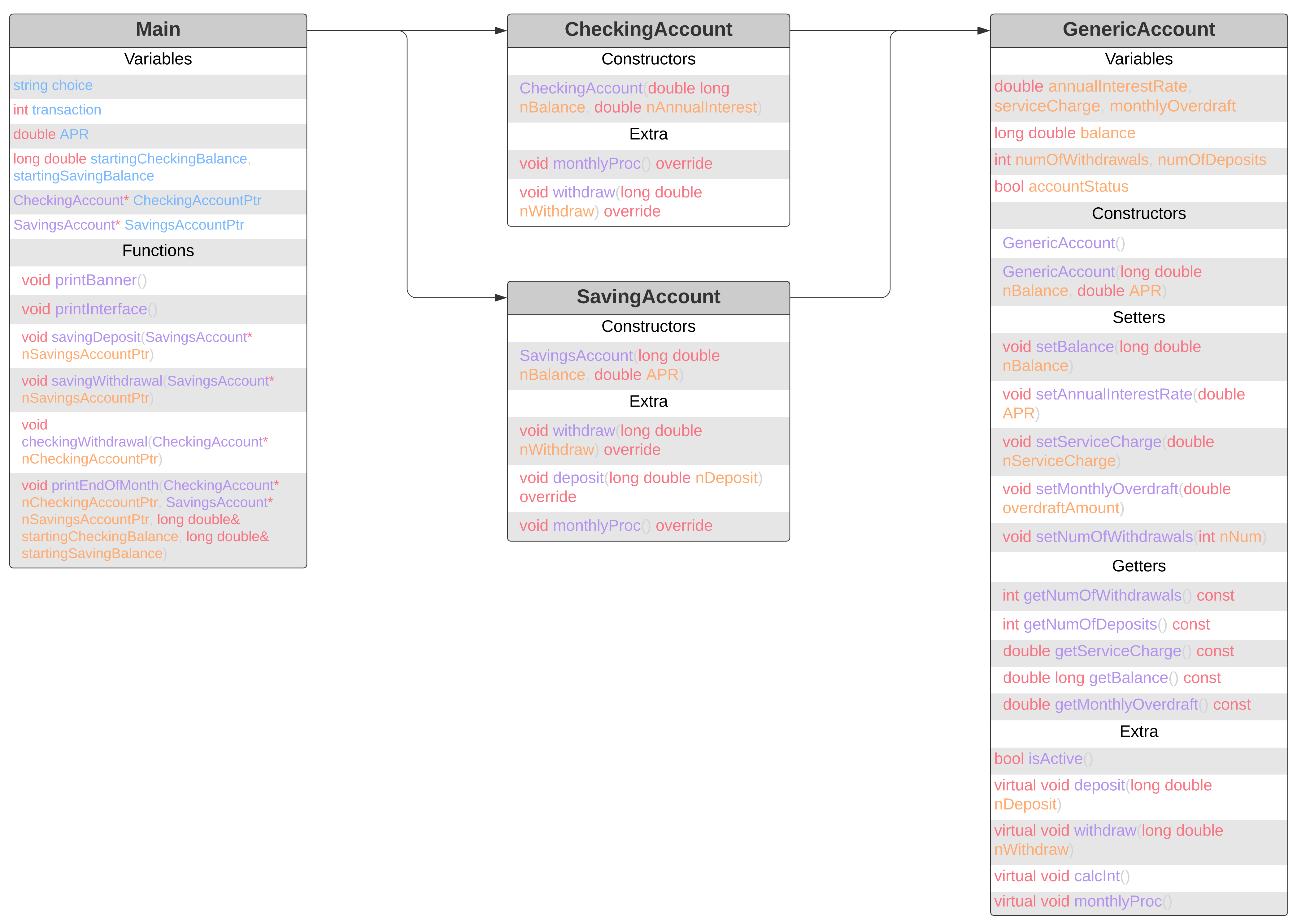
BankAccounts High Level Program Details

Yelsin Sepulveda | March 12, 2021



Main

Variables

- string choice
- int transaction
- double APR
- long double startingCheckingBalance, startingSavingBalance
- CheckingAccount* CheckingAccountPtr
- SavingsAccount* SavingsAccountPtr

Functions

- void printBanner()
- void printInterface()
- void savingDeposit(SavingsAccount* nSavingsAccountPtr)
- void savingWithdrawal(SavingsAccount* nSavingsAccountPtr)
- void checkingWithdrawal(CheckingAccount* nCheckingAccountPtr)
- void printEndOfMonth(CheckingAccount* nCheckingAccountPtr, SavingsAccount* nSavingsAccountPtr, long double& startingCheckingBalance, long double& startingSavingBalance)

CheckingAccount

Constructors

- CheckingAccount(double long nBalance, double nAnnualInterest)

Extra

- void monthlyProc() override
- void withdraw(long double nWithdraw) override

SavingAccount

Constructors

- SavingsAccount(long double nBalance, double APR)

Extra

- void withdraw(long double nWithdraw) override
- void deposit(long double nDeposit) override
- void monthlyProc() override

GenericAccount

Variables

- double annualInterestRate, serviceCharge, monthlyOverdraft
- long double balance
- int numOfWithdrawals, numOfDeposits
- bool accountStatus

Constructors

- GenericAccount()
- GenericAccount(long double nBalance, double APR)

Setters

- void setBalance(long double nBalance)
- void setAnnualInterestRate(double APR)
- void setServiceCharge(double nServiceCharge)
- void setMonthlyOverdraft(double overdraftAmount)
- void setNumOfWithdrawals(int nNum)

Getters

- int getNumOfWithdrawals() const
- int getNumOfDeposits() const
- double getServiceCharge() const
- double long getBalance() const
- double getMonthlyOverdraft() const

Extra

- bool isActive()
- virtual void deposit(long double nDeposit)
- virtual void withdraw(long double nWithdraw)
- virtual void calcInt()
- virtual void monthlyProc()

High Level Workflow

1

Declare varibles: Initialize 'choice', 'CheckingAccountPtr', and 'SavingsAccountPtr' with default values.
choice = "yes", CheckingAccountPtr = nullptr, SavingsAccountPtr = nullptr

2

Print program banner: Explains to user what the program is - void printBanner().
Collect user input: Starting balance for Checking Account(startingCheckingBalance), Saving Account (startingSavingBalance) and Annual Interest Rate (APR). Then create a new CheckingAccount, and SavingsAccount object.

3 while (choice[0] == 'y' || choice[0] == 'Y') { try {

Print program interface: Explains to user how to interface with program - void printInterface().
Collect user input for desired transaction (transaction). Check to make sure user input is valid.

4 switch (transaction) {

a. checkingDeposit(CheckingAccountPtr)

b. checkingWithdrawal(CheckingAccountPtr)

c. savingDeposit(SavingsAccountPtr)

d. savingWithdrawal(SavingsAccountPtr)

e. printEndOfMonth(CheckingAccountPtr, SavingsAccountPtr, startingCheckingBalance, startingSavingBalance)
return 0

5

* Catch any errors / exceptions and print them out to console, if necesarry.
Collect user input for determinig wether or not to perform another transaction (choice)

Function Definitions

GenericAccount

bool isActive()

- Checks if savings account balance falls below 25. if true , set accountStatus to false. if false , set accountStatus to true.

virtual void deposit(long double nDeposit)

- Accepts an argument for the amount f the deposit. The function should add the argument to the account ablance. It should also increment the variable holding the number of deposits.

virtual void withdraw(long double nWithdraw)

- Accepts an argument for the amount of the withdrawal. The funciton should subtract the argument from the balance. It should also increment the variable holding the number of wirthdrawals.

virtual void calcInt()

- Updates the balce by calculating the monthyl interest earned by the account, and adding this interest to the balance.
Formulas:
Monthly Interest Rate = (Annual Iterest Rate / 12)
Monthly interest Rate = Balance * Monthly Interest Rate
Balance = Balance + Monthly Interest

virtual void monthlyProc()

- Subtract the monthly service charges from the balance, calls the calcInt() function, then sets the variables that hold the number of withdrawals, number of deposits, ad monthly service charges to zero.

CheckingAccount

void monthlyProc() override

- Before the base class function is called, this function adds the monthly fee of 5 plus 0.10 per withdrawal to the base class bariable that holds the monthly service charge.

void withdraw(long double nWithdraw) override

- Before the base class function is called, this function will determine if a withdrawal will cause the balance to go below 0. If the balance goes below 0, a service charge of 15 will be taken from the account. The withdrawal will not be made. If there isnt enoughin the account to pay the service charge, the balance will become negative and the customer will owe the negative amount to the bank.

SavingAccount

void withdraw(long double nWithdraw) override

- Checks to see if te account is inactive before a withdrawal is made. No withdrawal is made if the account is not active. A withdrawal is then made by caling the base class version of the funtion.

void deposit(long double nDeposit) override

- Check to see if the account is inactive before a deposite is made If the account is inactive and the deposit brings the balance above 25, the account becomes active again. The deposit is then made by calling the base class version fo the funciton.

void monthlyProc() override

- Before the base class functions, this function checks the number of wirthdrawals. If the number of withdrawal for the month uf more than 4, a service charge of 1 for each withdrawal above 4 is added to the base class variable that holds the monthly service charges. If the account balance falls before 25, the account becomes inactive.

Main

void printBanner()

- Explains to user what the program is.

void printInterface()

- Explains to user how to interface with program.

void savingDeposit(SavingsAccount* nSavingsAccountPtr)

- Collect user input for a new variable, 'newDeposit', and then calls nSavingsAccountPtr->deposit(newDeposit)

void savingWithdrawal(SavingsAccount* nSavingsAccountPtr)

- Collect user input for a new variable, 'newWithdrawal', and then calls nSavingsAccountPtr->withdraw(newWithdrawal)

void checkingDeposit(CheckingAccount* nCheckingAccountPtr)

- Collect user input for a new variable, 'newDeposit', and then calls nCheckingAccountPtr->deposit(newDeposit)

void checkingWithdrawal(CheckingAccount* nCheckingAccountPtr)

- Collect user input for a new variable, 'newWithdrawal', and then calls nCheckingAccountPtr->deposit(newDeposit)

void printEndOfMonth(CheckingAccount* nCheckingAccountPtr, SavingsAccount* nSavingsAccountPtr, long double& startingCheckingBalance, long double& startingSavingBalance)

- Collect user input for account to print, stored in 'accountChosen' .
if accountChosen == 1, Print Cheking account. if accountChosen == 2, Print Saving account.