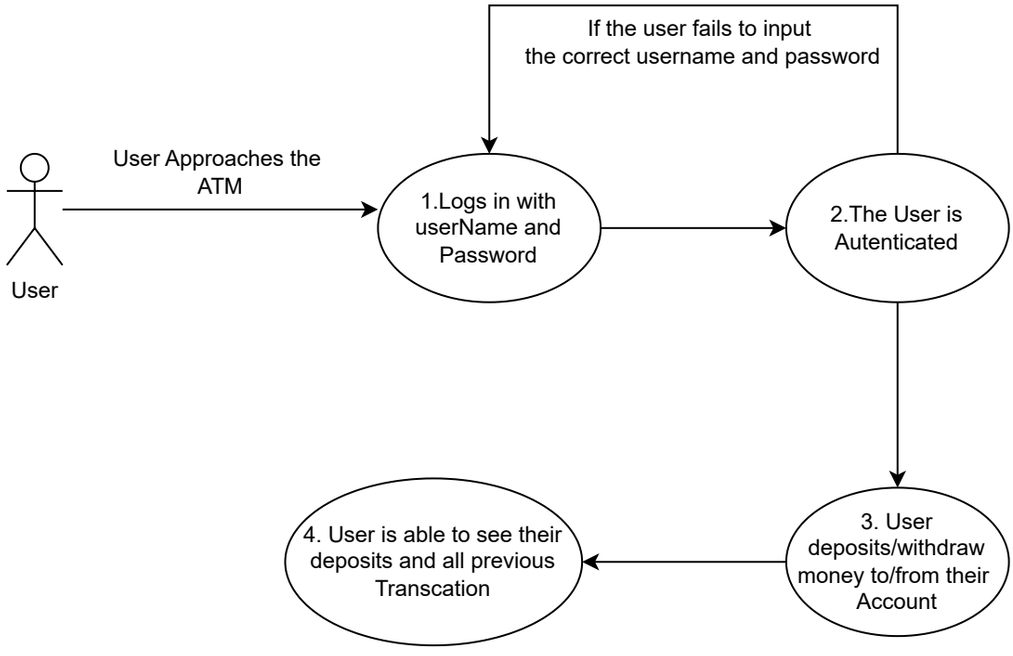
a valid bank account number

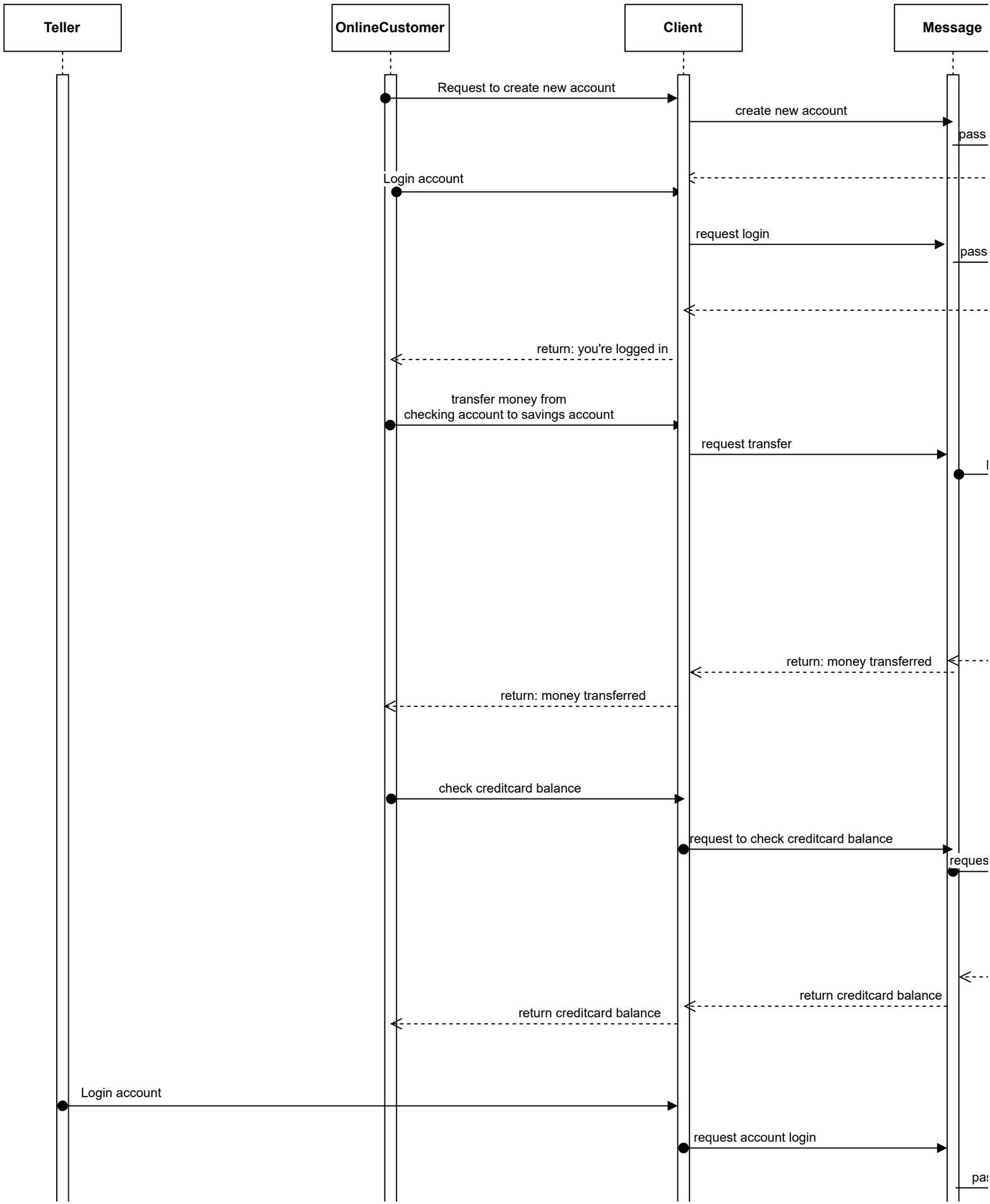


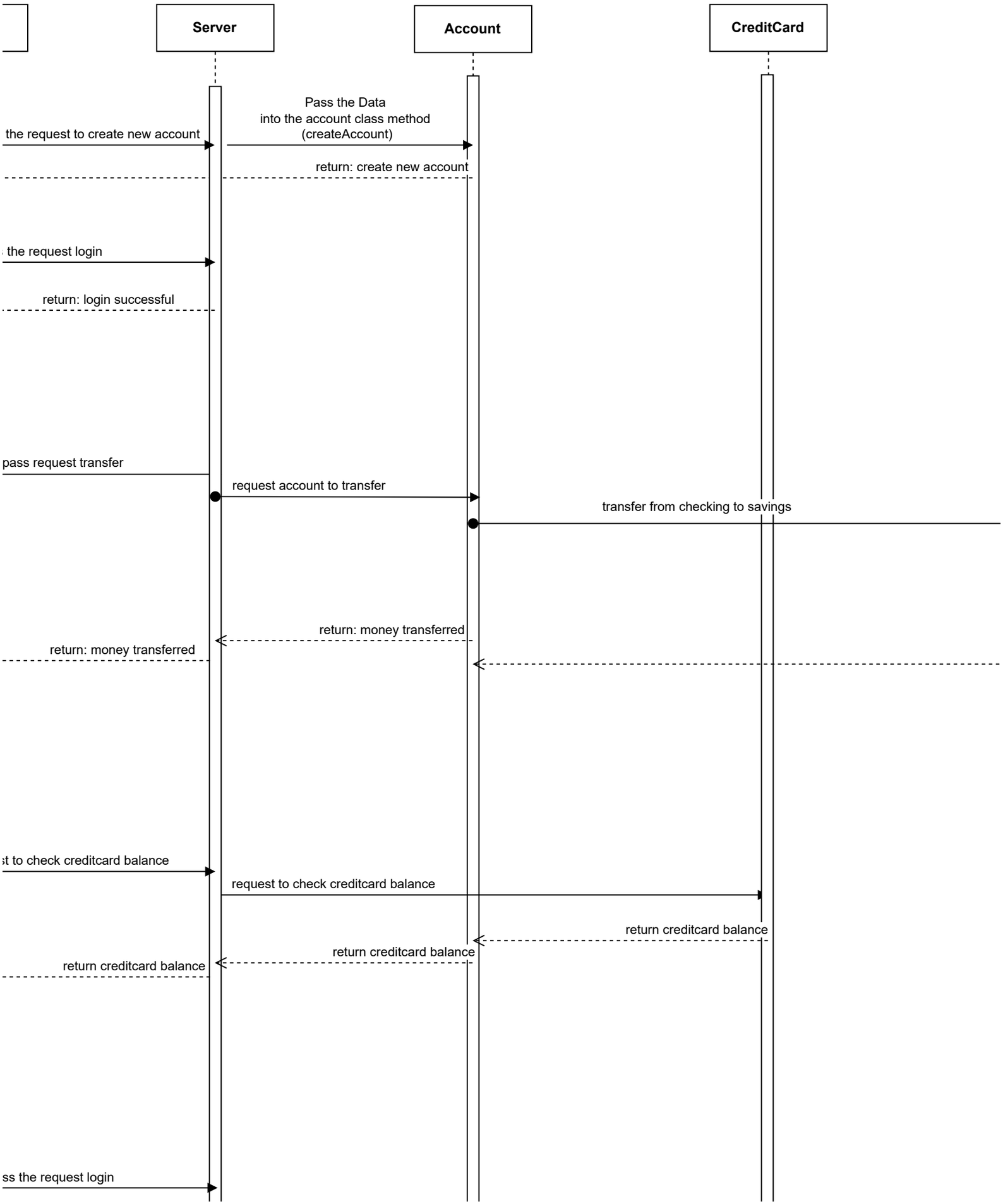
Use case ID: 3
Use case name: ATM
Relevant requirements:
Primary Actor: Client
Pre-condition: Valid username and password
Post-condition: None
Basic Flow or Main Scenario:
1. User chose to login
2. ATM verifies user input
3. ATM shows user bank account and prompt actions: deposit, show balance, withdraw, currency exchange
4. User chooses to deposit cash/check
5. ATM takes the user's input amount to deposit to the user's bank account
6. ATM then prompts actions after completion.
7. User chooses to show balance.
8. ATM shows user's bank account balance
9. ATM prompts action
10. User chooses to withdraw money
11. ATM withdraws money from user's bank account
12. User chooses to do log out
13. System safely log out user's acccount

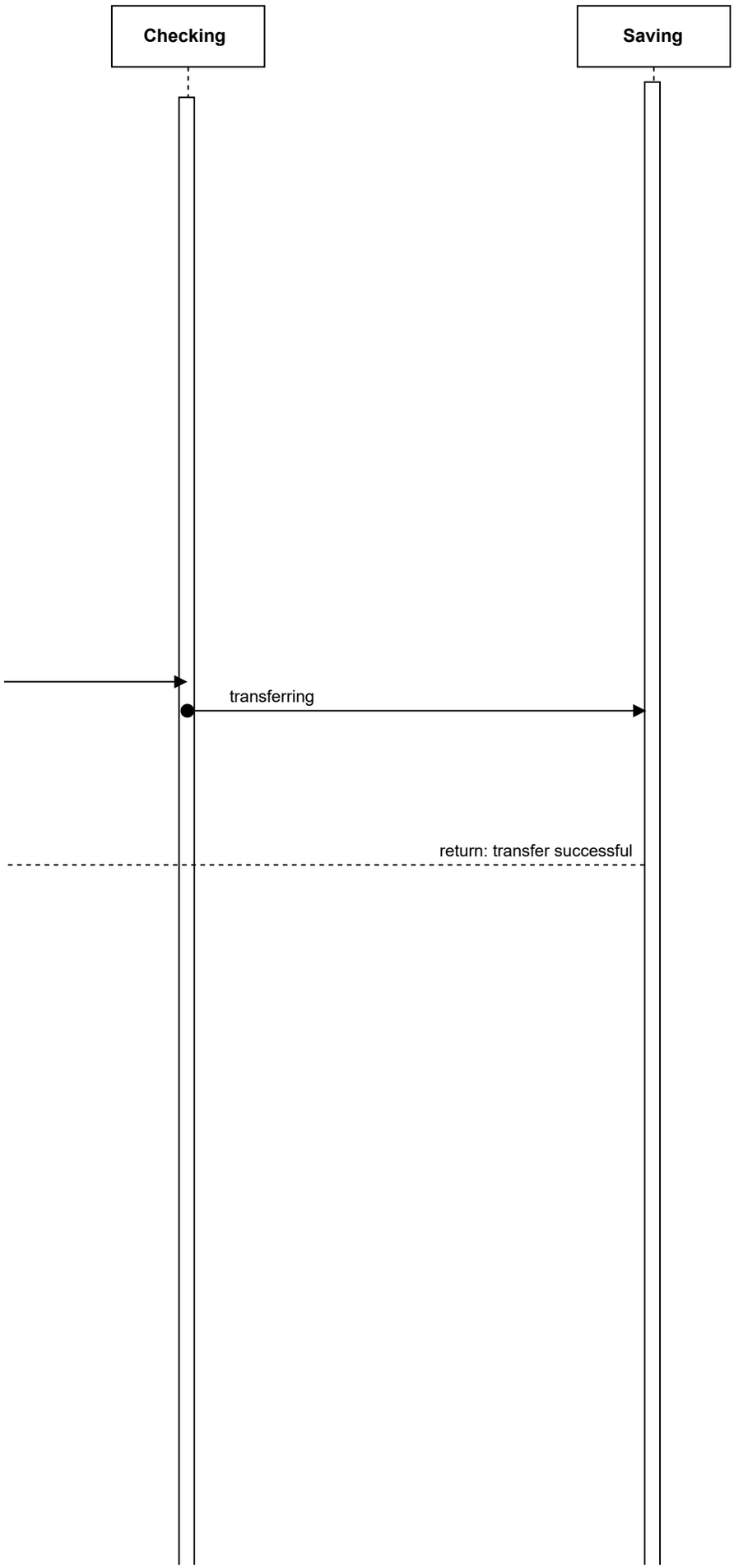
Extension or Alternative Flow:
1. User inputs incorrect username and password.

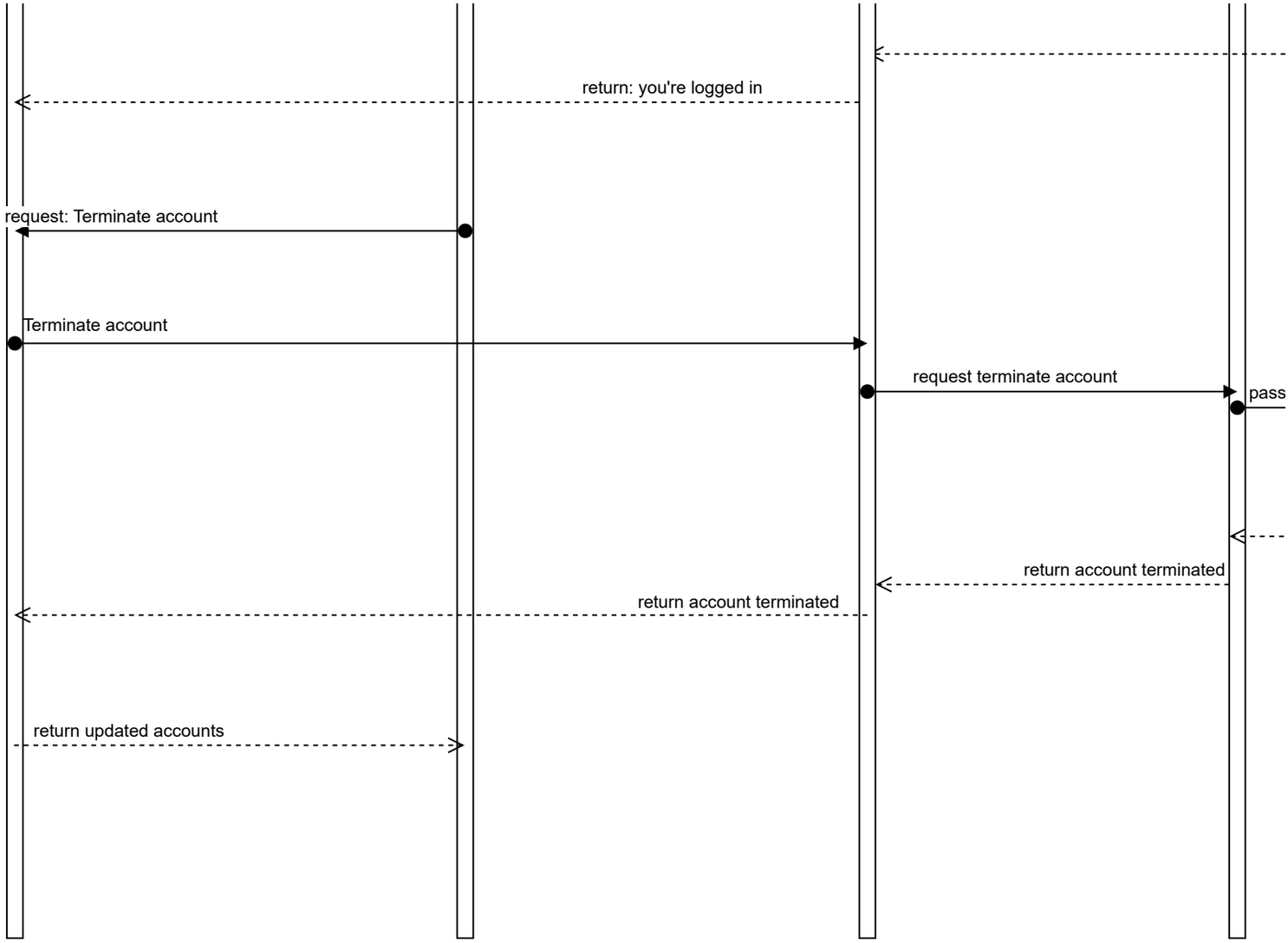
Exception: None

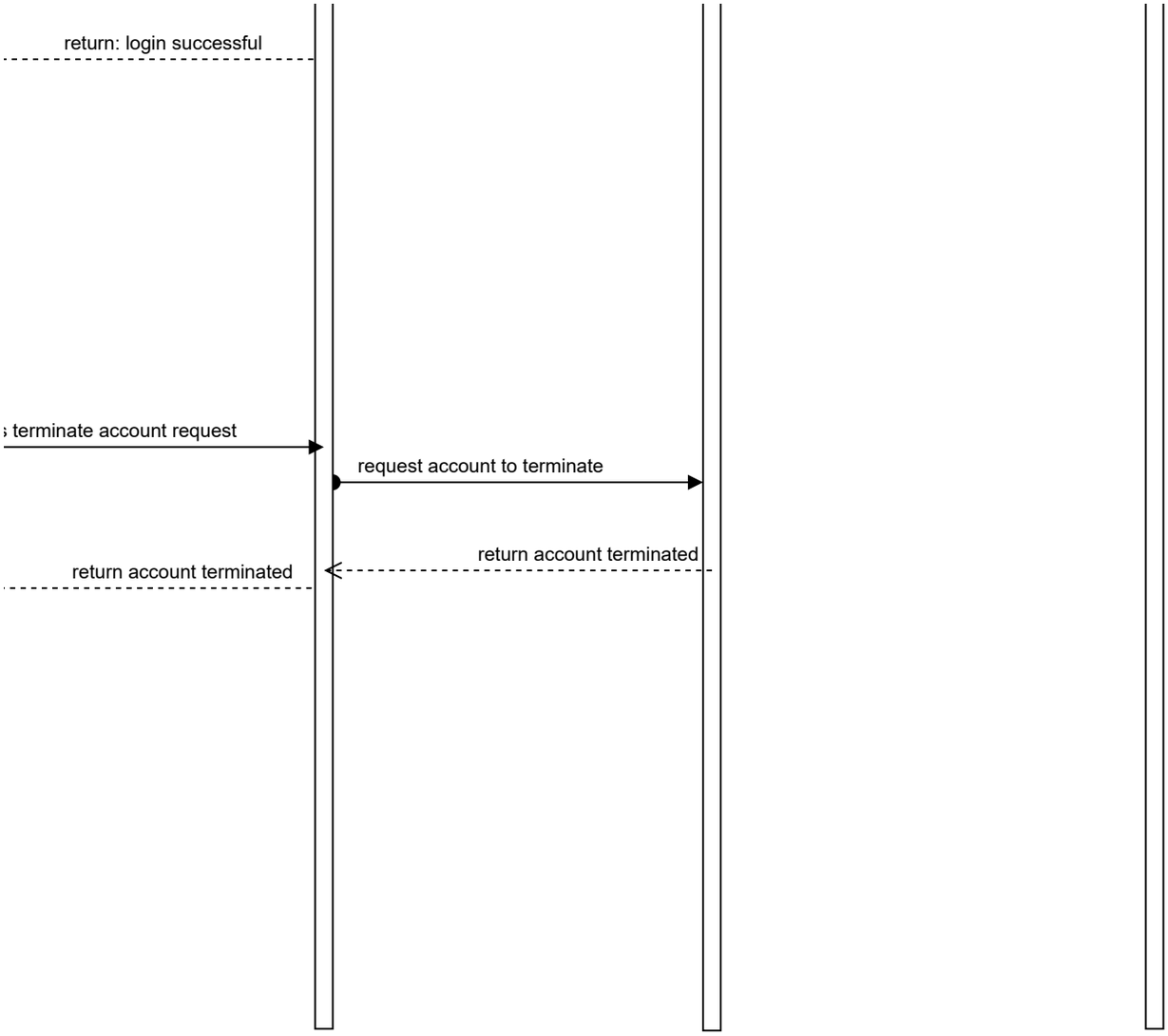


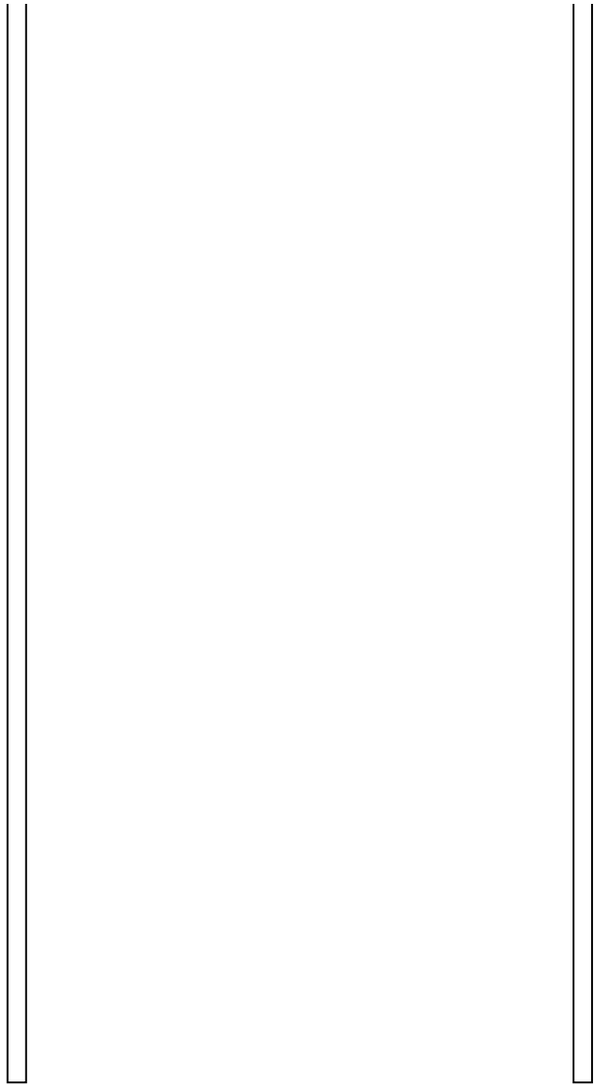

Teller
OnlineCustomer
Client
Message
Request to create new account
create new account
pass
Login account
request login
pass
return: you're logged in
transfer money from
checking account to savings account
request transfer
return: money transferred
return: money transferred
check creditcard balance
request to check creditcard balance
reques
return creditcard balance
return creditcard balance
Login account
request account login
pas

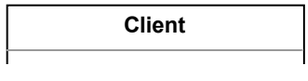

Server
Account
CreditCard
Pass the Data
into the account class method
(createAccount)
the request to create new account
return: create new account
the request login
return: login successful
pass request transfer
request account to transfer
transfer from checking to savings
return: money transferred
return: money transferred
st to check creditcard balance
request to check creditcard balance
return creditcard balance
return creditcard balance
return creditcard balance
ss the request login

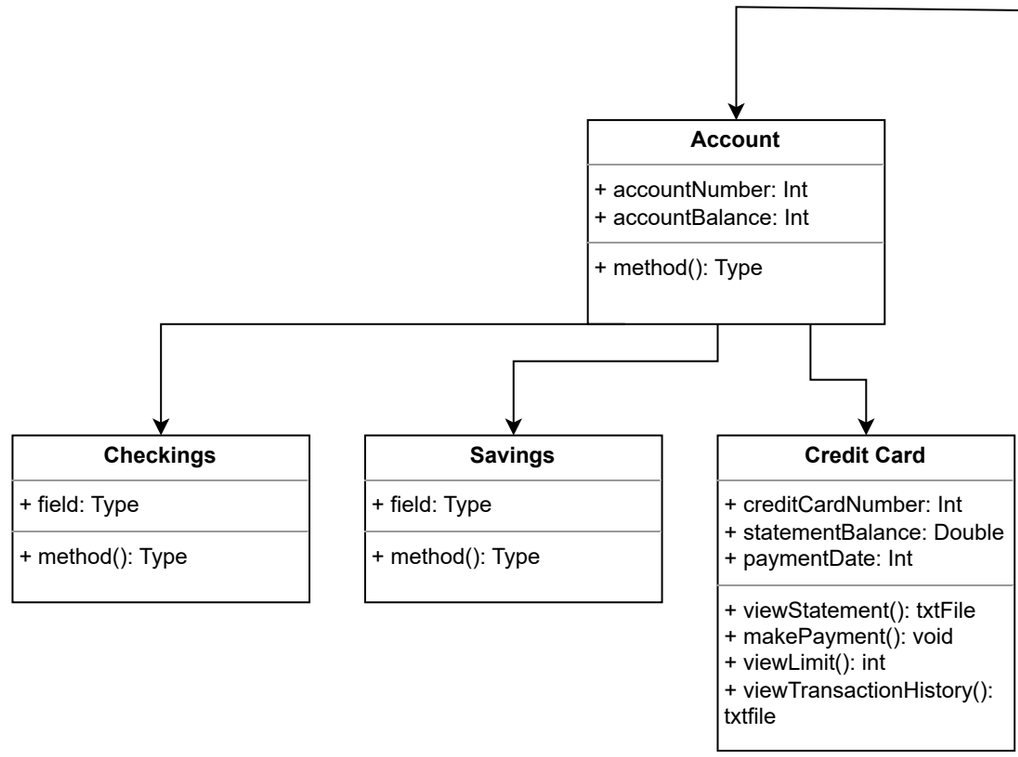

Checking
Saving
transferring
return: transfer successful

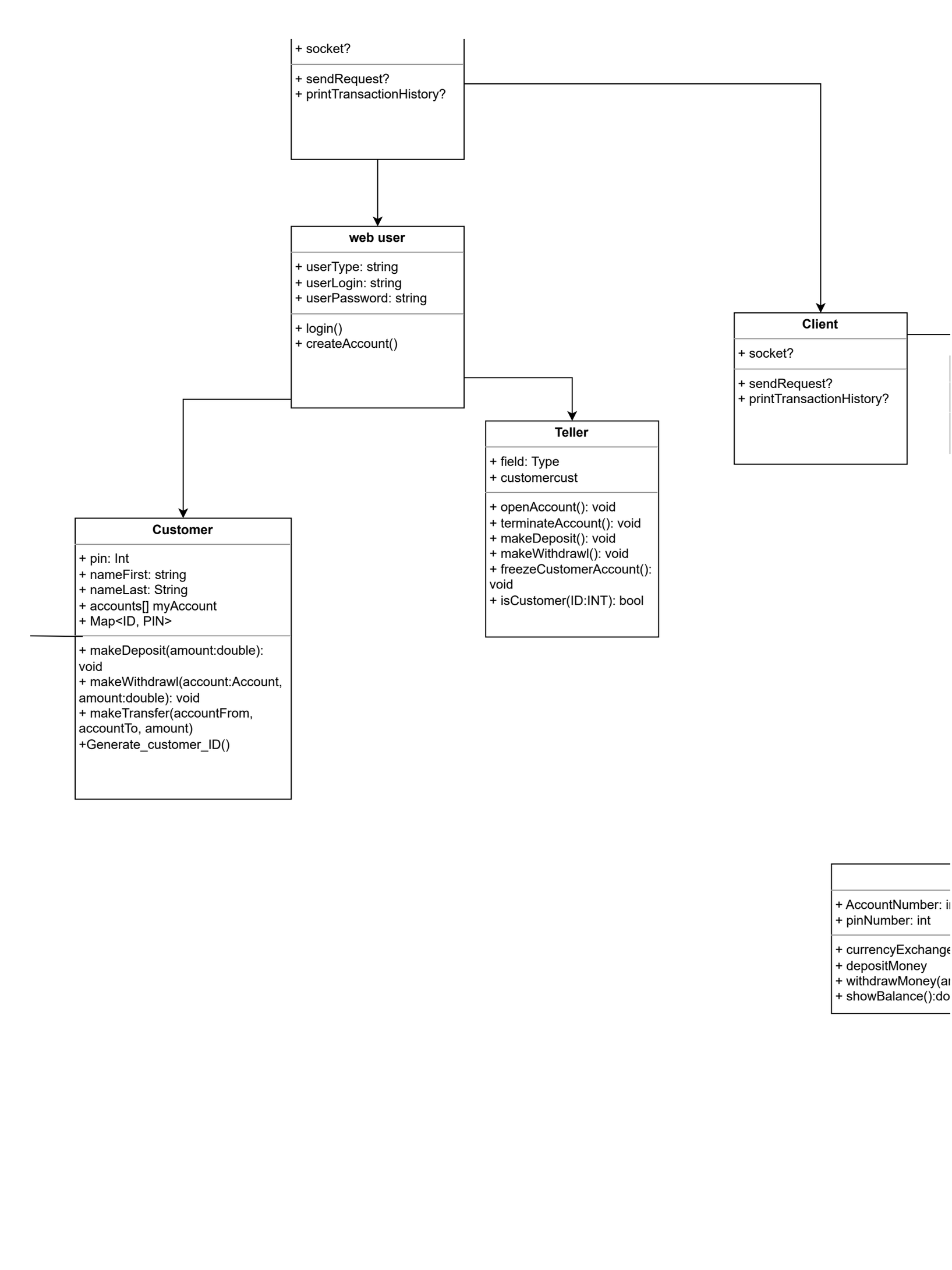

return: you're logged in
request: Terminate account
Terminate account
request terminate account
pass
return account terminated
return account terminated
return updated accounts

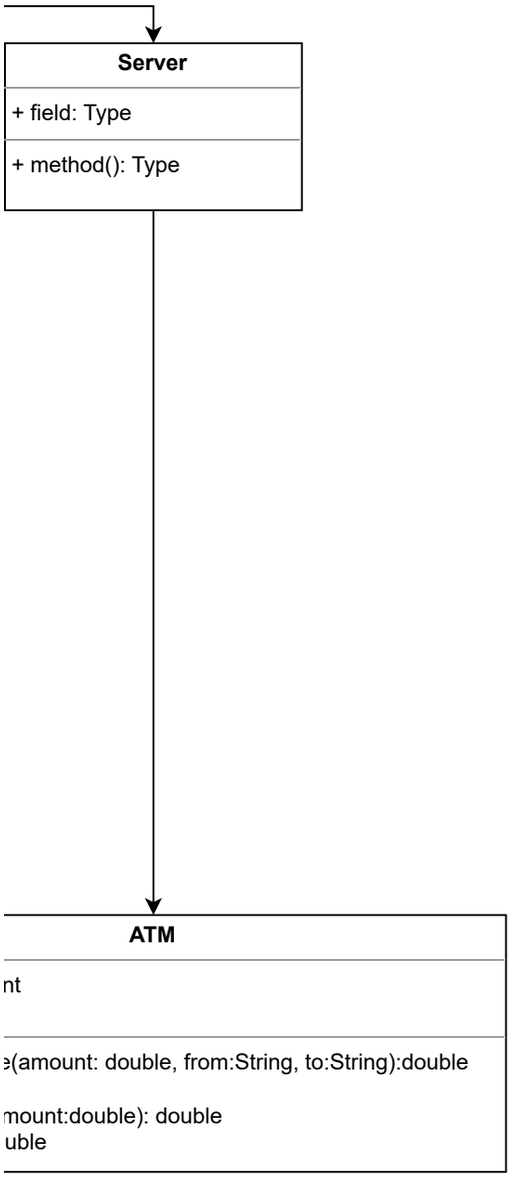

return: login successful
s terminate account request
request account to terminate
return account terminated
return account terminated

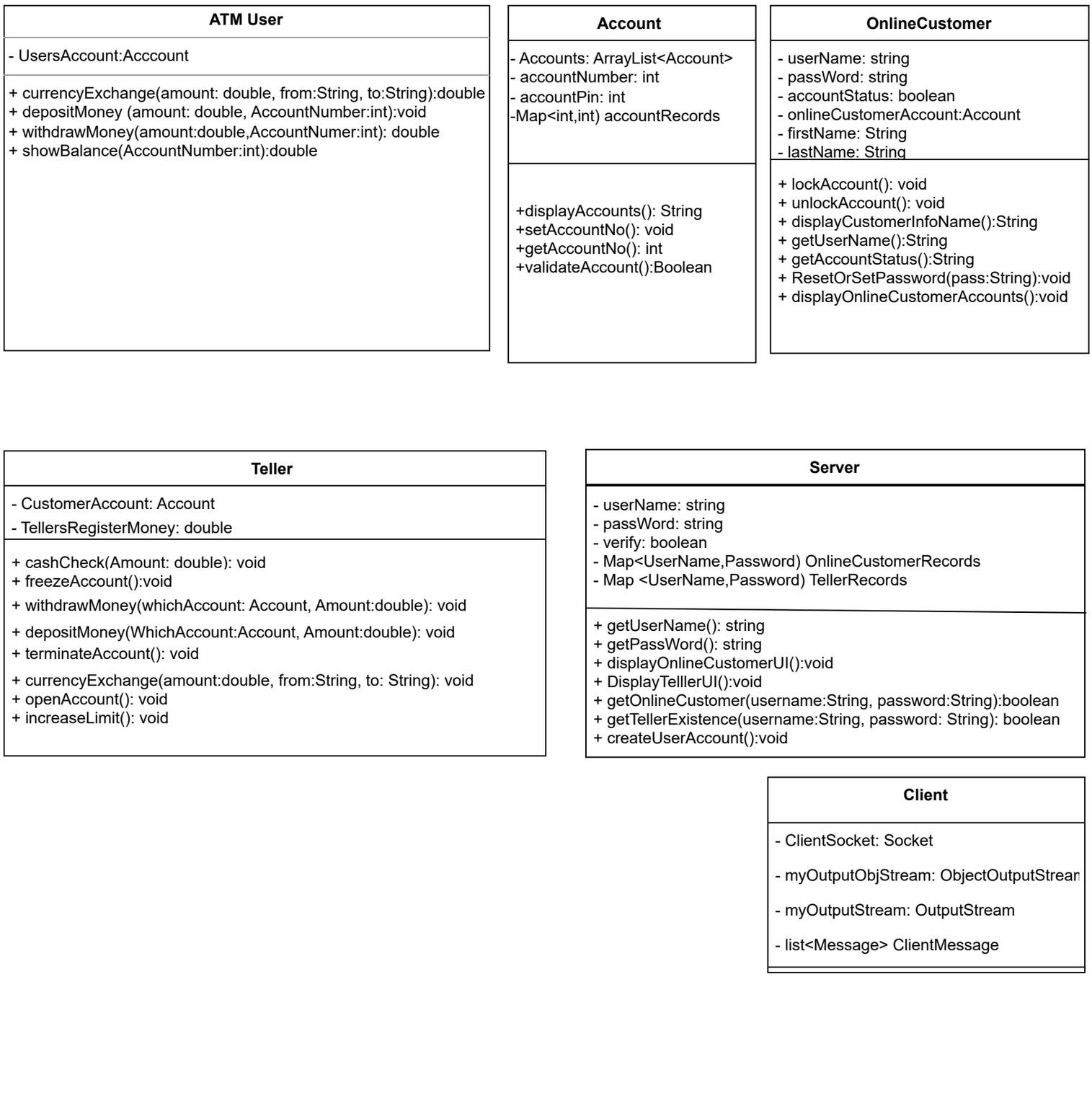



Client


Account
+ accountNumber: Int
+ accountBalance: Int
+ method(): Type
Checkings
+ field: Type
+ method(): Type
Savings
+ field: Type
+ method(): Type
Credit Card
+ creditCardNumber: Int
+ statementBalance: Double
+ paymentDate: Int
+ viewStatement(): txtFile
+ makePayment(): void
+ viewLimit(): int
+ viewTransactionHistory(): txtfile


+ socket?
+ sendRequest?
+ printTransactionHistory?
web user
+ userType: string
+ userLogin: string
+ userPassword: string
+ login()
+ createAccount()
Client
+ socket?
+ sendRequest?
+ printTransactionHistory?
Teller
+ field: Type
+ customercust
+ openAccount(): void
+ terminateAccount(): void
+ makeDeposit(): void
+ makeWithdrawl(): void
+ freezeCustomerAccount(): void
+ isCustomer(ID:INT): bool
Customer
+ pin: Int
+ nameFirst: string
+ nameLast: String
+ accounts[] myAccount
+ Map<ID, PIN>
+ makeDeposit(amount:double): void
+ makeWithdrawl(account:Account, amount:double): void
+ makeTransfer(accountFrom, accountTo, amount)
+Generate_customer_ID()
+ AccountNumber: i
+ pinNumber: int
+ currencyExchange
+ depositMoney
+ withdrawMoney(a
+ showBalance():do


Server
+ field: Type
+ method(): Type
ATM
nt
e(amount: double, from:String, to:String):double
mount:double): double
uble


ATM User
- UsersAccount:Acccount
+ currencyExchange(amount: double, from:String, to:String):double
+ depositMoney (amount: double, AccountNumber:int):void
+ withdrawMoney(amount:double,AccountNumer:int): double
+ showBalance(AccountNumber:int):double
Account
- Accounts: ArrayList<Account>
- accountNumber: int
- accountPin: int
-Map<int,int) accountRecords
+displayAccounts(): String
+setAccountNo(): void
+getAccountNo(): int
+validateAccount():Boolean
OnlineCustomer
- userName: string
- passWord: string
- accountStatus: boolean
- onlineCustomerAccount:Account
- firstName: String
- lastName: String
+ lockAccount(): void
+ unlockAccount(): void
+ displayCustomerInfoName():String
+ getUserName():String
+ getAccountStatus():String
+ ResetOrSetPassword(pass:String):void
+ displayOnlineCustomerAccounts():void
Teller
- CustomerAccount: Account
- TellersRegisterMoney: double
+ cashCheck(Amount: double): void
+ freezeAccount():void
+ withdrawMoney(whichAccount: Account, Amount:double): void
+ depositMoney(WhichAccount:Account, Amount:double): void
+ terminateAccount(): void
+ currencyExchange(amount:double, from:String, to: String): void
+ openAccount(): void
+ increaseLimit(): void
Server
- userName: string
- passWord: string
- verify: boolean
- Map<UserName,Password) OnlineCustomerRecords
- Map <UserName,Password) TellerRecords
+ getUserName(): string
+ getPassWord(): string
+ displayOnlineCustomerUI():void
+ DisplayTelllerUI():void
+ getOnlineCustomer(username:String, password:String):boolean
+ getTellerExistence(username:String, password: String): boolean
+ createUserAccount():void
Client
- ClientSocket: Socket
- myOutputObjStream: ObjectOutputStrean
- myOutputStream: OutputStream
- list<Message> ClientMessage