PROJE 2:

Teslim Tarihi: 15 Ağustos 2018, Saat 17:00

GRADING

Items	Grade (%)
Internal Documentation	20
Code quality (well formatted)	20
Functionality (implemented and correctly running functionalities)	60

Project Title: Banking System

Project Overview:

Develop a polymorphic banking program using the UML Class Diagram given below. There are two polymorphic class hierarchy with the abstract base classes Account and Customer.

Definitions:

- **Account Class**
  - **Member data**
    - balance: data member that stores the balance
    - accountNumber: Account Number which is unique for each account.
    - owner: points the owner of the account.
  - **Member Functions**
    - credit(double amount): a virtual function which adds an amount to the account balance
    - debit(double amount): a virtual function which subtracts an amount from the balance.
    - print(): a virtual function which prints the attributes of the acoount.
    - endOfDay(): a pure virtual function which adds/subtracts an interest to/from the balance.
    - setAccountOwner(): assigns the given customer pointer to owner.
    - getAccountType(): a pure virtual function which returns the account type (e.g. “CheckingAccount” or “SavingsAccount”)
- **CheckingAccount Class**

- **Member Data**
  - transactionFee: fee charged per transaction
  - creditInterestRate: daily interest rate for the credited amount.
- **Member Functions**
  - chargeFee(): a private utility function to charge fee, which subtracts the transaction fee from balance
  - credit(double amount): credit (add) an amount to the account balance and subtract the charge fee.
  - debit(double amount): subtracts an amount from the balance.If the balance is negative after subtraction , then subtract the transaction fee from the balance.
  - print(): prints the attributes of the account.
  - endOfDay(): subtracts an interest from the balance.
  - getAccountType():returns "CheckingAccount"

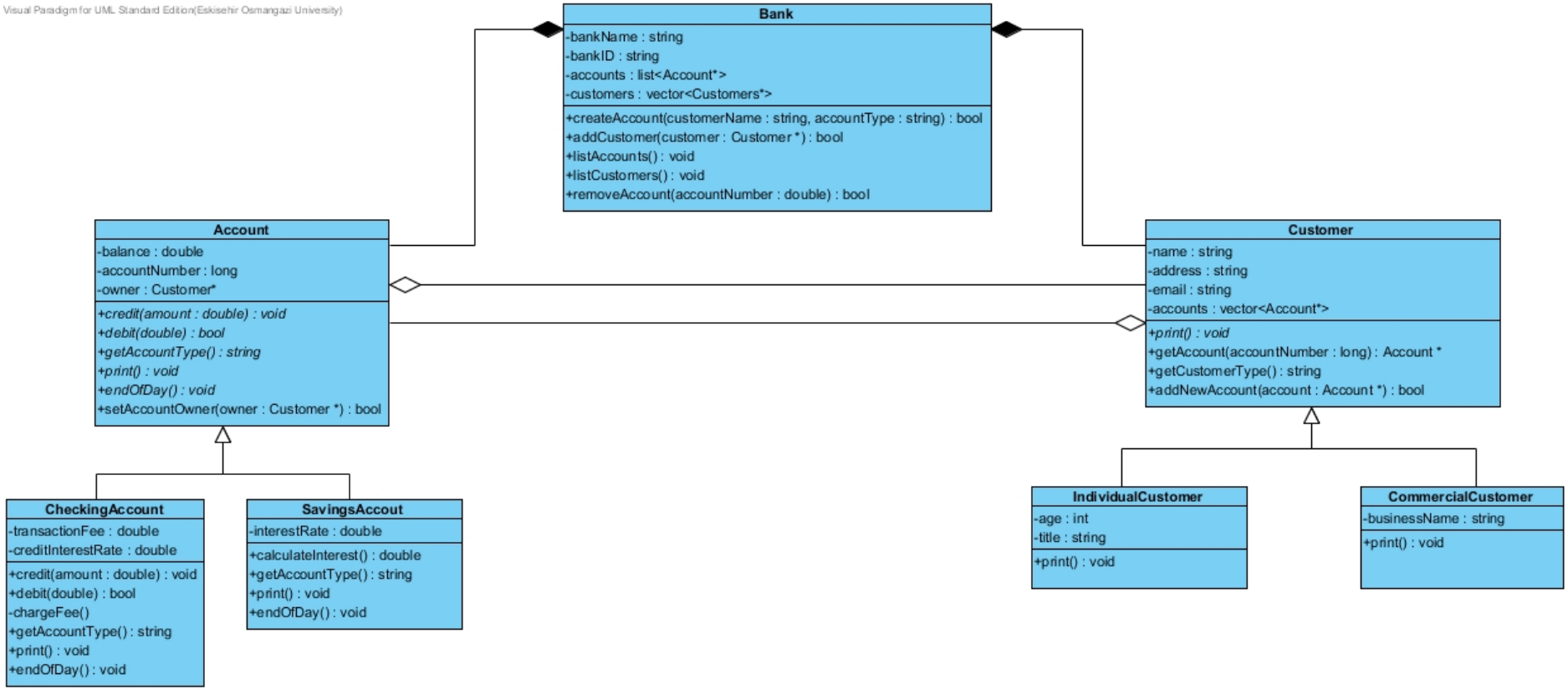
- **SavingAccount Class**
  - **Member Data**
    - interestRate: daily interest rate for the balanced amount.
  - **Member Functions**
    - debit(double amount): subtracts an amount from the balance if the balance is bigger than zero after subtraction.
    - print(): prints the attributes of the account.
    - endOfDay(): adds an interest to the balance.
    - getAccountType():returns "SavingAccount"
- **Customer Class**
  - **Member Data**
    - name: the name of customer
    - address: the address of the customer
    - email: the email of the customer
    - accounts: includes the pointers for the accounts of the customer.
  - **Member Functions**

- print(): a pure virtual function which prints the information of the customer and his accounts
- getAccount(): compares the given account number with the customer's accounts, if the given number is the customer' account returns a pointer to its account.
- getCustomerType():returns the customer type (e.g. "IndivudualCustomer" or "CommercialCustomer")

- **Bank Class**
  - **Member Data**
    - bankName: the name of the bank
    - bankID: the ID of the bank
    - accounts: includes the accounts in the bank.
    - customers: includes the customers in the bank.
  - **Member Functions**
    - createAccount(): creates an account for the customer with the given name. If there is no customer with the given name, does not create any account
    - addCustomer(): adds a new customer.
    - listAccounts(): prints the list of whole account s in the bank.
    - listCustomers(): prints the list of whole customers in the bank.
    - removeAccount(): removes the account with the given number. If there is no accounts with the given number, does not remove anything.
    - addNewAccount: adds the given account to customer account list.

Note that you should add set/get functions, constructors, destructors, and also another functions needed while coding.

As a result, you should write an application to test your classes. Your application should include at least 3 individual customers, 3 commercial customers, 3 checking accounts and 3 saving accounts.


Visual Paradigm for UML Standard Edition(Eskisehir Osmangazi University)
Bank
-bankName : string
-bankID : string
-accounts : list<Account*>
-customers : vector<Customers*>
+createAccount(customerName : string, accountType : string) : bool
+addCustomer(customer : Customer *) : bool
+listAccounts() : void
+listCustomers() : void
+removeAccount(accountNumber : double) : bool
Account
-balance : double
-accountNumber : long
-owner : Customer*
+credit(amount : double) : void
+debit(double) : bool
+getAccountType() : string
+print() : void
+endOfDay() : void
+setAccountOwner(owner : Customer *) : bool
Customer
-name : string
-address : string
-email : string
-accounts : vector<Account*>
+print() : void
+getAccount(accountNumber : long) : Account *
+getCustomerType() : string
+addNewAccount(account : Account *) : bool
CheckingAccount
-transactionFee : double
-creditInterestRate : double
+credit(amount : double) : void
+debit(double) : bool
-chargeFee()
+getAccountType() : string
+print() : void
+endOfDay() : void
SavingsAccout
-interestRate : double
+calculateInterest() : double
+getAccountType() : string
+print() : void
+endOfDay() : void
IndividualCustomer
-age : int
-title : string
+print() : void
CommercialCustomer
-businessName : string
+print() : void