Queen Mary
University of London

ECS505U Software Engineering Coursework

14/11/2020

This coursework makes up 20% of the total marks for the module (the rest is made up of the lab work 20% and 60% for the January exam). Answer all the questions to achieve a maximum score of 100 (Q1 has 24 marks, Q2 has 55 marks and Q3 has 21 marks).

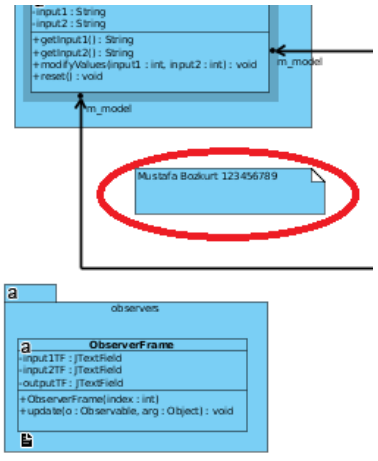
Submit your solution using the submission link on QMplus course webpage by Tuesday 1st December 9:00:00h. You should be able to see the details of the coursework submission on QMplus course page under assessed coursework section.

Please submit one PDF file for all questions with one diagram on each page. It is your responsibility to make sure the images are readable. We will not mark diagrams that are not readable. You will be penalized harshly if you do not submit your coursework with the required format: a three-page pdf file with each diagram in a separate page.

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 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. Staff might have two types of roles in a branch: cashier or service adviser.
Customer: A person who holds an account with the institution.
Individual customer (or individual): A customer with a personal account.
Business customer (or business): A customer with a business account.

Question 1 (24 marks)

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

1. Staff must be able to search for an account by the following:
    a. Customer name
    b. Account number
2. Staff must be able to order a new card for an account (customer in person request).
3. Staff must be able to cancel a card.
4. Staff must be able to display statement of a given account.
5. Cashiers must be able to withdraw money from a given account.
6. Cashiers must be able to deposit money to a given account.
7. Service advisors must be able to create mortgage accounts.
8. Customers must be able to deposit, withdraw and transfer money from their account.

HINTS:

- 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 or displaying a statement.
- Cashiers must search for an account before performing any account related activity.

You are expected use the one of the names below for each use case and actor in your use case diagram.
Staff, Customer, Cashier, Service Advisor, Order New Card, Cancel Card, Withdraw from Account, Deposit to Account, Display Statement, Search Account, Search by Customer Name, Search by Account Number, Deposit, Withdraw, Transfer Money, Create Mortgage Account

Question 2 (55 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 the name of the owner (might be different than the account holder name).
12. System must allow setting limits on bank cards.
13. All bank cards are kept in the registry.
14. Each bank card must be assigned to one account.
15. Checking accounts can be assigned one personal card.
16. Savings accounts cannot have assigned bank cards.
17. System should track interest rates for each savings account.
18. Business accounts can be assigned up to five business cards.
19. The system must be able to add and remove cards to/from business and checking accounts.
20. There are two types of users in the system: staff and customer.
21. There are two types of customers: individual and business.
22. All customers must be able to deposit, withdraw and transfer money.
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 roles in the branch: service advisor and cashier.
30. Cashiers can deposit or withdraw money to/from an account.
31. Service advisors can create mortgage accounts.
32. Service advisors can open new accounts.
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 in 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 staff switch between roles frequently. Think of a design that allows the staff switch between different contract types efficiently and fulfils 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. You can use them more than once.

Abstract Classes: User (both staff and customers are users), Customer, Account, BankCard
Interfaces: BranchRole
Classes: Registry, PersonalAccount, BusinessAccount, CheckingAccount, SavingsAccount, BusinessCard, PersonalCard, Branch, Staff, Cashier, ServiceAdvisor, 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)
- +findAccountByName(name : 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
- +createMorgageAccount(): boolean
- +openNewAccount() : Account
- -firstName : String
- -lastName : String
- -userName : String
- +deposit(account : Account, amount : float) : boolean
- +withdraw(account : Account, amount : float) : boolean
- +transferMoney(accountNo : String, amount : float) : boolean

Question 3 (21 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 (BusinessCard 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.

End of paper