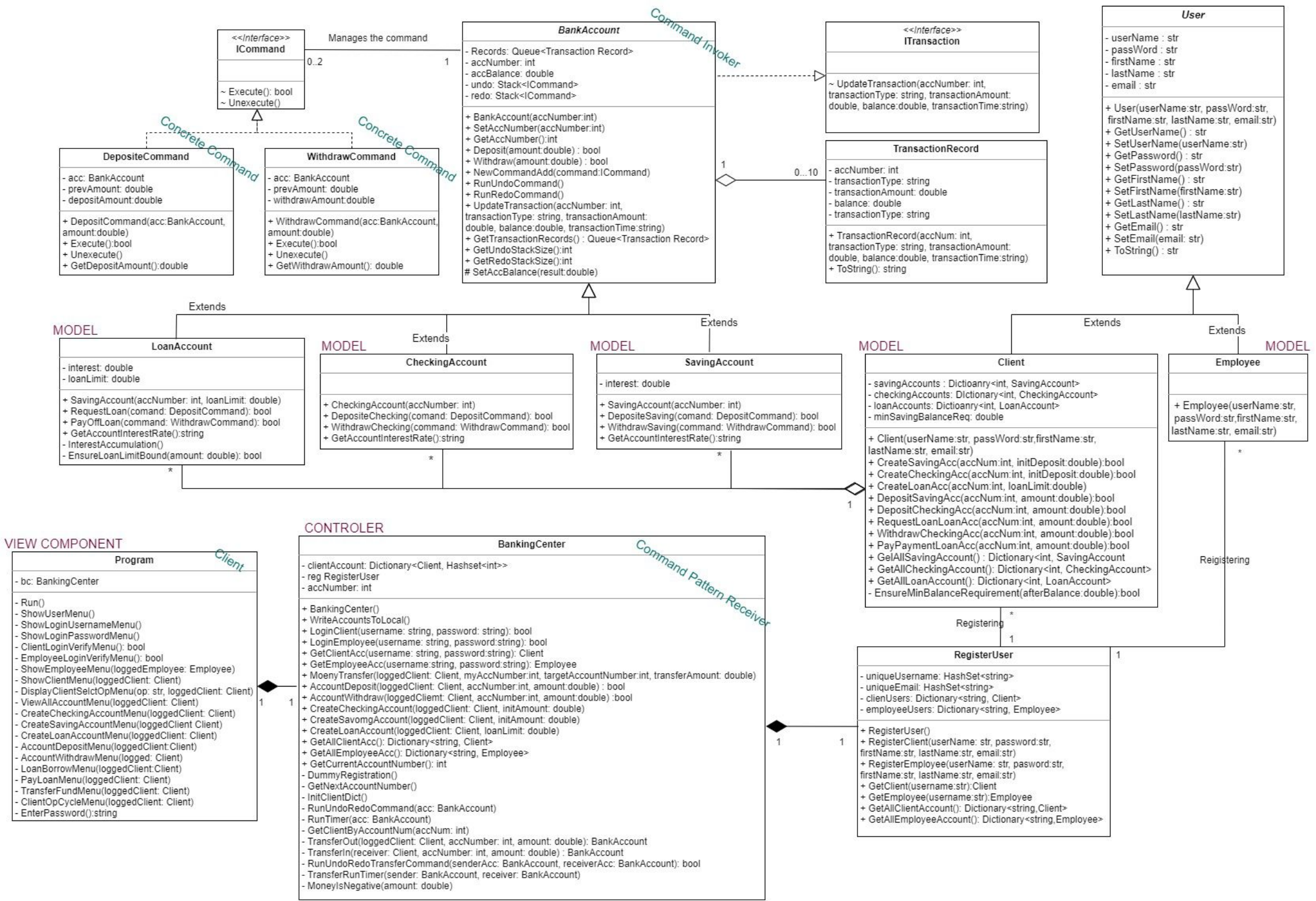
ICommand
<<Interface>>
~ Execute(): bool
~ Unexecute()
Manages the command
0..2
1
BankAccount
Command Invoker
- Records: Queue<Transaction Record>
- accNumber: int
- accBalance: double
- undo: Stack<ICommand>
- redo: Stack<ICommand>
+ BankAccount(accNumber:int)
+ SetAccNumber(accNumber:int)
+ GetAccNumber():int
+ Deposit(amount:double) : bool
+ Withdraw(amount:double) : bool
+ NewCommandAdd(command:ICommand)
+ RunUndoCommand()
+ RunRedoCommand()
+ UpdateTransaction(accNumber: int, transactionType: string, transactionAmount: double, balance:double, transactionTime:string)
+ GetTransactionRecords() : Queue<Transaction Record>
+ GetUndoStackSize():int
+ GetRedoStackSize():int
# SetAccBalance(result:double)
ITransaction
<<Interface>>
~ UpdateTransaction(accNumber: int, transactionType: string, transactionAmount: double, balance:double, transactionTime:string)
User
- userName : str
- passWord : str
- firstName : str
- lastName : str
- email : str
+ User(userName:str, passWord:str, firstName:str, lastName:str, email:str)
+ GetUserName() : str
+ SetUserName(userName:str)
+ GetPassword() : str
+ SetPassword(passWord:str)
+ GetFirstName() : str
+ SetFirstName(firstName:str)
+ GetLastName() : str
+ SetLastName(lastName:str)
+ GetEmail() : str
+ SetEmail(email: str)
+ ToString() : str
Concrete Command
DepositeCommand
- acc: BankAccount
- prevAmount: double
- depositAmount:double
+ DepositCommand(acc:BankAccount, amount:double)
+ Execute():bool
+ Unexecute()
+ GetDepositAmount():double
Concrete Command
WithdrawCommand
- acc: BankAccount
- prevAmount: double
- withdrawAmount:double
+ WithdrawCommand(acc:BankAccount, amount:double)
+ Execute():bool
+ Unexecute()
+ GetWithdrawAmount(): double
1
0...10
TransactionRecord
- accNumber: int
- transactionType: string
- transactionAmount: double
- balance: double
- transactionType: string
+ TransactionRecord(accNum: int, transactionType: string, transactionAmount: double, balance:double, transactionTime:string)
+ ToString(): string
Extends
MODEL
LoanAccount
- interest: double
- loanLimit: double
+ SavingAccount(accNumber: int, loanLimit: double)
+ RequestLoan(comand: DepositCommand): bool
+ PayOffLoan(command: WithdrawCommand): bool
+ GetAccountInterestRate():string
- InterestAccumulation()
- EnsureLoanLimitBound(amount: double): bool
Extends
MODEL
CheckingAccount
+ CheckingAccount(accNumber: int)
+ DepositeChecking(comand: DepositCommand): bool
+ WithdrawChecking(command: WithdrawCommand): bool
+ GetAccountInterestRate():string
Extends
MODEL
SavingAccount
- interest: double
+ SavingAccount(accNumber: int)
+ DepositeSaving(comand: DepositCommand): bool
+ WithdrawSaving(command: WithdrawCommand): bool
+ GetAccountInterestRate():string
Extends
Extends
MODEL
Client
- savingAccounts : Dictioanry<int, SavingAccount>
- checkingAccounts: DIctionary<int, CheckingAccount>
- loanAccounts: Dictioanry<int, LoanAccount>
- minSavingBalanceReq: double
+ Client(userName:str, passWord:str,firstName:str, lastName:str, email:str)
+ CreateSavingAcc(accNum:int, initDeposit:double):bool
+ CreateCheckingAcc(accNum:int, initDeposit:double):bool
+ CreateLoanAcc(accNum:int, loanLimit:double)
+ DepositSavingAcc(accNum:int, amount:double):bool
+ DepositCheckingAcc(accNum:int, amount:double):bool
+ RequestLoanLoanAcc(accNum:int, amount:double):bool
+ WithdrawCheckingAcc(accNum:int, amount:double):bool
+ PayPaymentLoanAcc(accNum:int, amount:double):bool
+ GelAllSavingAccount() : Dictionary<int, SavingAccount
+ GetAllCheckingAccount(): Dictionary<int, CheckingAccount>
+ GetAllLoanAccount(): Dictionary<int, LoanAccount>
- EnsureMinBalanceRequirement(afterBalance:double):bool
MODEL
Employee
+ Employee(userName:str, passWord:str,firstName:str, lastName:str, email:str)
Reigistering
Registering
RegisterUser
- uniqueUsername: HashSet<string>
- uniqueEmail: HashSet<string>
- clienUsers: Dictionary<string, Client>
- employeeUsers: Dictionary<string, Employee>
+ RegisterUser()
+ RegisterClient(userName: str, password:str, firstName:str, lastName:str, email:str)
+ RegisterEmployee(userName: str, pasword:str, firstName:str, lastName:str, email:str)
+ GetClient(username:str):Client
+ GetEmployee(username:str):Employee
+ GetAllClientAccount(): Dictionary<string,Client>
+ GetAllEmployeeAccount(): Dictionary<string,Employee>
VIEW COMPONENT
Client
Program
- bc: BankingCenter
- Run()
- ShowUserMenu()
- ShowLoginUsernameMenu()
- ShowLoginPasswordMenu()
- ClientLoginVerifyMenu(): bool
- EmployeeLoginVerifyMenu(): bool
- ShowEmployeeMenu(loggedEmployee: Employee)
- ShowClientMenu(loggedClient: Client)
- DisplayClientSelctOpMenu(op: str, loggedClient: Client)
- ViewAllAccountMenu(loggedClient: Client)
- CreateCheckingAccountMenu(loggedClient: Client)
- CreateSavingAccountMenu(loggedClient Client)
- CreateLoanAccountMenu(loggedClient: Client)
- AccountDepositMenu(loggedClient:Client)
- AccountWithdrawMenu(logged: Client)
- LoanBorrowMenu(loggedClient:Client)
- PayLoanMenu(loggedClient: Client)
- TransferFundMenu(loggedClient: Client)
- ClientOpCycleMenu(loggedClient: Client)
- EnterPassword():string
CONTROLER
Command Pattern Receiver
BankingCenter
- clientAccount: Dictionary<Client, Hashset<int>>
- reg RegisterUser
- accNumber: int
+ BankingCenter()
+ WriteAccountsToLocal()
+ LoginClient(username: string, password: string): bool
+ LoginEmployee(username: string, password:string): bool
+ GetClientAcc(username: string, password:string): Client
+ GetEmployeeAcc(username:string, password:string): Employee
+ MoenyTransfer(loggedClient: Client, myAccNumber:int, targetAccountNumber:int, transferAmount: double)
+ AccountDeposit(loggedClient: Client, accNumber:int, amount:double) : bool
+ AccountWithdraw(loggedCliemt: Client, accNumber:int, amount:double) :bool
+ CreateCheckingAccount(loggedClient: Client, initAmount: double)
+ CreateSavomgAccount(loggedClient: Client, initAmount: double)
+ CreateLoanAccount(loggedClient: Client, loanLimit: double)
+ GetAllClientAcc(): Dictionary<string, Client>
+ GetAllEmployeeAcc(): Dictionary<string, Employee>
+ GetCurrentAccountNumber(): int
- DummyRegistration()
- GetNextAccountNumber()
- InitClientDict()
- RunUndoRedoCommand(acc: BankAccount)
- RunTimer(acc: BankAccount)
- GetClientByAccountNum(accNum: int)
- TransferOut(loggedClient: Client, accNumber: int, amount: double): BankAccount
- TransferIn(receiver: Client, accNumber: int, amount: double) : BankAccount
- RunUndoRedoTransferCommand(senderAcc: BankAccount, receiverAcc: BankAccount): bool
- TransferRunTimer(sender: BankAccount, receiver: BankAccount)
- MoneyIsNegative(amount: double)