Umayir Feroze

FEUMI163

Final Assignment

InterBanking Pty

Assignment Stage 07

Contents

[Introduction 2](#_Toc505353892)

[Diagrams 2](#_Toc505353893)

[UML Class Diagram 2](#_Toc505353894)

[Flow Chart 3](#_Toc505353895)

[Flow Chart: Main Flow 3](#_Toc505353896)

[Flow Chart: Bank Branch 4](#_Toc505353897)

[Flow Chart: Create Account 4](#_Toc505353898)

[Flow Chart: View Account 5](#_Toc505353899)

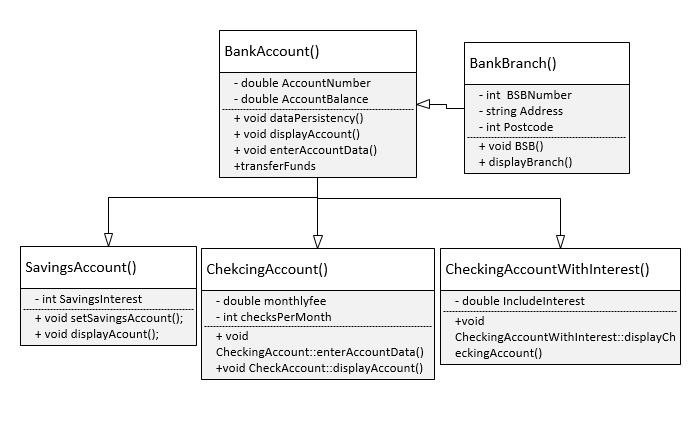
[Source Code 6](#_Toc505353900)

# Introduction

Inter Banking Pty is a new Internet Bank commissioned to develop a new generation customer and account management system. Thus through this case study we have been guided through the design, coding, and testing if required.

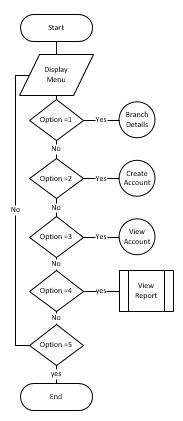
# Diagrams

## UML Class Diagram

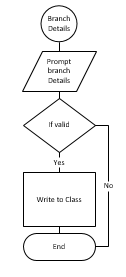


## Flow Chart

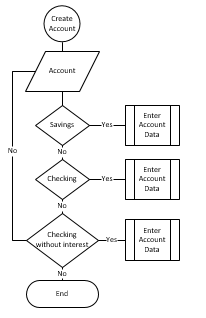
### Flow Chart: Main Flow



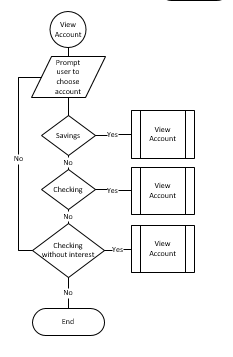
### Flow Chart: Bank Branch



### Flow Chart: Create Account



### Flow Chart: View Account



# Source Code

// Stage 07.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include <string>

#include <fstream>

#include <iomanip>

#include <sstream>

using namespace std;

class BankAccount

{

private:

double AccountNumber = 0;

double AccountBalance = 0;

public:

void dataPersistency();

void displayAccount();

void enterAccountData();

void computeInterest(int year);

void transferFunds();

BankBranch homebranch();

BankAccount();

};

BankAccount::BankAccount()

{

BankAccount::AccountBalance = 0;

}

void BankAccount::enterAccountData()

{

bool loop;

std::cout.precision(2);

std::cout << std::fixed;

while (BankAccount::AccountNumber < 1000 || BankAccount::AccountNumber)

{

BankAccount::AccountNumber = rand() % 9999 + 1000;

}

do

{

cout << "Account Number:" << BankAccount::AccountNumber;

do

{

cout << "Enter Opening Account Balance:";

cin >> BankAccount::AccountBalance;

if (BankAccount::AccountBalance <= 0 || BankAccount::AccountBalance >= 100000)

{

cout << "invalid Account Balance...! Please Re Enter" << endl;

loop = false;

}

else

{

loop = true;

}

} while (loop == false);

cout << endl;

counter++;

} while (loop = false);

}

void BankAccount::displayAccount()

{

std::cout.precision(2);

std::cout << std::fixed;

cout << "----Account Details----" << endl;

cout << "Account Number: " << BankAccount::AccountNumber << endl;

cout << "Account Balance: " << BankAccount::AccountBalance << endl;

}

void BankAccount::dataPersistency()

{

ofstream writeFile("BankAccount.txt", ios::app);

if (writeFile.is\_open())

{

writeFile << "\nAccount Number :" << BankAccount::AccountNumber << endl;

writeFile << "Account Number :" << BankAccount::AccountBalance << endl;

writeFile << "Account Interest Rate :" << BankAccount::InterestRate << endl;

cout << "\n----Account Created----";

}

else

{

cout << "\nCannot Open File";

}

writeFile.close();

}

class BankBranch

{

private:

int BSBNumber;

string Address;

int PostCode;

public:

BankBranch(int, string, int);

void BSB();

void displayBranch();

};

BankBranch::BankBranch(int BSB = 0000, string branchAddress = "-default-", int branchPostcode = 0000)

{

BSBNumber = BSB;

Address = branchAddress;

PostCode = branchPostcode;

}

void BankBranch::BSB()

{

cout << "Enter BSB Number";

cin >> BankBranch::BSBNumber;

cout << "Enter Branch Address";

cin >> BankBranch::Address;

cout << "Enter Branch Post Code";

cin >> BankBranch::PostCode;

cout << endl;

}

void BankBranch::displayBranch()

{

cout << "BSB number: " << BankBranch::BSBNumber << endl;

cout << "Branch Address: " << BankBranch::Address << endl;

cout << "Branch Post Code: " << BankBranch::PostCode << endl;

}

class SavingsAccount :public BankAccount

{

protected:

int SavingsInterest;

public:

SavingsAccount(double);

void setSavingsAccount();

void displayAccount();

};

SavingsAccount::SavingsAccount(double interestRate = 3.0)

{

SavingsAccount::SavingsInterest = interestRate;

SavingsAccount::setSavingsAccount();

}

void SavingsAccount::setSavingsAccount()

{

SavingsAccount::BankAccount::enterAccountData();

cout << "Enter interest rate: ";

cin >> SavingsAccount::SavingsInterest;

cout << endl;

}

void SavingsAccount::displayAccount()

{

cout << "\nAccount Number: " << SavingsAccount::AccountNumber;

cout << "\nAccount Balance: " << SavingsAccount::AccountBalance;

cout << "\nInterest Rate: " << SavingsAccount::SavingsInterest << endl;

}

class CheckingAccount :public BankAccount

{

protected:

double monthlyfee;

int checksPerMonth;

public:

CheckingAccount();

void CheckingAccount::enterAccountData();

void CheckingAccount::displayAccount();

};

CheckingAccount::CheckingAccount()

{

monthlyfee = 0;

checksPerMonth = 0;

}

void CheckingAccount::enterAccountData()

{

CheckingAccount::BankAccount::enterAccountData();

cout << "Monthly Fee: ";

cin >> monthlyfee;

cout << "Checks Per Month: ";

cin >> checksPerMonth;

cout << endl;

}

void CheckingAccount::displayAccount()

{

cout << "----Display Account----" << endl;

cout << "Account Number: " << CheckingAccount::accountNumber << endl;

cout << "Account Balance: " << CheckingAccount::AccountBalance << endl;

cout << "Monthly fee: " << CheckingAccount::monthlyfee << endl;

cout << "Checks Allowed Per Month: " << CheckingAccount::checksPerMonth << endl;

}

class CheckingAccountWithInterest :public SavingsAccount, public CheckingAccount, public BankAccount

{

protected:

double IncludeInterest;

public:

CheckingAccountWithInterest(double);

void CheckingAccountWithInterest::displayCheckingAccount();

};

CheckingAccountWithInterest::CheckingAccountWithInterest(double interest = 0.02)

{

CheckingAccountWithInterest::SavingsAccount::SavingsInterest = interest;

cout << "----Check With Interest----" << endl;

cout << "\nEnter Monthly Fee: ";

cin >> monthlyfee;

cout << "Enter Number of Checks Allowed Per Month: ";

cin >> checksPerMonth;

cout << endl;

}

void CheckingAccountWithInterest::displayCheckingAccount()

{

CheckingAccountWithInterest::SavingsAccount::displayAccount;

cout << "----View Account----" << endl;

cout << "\nMonthly Fee: ";

cout << "\nChecks Allowed Per Month: " << endl;

}

template <class T>

void produceReport(string get, T showArray)

{

cout << "::::::" << get << "::::::";

for (int count = 0; count < 3; count++)

{

cout << "Account Number " << count + 1 << " Details ";

showArray[count].displayAccount();

cout << endl;

}

}

BankAccount Customers[10];

BankBranch branch;

SavingsAccount savingsAccount;

CheckingAccount checkingAccount;

CheckingAccountWithInterest checkingAccountInt[5];

string Again;

int year;

static int counter;

int AccountA;

int AccountB;

int Amount;

int main()

{

bool loop=false;

int option,accountType,checkAccountType;

do

{

cout << "\n\n\t InterBanking Pty \t" << endl << endl;

cout << "(1) Branch Details \n(2) Create Account \n(3) View Account \n(4) View Report \n(5) Exit \nOption: ";

cin >> option;

//Branch Details

if (option == 1)

{

}

//Create Account

else if (option == 2)

{

do

{

cout << "\n----Create Account----" << endl;

cout << "Select Account Type: \n(1) Savings Account \n(2) Checking Account \n(3) Back\nOption: ";

cin >> accountType;

//Create Savings Account

if (accountType == 1)

{

cout << "\n----Create Savings Account----" << endl;

savingsAccount.SavingsAccount::setSavingsAccount();

savingsAccount.dataPersistency();

loop = false;

}

//Create Checking Account

else if (accountType == 2)

{

do

{

cout << "\n----Create Checking Account----" << endl;

cout << "Select Account Type: \n(1) Checking Account wihtout Interest \n(2) Checking Account with Interest \n(3)Back \nOption: ";

cin >> checkAccountType;

//Create Checkng Account without Interest

if (checkAccountType == 1)

{

cout << "\n----Checking Account w/ Interest----" << endl;

checkingAccount.CheckingAccount::enterAccountData();

checkingAccount.dataPersistency();

loop = false;

}

//Create Checking Account wihtout Interest

else if (checkAccountType == 2)

{

cout << "\n----Checking Account w/o Interest" << endl;

for (int count = 0;count < 5;count++)

{

checkingAccountInt[count].CheckingAccountWithInterest::CheckingAccountWithInterest();

}

loop = false;

}

//Back

else if (checkAccountType == 3)

{

loop = true;

}

else

{

cout << "\nIncorrect Account Choice...!" << endl;

loop = false;

}

} while (loop == false);

loop = false;

}

else if (accountType == 3)

{

loop = true;

}

else

{

loop = false;

}

} while (loop == false);

}

//View Account

else if (option == 3)

{

do

{

cout << "\n----View Account----" << endl;

cout << "Choose Account: \n(1) Savings Account \n(2) Checking Account/w/ Interest \n(3) Checking Account w/o Interest \n(4)Back \nOption: ";

cin >> option;

//View Savings Account

if (option == 1)

{

cout << "\n----View Savings Account----" << endl;

savingsAccount.SavingsAccount::displayAccount();

loop = false;

}

//View Checking Account without Interest

else if (option == 2)

{

cout << "\n----View Checking Account w/o Interest----" << endl;

checkingAccount.CheckingAccount::displayAccount();

loop = false;

}

else if (option == 3)

{

cout << "\n----View Checking Account w/ Interest----" << endl;

for (int count = 0; count < 5;count++)

{

checkingAccountInt[count].CheckingAccountWithInterest::displayCheckingAccount();

}

loop = false;

}

//Back

else if (option == 4)

{

loop = true;

}

else

{

cout << "\ninvalid Input...!" << endl;

loop = false;

}

} while (loop == false);

}

//View Report

else if (option == 4)

{

for (int count = 0;count < counter;count++)

{

//produceReport(get,showArray)

}

loop = false;

}

//Exit

else if (option == 5)

{

cout << "\nThank You.. ! See You Agian..!" << endl;

break;

loop = true;

}

else

{

cout << "\nInvalid Input..!" << endl;

loop = false;

}

} while (loop==false);

return 0;

}