Queen Mary
University of London

ECS505U Software Engineering Coursework

04/12/2017

This coursework makes up 10% of the total marks for the module (the rest is made up of the labwork 10% and 80% for the summer exam). Answer all the questions to achieve a maximum score of 100 (Q1 has 20 marks, Q2 has 40 marks, Q3 has 20 marks and Q4 has 20 marks).

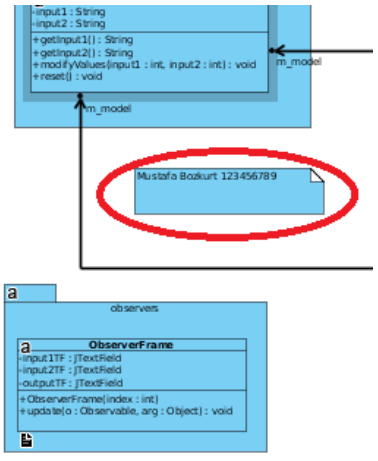
Submit your solution to the submission link on QMplus course webpage by Tuesday 12th December 09:00:00h. You should be able to see the details of the coursework submission on QMplus web page under assessed coursework page.

Please submit one PDF file for all questions with one diagram on each page. It's your responsibility to make sure the images are readable. We will not mark diagrams that are not readable. You will lose marks if you don't submit your coursework with the required format.

This coursework should take you about 12-15 hours to complete. **You are expected to complete it without collaborating with other students.**

It is recommended that you use Visual Paradigm UML tool to create diagrams as marks will be deducted if the tool you choose uses a non-standard UML notation.

Please add a note with your name and student ID on every diagram as depicted below in order to avoid any marking issues as we mark each diagram separately.



Overview

You are given the task of designing a software that handles day to day activities of a financial institution. Beware, the following requirements are a subset of an actual system. The requirements in this document are selected with the aim of enabling you to demonstrate your system design ability using UML. Thus, some of the requirements were omitted. However, you are not expected to improve the requirements of the software. Please, do not make assumptions and follow only the instructions.

If you have any questions regarding requirements, please ask it through the QMplus forum. I might not respond to questions via email.

Glossary:

The system: The software system designed to help staff with their day to day activities.
System User: A person who is using the system. Includes all types of users described below.
Staff: Employees who deals with day to day activities of the institution.
Customer: A person who holds an account with the institution. The system has two types of customers business and individual.
Individual customer (or individual): A customer with a personal account.
Business customer (or business): A customer with a business account.

Question 1 (20 marks)

Draw a use case diagram following the set of requirements below.

1. Staff must be able to search by an account by the following:
    a. Customer name
    b. Account number
2. Staff must be able to order a new card to an account (customer request).
3. Staff must be able to cancel a card.
4. Staff must be able to withdraw money from a given account.
5. Staff must be able to deposit money to a given account.
6. Staff must be able to display statement of a given account.
7. The system must ask the staff if he/she wants to specify a date range when displaying statement.
8. All customers must be able to deposit, withdraw and transfer money from their account.

HINTS:

- Specifying a date when displaying statement is optional.
- Staff must first search for an account before performing any account related activity. Search is required for **all account related activities** such as cancelling a card, depositing money and displaying statement.

You are expected use the one of the names below for each use case and actor in your use case diagram.
Staff, Customer, Business, Individual, Order New Card, Cancel Card, Withdraw from Account, Deposit to Account, Display Statement, Specify Date Range, Search Account, Search By Customer Name, Search By Account Number, Deposit, Withdraw, Transfer Money

Question 2 (40 marks)

Draw a class diagram using the following set of requirements.

1. There are two types of accounts: personal and business.
2. There are two types of personal accounts: checking and savings.
3. Each account is identified by a unique account number.
4. The system must be able to track the balance of each account.
5. All accounts can be deposited and withdrawn money.
6. Each account belongs to a branch (the branch where the account was opened).
7. Business and checking accounts have an overdraft allowance.
8. All accounts are kept in the registry.
9. System should be able to track bank cards.
10. Bank cards are identified by a unique number.
11. Bank cards has a name on the cards.
12. System must allow setting limits on bank cards.
13. All bank cards are kept in the registry.
14. Each bank card can be assigned to one account.
15. Checking accounts can be assigned one bank card.
16. Savings accounts cannot have assigned bank cards.
17. System should allow setting interest rates for savings accounts.
18. System should allow adding and removing cards to/from accounts.
19. Business accounts can be assigned up to ten bank cards.
20. The system must be able to add and remove cards to/from business and checking accounts.
21. There are two types of users in the system: staff and customer.
22. There are two types of customers: individual and business.
23. All individual customers must have a checking account.
24. A checking account is for one individual customer.
25. Individual customers can have one savings account.
26. Individual customers are not required to have a savings account.
27. Savings accounts can be shared by two individuals.
28. Businesses can have one business account.
29. Staff can have two types of contracts: full time and part time.
30. Full time staff can request paid vacations.
31. Part time staff can request unpaid vacations.
32. The system must be able track number of vacation days left for fulltime staff.
33. Branches have one or more staff.
34. Staff can work in one branch only.
35. System must keep first and last name of each user.
36. Each user is identified by a unique username.

HINTS:

- The requirements from question one are also part of the requirements of this question.
- Savings account can have up to two individual account holders.
- You might want your system to have a single registry object at any given time to make sure all records are kept in one object. Think of a design that ensures it.
- Assume that the certain staff can switch between contracts frequently. Think of a design that allows the employees switch between different contract types efficiently and fulfil requirements 29 to 32.
- Certain entities are represented as abstract classes and interfaces as a part of my design choice so draw these elements as specified in your diagrams.

You are expected to use the one of the names below for the class names.

Abstract Classes: User (both staff and customers are users), Customer, Account
Interfaces: Contract
Classes: Registry, PersonalAccount, BusinessAccount, CheckingAccount, SavingsAccount, BankCard, Branch, Staff, FullTime, PartTime, Business, Individual

Place following methods into the classes/interfaces in your diagram

- -instance : Registry
- -Registry()
- +getInstance() : Registry
- +findAccountByNumber(accountNumber : String) : Account (finds accounts in registry)
- -accountNumber : String
- -balance : float
- +withdraw(amount : float) : boolean
- +deposit(amount : float) : void
- -overdraftAllowance : float
- +getCards() : List<BankCard>
- +addCard(card : BankCard) : boolean
- +removeCard(card : BankCard) : void
- -overdraftAllowance : float
- +addCard(card : BankCard) : boolean
- +removeCard(card : BankCard) : void
- +getCard() : BankCard
- -interestRate : float
- -cardNumber : String
- -nameOnCard : String
- -limit : float
- +BankCard(number : String, nameOnCard : String)
- +getCardNumber() : String
- +getNameOnCard() : String
- +searchByAccountNumber(accountNumber : String) : Account
- +searchByName(name : String) : Account
- +orderNewCard(account : AccountI) : boolean
- +cancelCard(card : BankCard) : boolean
- +depositToAccount(account : Account, amount : float) : boolean
- +withdrawFromAccount(account : Account, amount : float) : boolean
- -vacationDaysUsed : int
- +requestPaidVacation(startDate : Calendar, endDate : Calendar) : void
- +requestUnpaidVacation(startDate : Calendar, endDate : Calendar) : void
- -firstName : String
- -lastName : String
- -userName : String
- +deposit(account : AccountI, amount : float) : boolean
- +withdraw(account : AccountI, amount : float) : boolean
- +transferMoney(accountNo : String, amount : float) : bolean

Question 3 (20 marks)

In this question, you are expected to draw a sequence diagram depicting the interactions of the following objects in removing a bank card from a business account. The following objects already available in the system for this scenario: user interface (use ":GUI" no object name is necessary), staff (Staff object), registry (Registry object), account (BusinessAccount object), card (BankCard object)

1. A staff makes a **search** for an **account** using GUI.
2. GUI calls the staff object for the search with the given number.
3. Staff object calls registry to find if an account with the given number exists and assigns the return value to a local variable result and returns result to GUI.
4. If result is null GUI **display**s **error** message.
5. If result is not null GUI **display**s **account** info to user.
6. If the account found (result not null), the staff makes a request system to **get bank cards** for this account.
7. GUI gets the list of cards from the account.
8. To display cards GUI gets the name on card and number for all business cards from the list.
9. GUI **display cards** to the staff.
10. Staff **select**s **card** to remove from the account using the GUI.
11. GUI calls the account to remove the selected card.
12. Account removes the card from the system.

Question 4 (20 marks)

In this question, you are expected draw a state chart diagram describing the possible states of checking accounts.

1. A new account added to the system is in good standing.
2. Money can be deposited and withdrawn from/to accounts in good standing.
3. Accounts in good standing become overdrawn when their balance fall below zero.
4. After 30 days, overdrawn accounts become delinquent.
5. As soon as account becomes delinquent the date is set (and the date is cleared on leaving).
6. Money can be withdrawn from overdrawn accounts.
7. Money cannot be withdrawn from delinquent accounts.
8. Below zero accounts are either overdrawn or delinquent.
9. Accounts below zero can be deposited money.
10. When the balance of a below zero account reaches zero or more it becomes a good standing account.
11. System should allow withdrawals only if the amount is less than or equal to the (account) balance plus the overdraft allowance.
12. Active accounts can be good standing or below zero.
13. Active accounts can be put on hold.
14. Accounts on hold are not allowed any activity (deposit or withdraw).
15. Accounts on hold become active when they are released.
16. Accounts on hold can be removed from the system.
17. Accounts in good standing can be closed.
18. An active account without any activity in 180 days becomes in active.
19. Inactive accounts are not allowed any activity.
20. Staff can activate inactive accounts.
21. Inactive accounts can be removed from the system.

HINTS:

- Withdraw amount should not be over the account balance plus overdraft allowance.
- We are assuming the customer pays any negative balance for activation and releasing of account so after these activities account goes back to a good standing state.

You are expected to use the one of the names below in your state chart diagram.
Active, GoodStanding, Overdrawn, Delinquent, BelowZero, OnHold, Inactive, closeAccount, putOnHold, releaseAccount, withdraw, deposit, balance, amount, overdraftAllowance, setDelinquentDate, clearDate, noActivityFor180days, activate, remove

End of paper