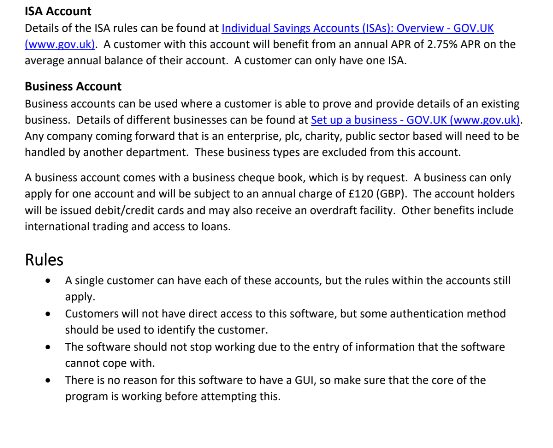
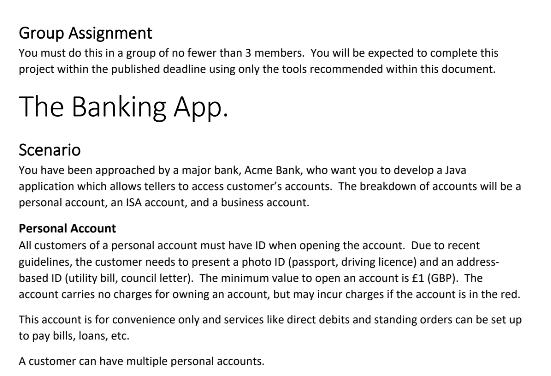
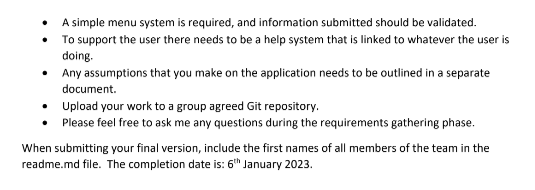
**Banking Application**

Text based Java Application that interfaces with an SQL database. Banking application that allows a bank teller to access and create accounts for a bank's clients. Handles three types of accounts for clients: Personal Accounts, ISA Accounts and Personal Business Accounts.

Developed by Ewan Hughes and Joshua Bhawanlall.

**Assignment Brief**



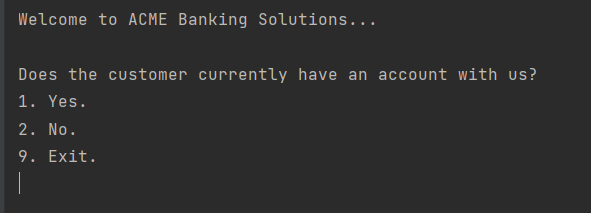


**Interface Overview**

We approached this task by creating a text based application that would function with menu systems at each stage for the bank teller.

At each stage the bank teller would be prompted to choose an option from the menu depending on the customer request.

Upon starting the application the program would ask the teller whether the bank client already had an account with the bank or not.

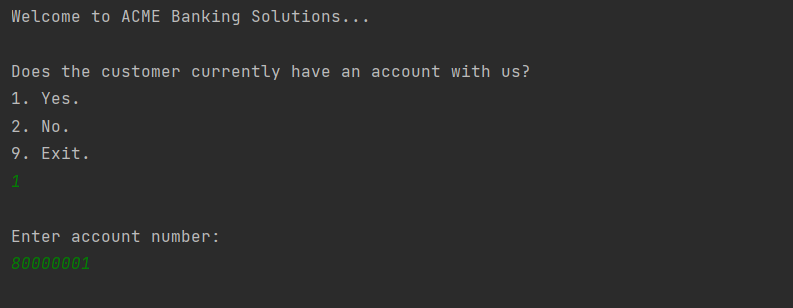


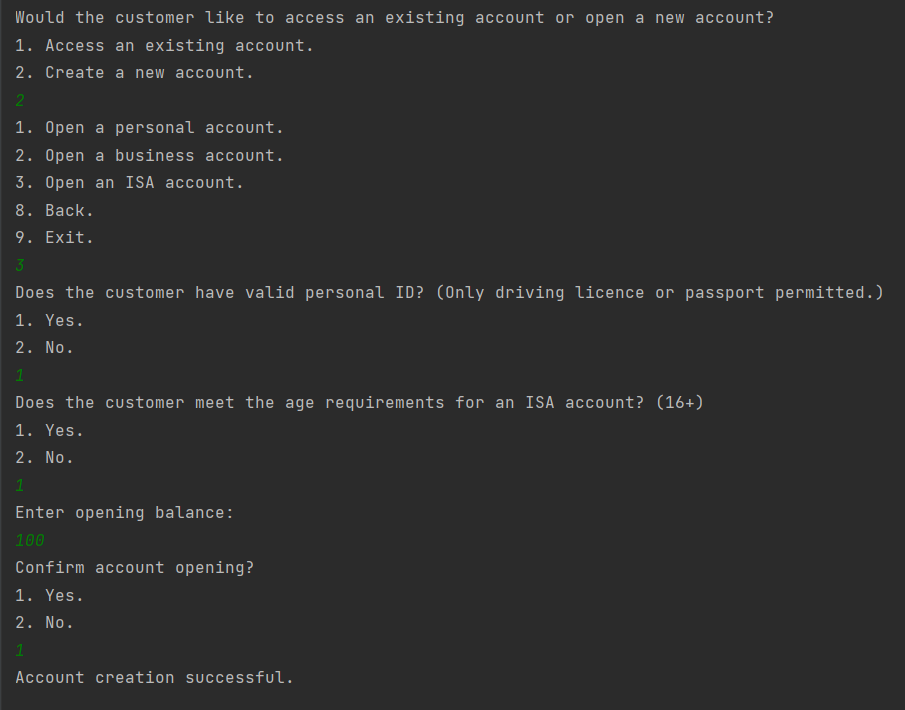
*Figure 1 - Application start menu.*

If the client had an account already the program would ask the teller to input one of the account numbers of an account that the client had.

This would prompt an input from the teller asking the client whether they would like to open a new account or access an existing account

Accessing an existing account would allow the program to fetch all of the users information from the database, and display it.

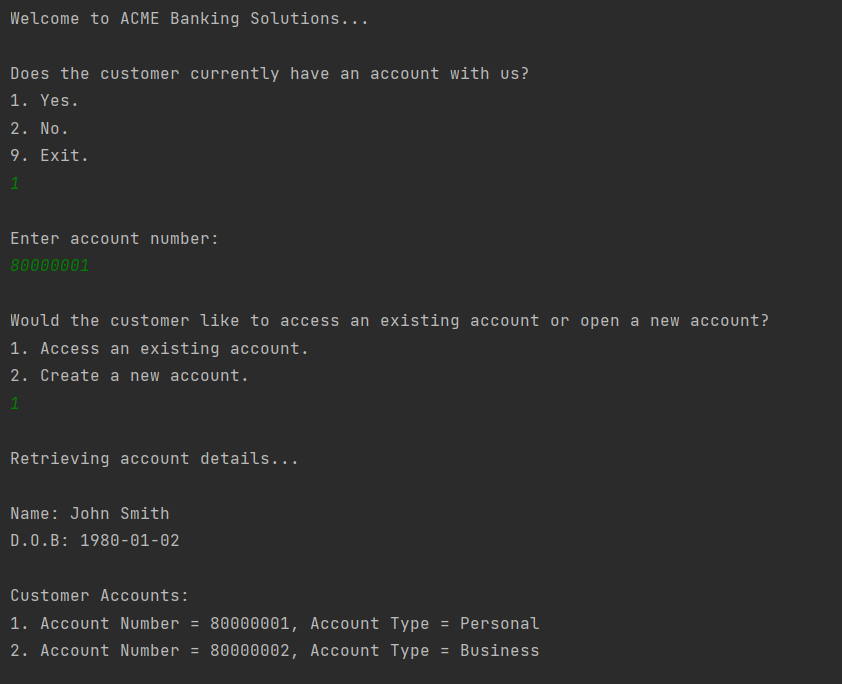




*Figure 4 - Opening a new ISA account*

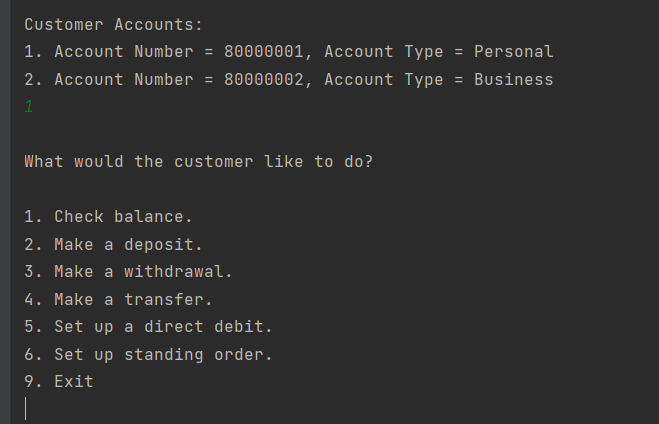
In the above figure (Fig 4) the client already has an account with the bank, but wants to open a new ISA account. This is an example of the menu system.

In the figure below (Fig 5) the customer already has an account with the bank and wants to access one of their existing accounts. When prompted to via the tellers input the program would display the customers details (Name, D.O.B) and all the existing accounts they hold.

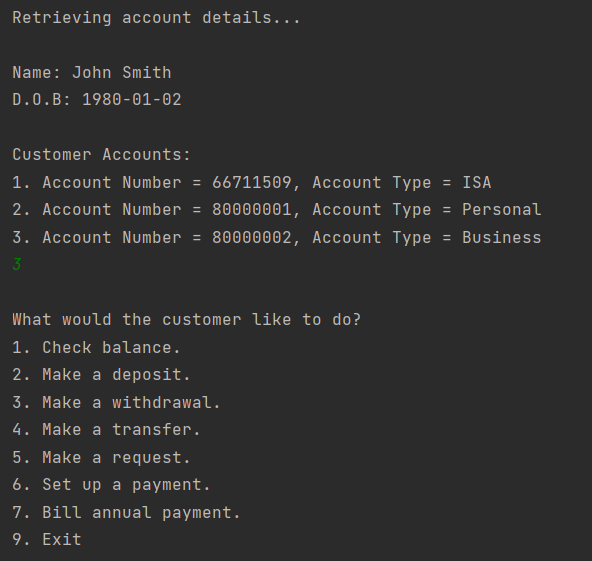


*Figure 3 - Accessing an existing account*

The teller would then ask the client what account they wanted to access. Depending on the account number/type the program would bring up a different menu for the options the user had. For example a personal account would have different options to a business account.



*Figure 4 - Accessing personal account menu.*



*Figure 5 - Accessing an ISA account menu.*

Using this menu format the bank teller can access all the functionality of the system in a clear and concise way, with appropriate exits at each step of the menu stage.

Assumptions we made.

* The primary assumption we made was that it was the bank teller inputting the information and so all checks/processes were done by the bank rather than the program.
  + ID checks for the personal account would be done by the bank teller and only a ‘Yes’ or ‘No’ input into the program would be required for the client providing a valid ID rather. ID checking functionality done by the bank rather than being included in the program.
* We assumed that the sort code would be the same for this bank, so we chose a number at random: 02-12-20.
* We assumed that the account number could be any 8 digit random number, and thus we generate a random 8 digit number for each new account (duplicates are not permitted).
* Transactions and transfers could only be done internally for the Bank, the bank does not communicate with any other banks, i.e. a transfer cannot be made from this bank to another bank but can be made between 2 accounts within this bank.
* All transfers, Deposits, Withdrawals are made instantly. Time does not factor into any of the functionality of this application.
* No time functionality for ISA interest (interest is not paid by year as there is no way of determining the time passed within this program), rather the bank teller can dictate to the program when interest needs to be paid via an input, and the program will pay the monthly equivalent of 3% APR using a method within the application. Thus every month that interest needs to be paid the teller would have to manually dictate to the program that this would need to happen before the program fulfilled this.

**Initial Functional Requirements**

All Accounts

* Teller ability to check balance
* Teller ability to set up direct debit.
* Teller ability to set up standing order.
* Teller ability to move money to/from accounts.
* First Name, Last Name, Date of Birth

Personal Account

* User ability to open an account with identification and £1 minimum balance
* Application ability to verify identification
* Application ability to verify £1 balance
* User ability to have multiple accounts

ISA Account

* Application ability to limit user balance to £20,000 maximum
* Application ability to pay 2.75% APR interest on average user balance
* User ability to open account with identification
* Application ability to verify identification and user age (age dependent on ISA type)
* Application ability to impose limit of one ISA per user

Business Account

* User ability to open account with business credentials
* Application ability to verify business credentials (gov.uk registration?)
* Application ability to verify business type (no PLC, charity etc.)
* Application ability to limit business to one account
* User ability to request cheque book
* User ability to request overdraft
* Application ability to provide overdraft to user account
* User ability to request credit/debit cards (issuing handled by another department?)
* User ability to request loans (issuing handled by another department?)
* Application ability to impose £120 annual charge

**SQL Queries:**

A table denoting the queries we used to fulfil aspects of the requirements of the program.

| **Requirement** | **Arguments** | **Query Example(s)** |
| --- | --- | --- |
| Ability to check balance | AccountNumber (e.g. 80000001) | SELECT Balance FROM Accounts WHERE AccountNumber = 80000001; |
| Ability to set up direct debit / standing order / transfer between accounts | AccountNumber (x2), Balance (x2), Amount (e.g. £10) | UPDATE Accounts SET Balance = Balance + 10 WHERE AccountNumber = 80000001; |
| UPDATE Accounts SET Balance = Balance - 10 WHERE AccountNumber = 80000002; |
| Ability to make a deposit | AccountNumber, Balance, Amount | UPDATE Accounts SET Balance = Balance + 10 WHERE AccountNumber = 80000001; |
| Ability to make a withdrawal | AccountNumber, Balance, Amount | UPDATE Accounts SET Balance = Balance - 10 WHERE AccountNumber = ‘80000001’; |
| Ability to get user details | AccountNumber | SELECT FirstName, LastName, DateOfBirth from Users WHERE ID in (SELECT UserID FROM Accounts WHERE AccountNumber = ‘80000001’); |
| Ability to get user accounts | AccountNumber | SELECT AccountNumber, AccountType FROM Accounts WHERE UserID IN (SELECT UserID FROM Accounts WHERE AccountNumber = ‘80000001’); |
| Ability to open new personal account | FirstName, LastName, DateOfBirth (create new user) | INSERT INTO Users (FirstName, LastName, DateOfBirth) VALUES (‘John’, ‘Smith’, 2000-01-01); |
| AccountNumber, FirstName, LastName, DateOfBirth, Balance (create new account) | INSERT INTO Accounts (UserID) SELECT ID FROM Users WHERE FirstName = ’John’ AND LastName = ’Smith’ AND DateOfBirth = 2000-01-01;  INSERT INTO Accounts (AccountNumber, SortCode, AccountType, Balance, Overdraft) VALUES (‘80000001’, ‘02-12-20’, ‘Personal’, 1, 0) WHERE UserID IN (SELECT ID FROM Users WHERE FirstName = ‘John’ AND LastName = ‘Smith’ AND DateOfBirth = 2000-01-01);; |
| Ability to open new business account | FirstName, LastName, DateOfBirth (create new user) | INSERT INTO Users (FirstName, LastName, DateOfBirth) VALUES (‘John’, ‘Smith’, 2000-01-01); |
| AccountNumber, FirstName, LastName, DateOfBirth, Balance (create new account) | INSERT INTO Accounts (UserID) SELECT ID FROM Users WHERE FirstName = ’John’ AND LastName = ’Smith’ AND DateOfBirth = 2000-01-01;  INSERT INTO Accounts (AccountNumber, SortCode, AccountType, Balance, Overdraft) VALUES (‘80000002’, ‘02-12-20’, ‘Personal’, 1, 0) WHERE UserID IN (SELECT ID FROM Users WHERE FirstName = ‘John’ AND LastName = ‘Smith’ AND DateOfBirth = 2000-01-01);; |
| BusinessName, Account Number (create new business) | INSERT INTO Businesses (Name, AccountNumber) VALUES (‘Smith Computing Ltd.’, ‘80000002’); |
| Ability to display business name | AccountNumber | SELECT Name FROM Businesses WHERE AccountNumber = ‘80000002’; |