# Final Project
# Banking System

**Program:**

*Course Code: CSE 323*

*Course Name: Software Formal Specifications*

*Examination Committee*

**Dr. Islam El-Maddah**

**Ain Shams University**
**Faculty of Engineering**
**International Credit Hours Engineering Programs (I-CHEP)**

## Student Personal Information for Group Work

| Student Names: | Student Codes: |
|---|---|
| Gina Emil Attia | 16p3003 |
| Rowan Hazem Wagieh | 16P3023 |

## Plagiarism Statement

**Signature/Student Name:** **Gina Emil, Rowan Hazem** **Date:** **7/6/2020**

## Submission Contents

**01:** **Introduction**

**02:** **The reasons to develop this system formally and Problems that could happen in case of usual developments**

**03:** **Set of informal requirements**

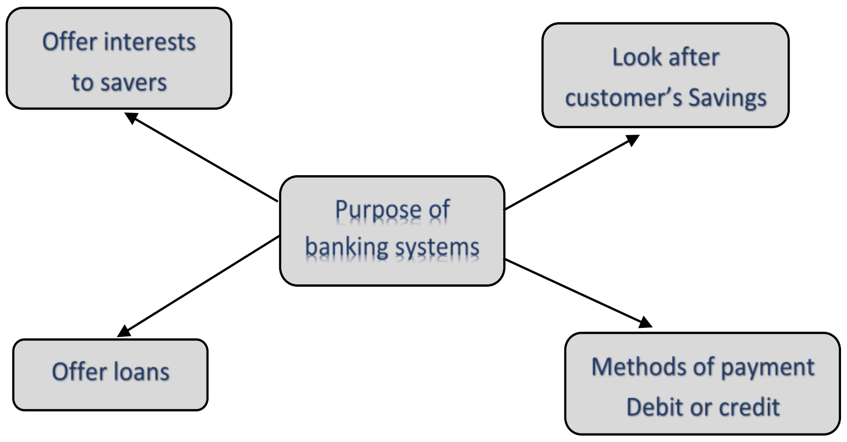
**04:** **Formal specification and restriction**

**05:** **Run Screen**

# 01 Introduction

A group or network of institutions is known as banking system that plays a significant role in the economy for providing financial services for customers. Such institutions are responsible for managing a payment system, giving loans, taking deposits, and to help with investments.

Depending on the network of institutions, the bank executes various functions. Payment and loans allow us to deposit funds, for example. Also, using the checking accounts and debit cards to buy things and pay our bills. They even play a significant role in supplying funding to those businesses who wish to invest and grow.


Offer interests to savers
Look after customer's Savings
Purpose of banking systems
Offer loans
Methods of payment Debit or credit

## Description of our system

A Banking system which has multiple branches and it can open a new one or close an already existing one, where each branch contains a set of different customers' accounts. Each customer must have a set of requirements to open a new account and chooses its type, then through this account he can do multiple of things such as; deposit or withdraw money, transfer money to another accounts, have an interest rate , see his debit and credit balances, and he can also open another account or request to close an existing one. The Banking system also provides a service in which a customer can take a loan whether he has an account or not, also he can take more than one loan but under specific conditions.

# 02 The reasons to develop this system formally and Problems that could happen in case of usual developments

## The reasons to develop this system formally

- **Increasingly, all systems depend on software components**
  - Fault protection and safety are no longer allocated solely to hardware.
  - Software must be capable of identifying and isolating faults, and then executing recovery scenarios.
  - Computer systems fail in different ways than hardware.
- **The complexity of embedded software systems has rapidly increased**
- **Developing a formal specification provides insights and understanding of software and software design requirements.**
- **Because of the formal system specification and the complete formal programming language definition, it may be possible to show that the program meets its specifications.**
- **The formal specifications can be generated automatically. Software tools can be built to assist the developers develop, understand, and debug.**

## Problems that could happen in case of usual developments:

- **Contradictions**
  - The declarations do not agree with each other.
- **Ambiguity**
  - Words are read in more than one way.
- **Vagueness**
  - The specifications are still not written clearly enough in large documents.
  - Statements written down in lower details.
- **Incompleteness**
  - Failure to list the limitations and manage errors needed for a function.
- **Abstraction mixed levels**
  - It happens when very complex statements are accidentally mixed with statements written down in lower details.

# 03 Set of informal requirements

- All account balances must be greater than zero, accounts can't have negative balance.

- No two customers can have the same ID.

- No two accounts can have the same account number.

- When a customer wants to withdraw an amount of money from his account, first his account balance must be greater than the amount needed to be withdrawn, second his account balance must be updated after the withdraw operation.

- When a customer wants to deposit an amount of money to his account, his account balance must be updated by adding the new amount to the old one after the deposit operation.

- A customer's age must be at least 21 years and his balance must be 10,000 EGP or more so he could open an account, also we check on the accounts to give him a unique account number.

- To delete an account, we have to check first whether it is an existing one or not so it can be deleted.

- To take a loan, the customer's age must be 21 years or more, and his prepayment amount must be at least 10% of the loan's amount.

- To remove a loan from the system, we must check that the customer's prepayment amount is equal to the loan amount and the loan already exists so it can be deleted.

- To add a new branch, we have to give it a unique code.

- To delete a branch, we have to check first whether it is an existing one or not so it can be deleted.

- If customer A wants to transfer an amount of money to customer B, then the amount must be at least equal or greater than customer A's balance. Then after transferring Customer A's and customer B's balances must be updated with the new amounts.

- Class Bank must have a name and bank _code, and it can add a branch or remove a branch owned by it.

- Class Branch must have a code, city and location, and it can add a new account or remove an existing one, also it can add a Loan or remove a loan from the system.

- Class customer must have a name, ID, address, mobile number and age

- Class Account must have a number, customer's balance, account fees, account type, date of opening the account and interest rate. It can show the amount of the debit and the credit, also it's responsible for the withdraw, deposit and transfer operations requested by the customer.

- Class Loan must have a number, loan amount, type and a prepayment amount, and it saves the customer's information.

## 04 Formal specification and restriction

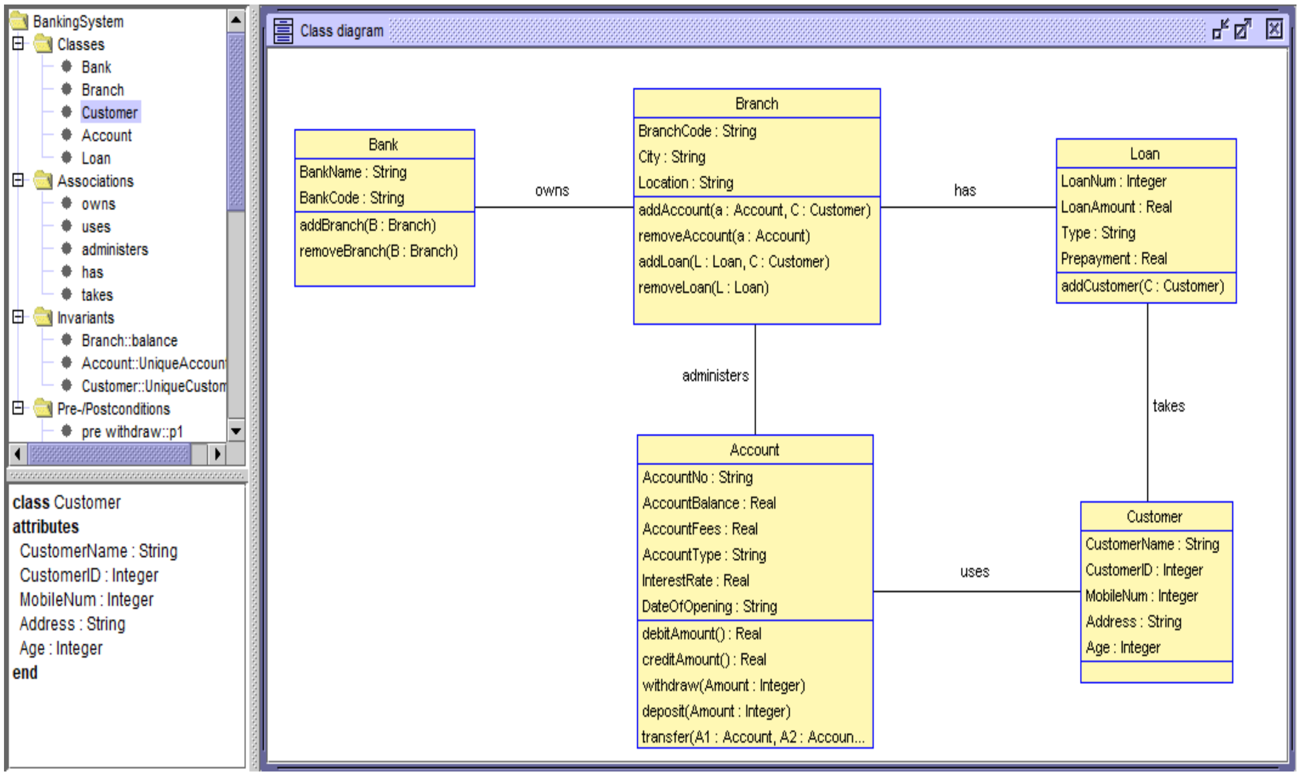
```
model BankingSystem
--classes
class Bank
attributes
BankName : String
BankCode : String

operations
addBranch(B: Branch)
removeBranch(B: Branch)
end

class Branch
attributes
BranchCode: String
City: String
Location: String

operations
addAccount(a: Account, C: Customer)
removeAccount(a: Account)
```

```
addLoan(L: Loan, C: Customer)
removeLoan(L: Loan)
end

class Customer
attributes
CustomerName : String
CustomerID : Integer
MobileNum: Integer
Address: String
Age: Integer
end

class Account
attributes
AccountNo : String
AccountBalance : Real
AccountFees : Real
AccountType : String
InterestRate: Real
DateOfOpening : String

operations
debitAmount(): Real
creditAmount(): Real
withdraw(Amount: Integer)
deposit(Amount: Integer)
transfer(A1:Account, A2: Account, amount: Integer)
end

class Loan
attributes
LoanNum: Integer
LoanAmount: Real
Type: String
Prepayment: Real

operations
addCustomer(C: Customer)
end

-- associations
association owns between
  Bank[1]
  Branch[1..*]
end

association uses between
        Customer[*]
        Account[1]
end


association administers between
  Account[0..*]
  Branch[1]
end

association has between
  Loan[0..*]
  Branch[1]
end

association takes between
  Loan[*]
  Customer[1]
end

--invariants
constraints
context Branch
inv balance: Account.allInstances.forAll(a:Account | a.AccountBalance > 0)

context Account
inv UniqueAccount: Account.allInstances.forAll(A1 , A2: Account |( (A1.AccountNo <>
A2.AccountNo) implies (A1 <> A2)))

context Customer
inv UniqueCustomer: Customer.allInstances.forAll(C1, C2: Customer | (C1.CustomerID <>
C2.CustomerID) implies (C1<>C2))
context Account::withdraw(Amount:Integer)
pre p1: Amount > 0
pre p2: AccountBalance > Amount
post: AccountBalance = AccountBalance@pre - Amount

context Account::deposit(Amount: Integer)
pre: Amount>0
post: AccountBalance = AccountBalance@pre + Amount
```

```
context Branch::addAccount(a: Account, C: Customer)
pre p1: C.Age >= 21 and a.AccountBalance >= 10000
pre p2: Account.allInstances->excludes(a) --Account doesn't exist
post : Account.allInstances->includes(a) -- Add account

context Branch::removeAccount(a:Account)
pre p1: Account.allInstances->includes(a)
post: Account.allInstances->excludes(a) --Remove account

context Branch::addLoan(L: Loan, C: Customer)
pre p1: L.Prepayment >= 0.1*L.LoanAmount and C.Age >= 21
pre p2: Loan.allInstances->excludes(L) --Loan doesn't exist before
post: Loan.allInstances->includes(L) --Add Loan

context Branch::removeLoan(L: Loan)
pre p1: L.Prepayment = L.LoanAmount
pre p2: Loan.allInstances->includes(L)
post: Loan.allInstances->excludes(L) --Delete Loan

context Bank::addBranch(B:Branch)
pre : Branch.allInstances->excludes(B)
post: Branch.allInstances->includes(B)

context Bank::removeBranch(B:Branch)
pre: Branch.allInstances->includes(B)
post: Branch.allInstances->excludes(B) --remove Branch

context Account::transfer(A1: Account, A2: Account, amount: Integer)
pre : A1.AccountBalance >= amount
post Account1: A1.AccountBalance = A1.AccountBalance@pre - amount
post Account2: A2.AccountBalance = A2.AccountBalance@pre + amount
```

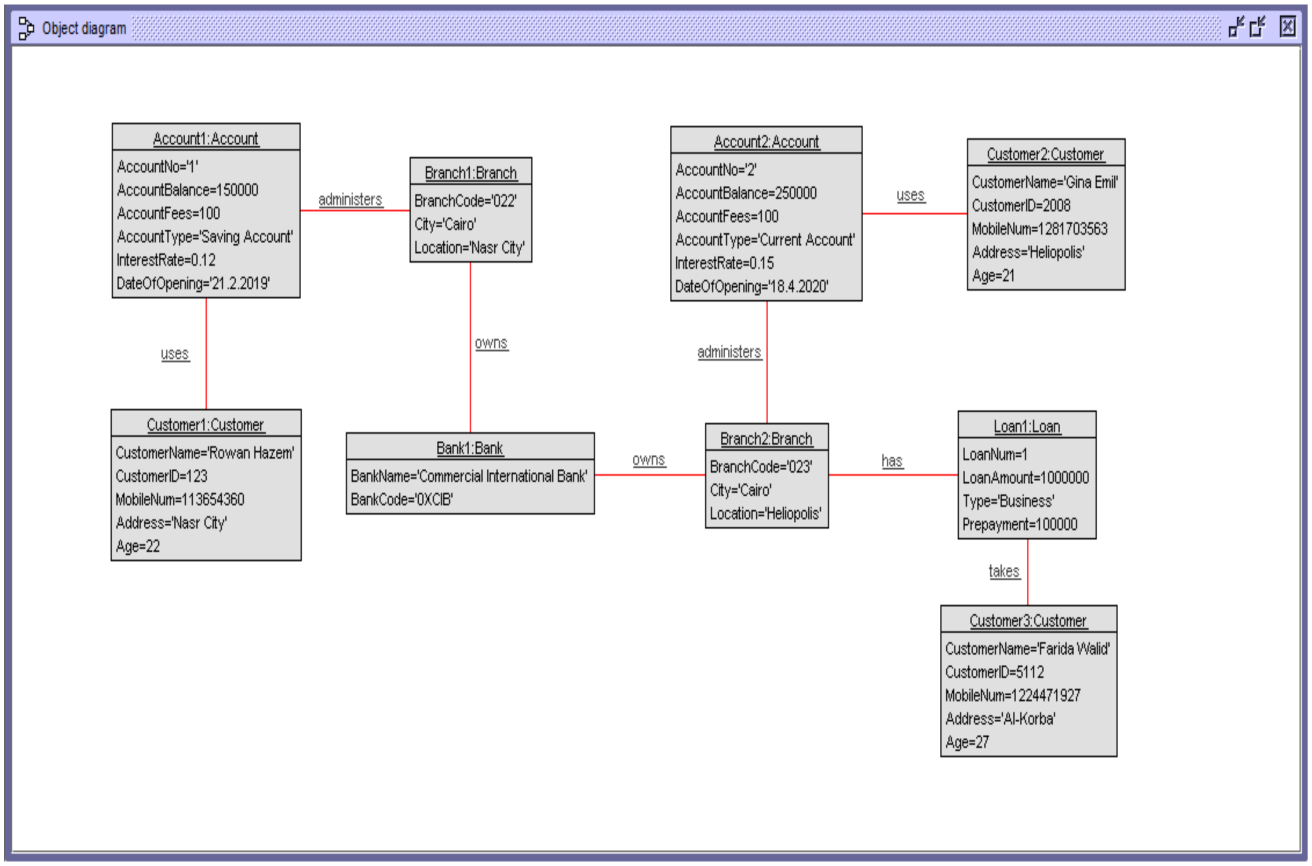
## 05 Run Screen

- **Class Diagram:**



```
class Customer
attributes
  CustomerName : String
  CustomerID : Integer
  MobileNum : Integer
  Address : String
  Age : Integer
end
```

- **Object Diagram:**


Object diagram
Account1:Account
AccountNo='1'
AccountBalance=150000
AccountFees=100
AccountType='Saving Account'
InterestRate=0.12
DateOfOpening='21.2.2019'
administers
Branch1:Branch
BranchCode='022'
City='Cairo'
Location='Nasr City'
Account2:Account
AccountNo='2'
AccountBalance=250000
AccountFees=100
AccountType='Current Account'
InterestRate=0.15
DateOfOpening='18.4.2020'
uses
Customer2:Customer
CustomerName='Gina Emil'
CustomerID=2008
MobileNum=1281703563
Address='Heliopolis'
Age=21
uses
owns
administers
Customer1:Customer
CustomerName='Rowan Hazem'
CustomerID=123
MobileNum=113654360
Address='Nasr City'
Age=22
Bank1:Bank
BankName='Commercial International Bank'
BankCode='0XCIB'
owns
Branch2:Branch
BranchCode='023'
City='Cairo'
Location='Heliopolis'
has
Loan1:Loan
LoanNum=1
LoanAmount=1000000
Type='Business'
Prepayment=100000
takes
Customer3:Customer
CustomerName='Farida Walid'
CustomerID=5112
MobileNum=1224471927
Address='Al-Korba'
Age=27