BIT706 Programming III

Assignment 3

Weighting: 40%

Note: You must submit your work through the assignment section. Upload your assignment, then use the 'Save' and 'Submit assessment' buttons – or we will not receive your assignment. You will get an electronic receipt and your work is then logged and tracked through our system. Open Polytechnic will not accept assignments emailed directly to lecturers.

Overview

This assignment is worth 40% of your total course mark. This assignment focuses on the following learning outcomes:

LO1: Analyse business requirements and select an appropriate design, based on a contemporary object-oriented programming language

LO2: Design and develop a structured, modular and efficient prototype that meets the design criteria

LO3: Design and construct a graphical user interface (GUI) that meets user requirements

LO4: Execute debugging and testing techniques, according to a plan, and demonstrate correctness using final module testing

LO5: Implement source and version control processes, exemplifying best practice

LO6: Produce relevant internal and external documentation

Assessment instructions

In the previous two assignments you created a hierarchy of account classes with unit tests. You also created a simple customer class managed by a controller class and accessed by forms with a common branding. You will combine and enlarge on these two projects for this assignment.

Requirements

Along with the previous requirements for depositing and withdrawing money from different account types your project must now also enable a user to transfer money within their own accounts (but not between customers) and incorporate the requirement that bank staff are only charged 50% of any fees incurred (as in Assignment 1).

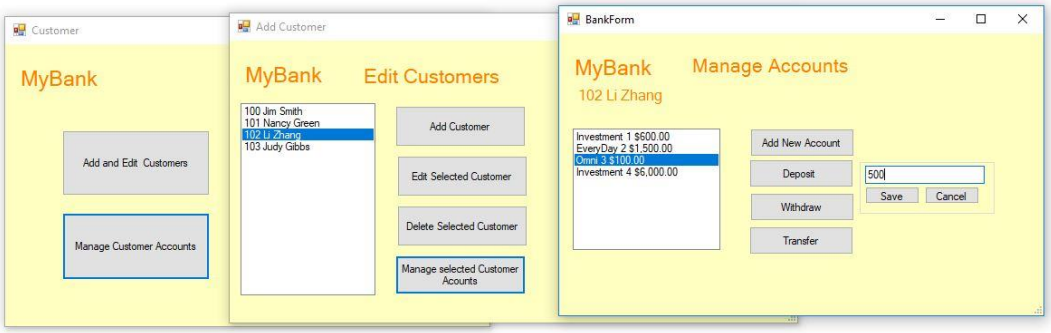
For this prototype the client has specified that the base fee for a failed transaction is $10 and the interest rate on all accounts is 4%. (You do not have to provide an interface for users to change these.)

The client still has the requirement that the code must easily support different user interfaces so you should use the model view controller pattern.

There will be many ways to develop a prototype to fulfil these requirements – you already have forms to add and edit a customer. Now you will add the facility to choose a customer and then:

- create new accounts for them
- deposit and withdraw funds
- transfer funds between accounts
- calculate and add interest
- write and read the customer and account data to and from a serialised binary file.

Below is one incomplete and unsophisticated way to design forms to do this. You may choose to design a different user interface. The middle form has an additional button that creates a new form for the selected customer and then has the opportunity to carry out the required account transactions.



What you are to do

Part A Analyse new requirements

You should create gherkin syntax features to represent all the different possible scenarios for transferring money between accounts. The scenarios should cover all the possibilities for different account types, balances, overdrafts and the different fees for bank staff. You will probably find it useful to use scenario outlines and templates.

(As your unit tests from Assignment 2 will already cover boundary conditions you do not need to provide gherkin scenarios for transfers that are on the boundaries of the balances. Nor do you need to include scenarios for withdrawals, deposits and interest calculations as these are also covered by your unit tests.)

(15%)

Part B Designing extensions to your existing classes

You should use the data model you prepared in Assignment 1. The relationship between customer and accounts is one-to-many so you will need to make small additions to the implementation of your customer and account classes to reflect that requirement. You will also need to distinguish customers who are bank staff.

Determine what additional methods the Controller class will need to meet the requirements of creating accounts and performing transactions on them.

Submit a report that shows the attributes and method signatures for the customer, account and controller classes.

(15%)

Part C Version control

You should start with the project containing your customer forms and controller from Assignment 2 and create a new branch. As you complete small parts of the code you should commit your code with appropriate comments. As you complete larger chunks you should merge your branch with master and create a new branch. Your final code should all be uploaded to your GitHub account. Make sure your repository is private and that your tutor is a collaborator.

(5%)

Part D Complete a prototype

It will make life easier for you if you follow the plan below:

- Import your account classes into the customer project for Assignment 2. (You may have to rename their namespace to match that of the customer project.)
- Design and create an inherited form or forms to manage a customer's accounts.
- Complete the controller and form code to make a working prototype.

(45%)

Part E Persist the data

Provide methods to write all the data to a serialised binary file and to read that data back into the project. These methods should run automatically when the prototype starts up or closes so that the data (including the nextID) is maintained.

(5%)

Part F Document your code

Add XML comments to your Controller class and generate a report.

(5%)

Part G Provide a short user guide for your prototype

This should include a brief overview of the functionality, and instructions for the main processes a user will carry out – this should not need to be more than 3 pages (including screenshots).

(5%)

Part H Module testing

Using your gherkin feature files as a test plan, carry out manual testing of the final prototype. Provide a table showing if each scenario has been satisfied or not. Where a scenario is not correct (and you haven't been able to fix the code) briefly describe what you think causes the error.

(5%)

Marking rubric

Part A Analyse new requirements (produce gherkin feature files) (15%)

(**Note:** Gherkin files act as both a method of specifying requirements and as a plan for acceptance testing.)

LO1: Analyse business requirements and select an appropriate design based on a contemporary object-oriented programming language.

LO4: Execute debugging and testing techniques, according to a plan, and demonstrate correctness using final module testing.

E < 40% 0-5.85	D 40%–49% 6-7.35	C C+ B- 50%–64% 7.5-9.6	B B+ A- 65%–79% 9.75-11.85	A A+ 80% ≤ 12-15
No real understanding of feature files and scenarios	Limited understanding but poorly and incompletely expressed	Most combinations of transfers between accounts and for different customer types are present	Most combinations are presented and the syntax is concise and understandable for automation	All combinations are presented and the syntax is suitable for automation

Part B Designing extensions to your existing classes (15%)

LO1: Analyse business requirements and select an appropriate design, based on a contemporary object-oriented programming language.

E < 40% 0-5.85	D 40%–49% 6-7.35	C C+ B- 50%–64% 7.5-9.6	B B+ A- 65%–79% 9.75-11.85	A A+ 80% ≤ 12-15
Little understanding of the purpose of a controller class Poor understanding of OO design for a one-to-many relationship	Shows limited understanding Some correct methods suggested	Shows a good understanding of what a controller class needs to do Lacks some methods and some signatures	Control class has suitable methods to manage objects and most requirements Most method signatures are	Control class has methods to manage objects and transactions Method signatures are all appropriate

		inappropriate One-to-many relationship properly designed Customer that is a staff member is handled appropriately	appropriate Design of one-to-many relationship is correct Customer that is a staff member is handled appropriately	One-to-many relationship is correctly designed Customer that is a staff member is handled appropriately

Part C Version control (5%)

LO5: Implement source and version control processes, exemplifying best practice.

E < 40% 0-1	D 40%–49% 2	C C+ B- 50%–64% 2.5-3	B B+ A- 65%–79% 4	A A+ 80% ≤ 5
No evidence of using version control tool	Evidence of some commits, little or no branching Correct upload to private Git repository	Some evidence of branching and regular commits	Good evidence of appropriate branching, commits and merges	All major features have their own branch Evidence of regular and appropriate commits and merge to master

Part D Complete a prototype (45%); **Part E** Persist data (5%) (50% total)

LO2: Design and develop a structured, modular and efficient prototype that meets the design criteria.

LO3: Design and construct a graphical user interface (GUI) that meets user requirements.

E < 40% 0-19.5	D 40%–49% 20-24.5	C C+ B- 50%–64% 25-32	B B+ A- 65%–79% 32.5-39.5	A A+ 80% ≤ 40-50
Many requirements missing Poor implementation of controller class Poor management of forms	Some requirements met Some requirements use controller class appropriately Forms have components to manage some requirements	Accounts can be made and attached to a customer Most transactions can be carried out correctly through the forms Most of the functionality is handled appropriately by the controller	Contains all functionality for previous grade. Plus**:** Methods exist for the serialisation of customer and account data Forms are well designed and are well managed (created, loaded, showed, and hidden appropriately)	Contains all functionality for previous grade. Plus: Control class manages the withdrawal exception appropriately NextID is persisted in an elegant manner and serialisation occurs automatically on start up and closing of the project Code is minimal and elegant

Part F XML comments and report (5%); **Part G** User guide (5%) (10% total)

LO6: Produce relevant internal and external documentation.

E < 40% 0-3.9	D 40%–49% 4-4.9	C C+ B- 50%–64% 5-6.4	B B+ A- 65%–79% 6.5-7.9	A A+ 80% ≤ 8-10
Few or no comments or report No (or very limited) user guide	Some appropriate XML comments in controller class Poor attempt at a user guide	Shows good understanding of purpose of XML comments but some missing Good attempt at user guide but some important functionally or instructions missing	Good understanding of XML comments and most important methods in controlled commented appropriately and report generated Good user guide, mostly complete	Contains all functionality for previous grade. Plus: Concise and informative comments and report generated Excellent user guide that covers all important functionality and is very well presented

Part H Module testing (5%)

(**Note:** Test plan is developed with gherkin syntax in Part A

LO4: Execute debugging and testing techniques, according to a plan, and demonstrate correctness using final module testing.

E < 40% 0-1	D 40%–49% 2	C C+ B- 50%–64% 2.5-3	B B+ A- 65%–79% 4	A A+ 80% ≤ 5
Poor understanding and poor implementation resulting in failing tests	Some understanding of the requirements but few tests pass	Some important tests missing and some not passing	Most tests present in feature files and most passing	All tests present in feature files and all passing