**ARUBANK SYSTEM IN C++**

### May 07, 2018

**MUSTAFA ARU**

### 030716006

**Note:** The **main** purposes of the project are described in this file and some of them are explained in the presentations.

**Sources:** 1.cplusplus.com 2.Code Yarned C++ Forms

### ISTANBUL SABAHATTIN ZAIM UNIVERSITY

## The Bank

Banking is an industry that handles cash, credit, and other financial transactions. Banks provide a safe place to store extra cash and credit. They offer savings accounts, [certificates of deposit](https://www.thebalance.com/certificates-of-deposit-3305913) and checking accounts. Banks use these deposits to make loans. These loans include home mortgages, business loans, and car loans.

Banking is one of the key drivers of the U.S. economy. Why? It provides the [liquidity](https://www.thebalance.com/liquidity-definition-ratios-how-its-managed-3305939) needed for families and businesses to invest for the future.

Bank loans and credit mean families don't have to save up before going to college or buying a house. Companies can start hiring immediately to build for future [demand](https://www.thebalance.com/what-is-demand-definition-explanation-effect-3305708) and expansion.

**How it works:** Banks are a safe place to deposit excess cash. That's because the [Federal Deposit Insurance Corporation](https://www.thebalance.com/federal-deposit-insurance-corporation-fdic-3305984) insures them. Banks also pay a small percent, the interest rate, on the deposit.

Banks can turn every one of those saved dollars into $10. They are only required to keep 10 percent of each deposit on hand. That regulation is called the [reserve requirement](https://www.thebalance.com/reserve-requirement-3305883). Banks lend the other 90 percent out. They make money by charging higher interest rates on their loans than they pay for deposits.

## The Abstract

Data abstraction refers to providing only essential information to the outside world and hiding their background details, i.e., to represent the needed information in program without presenting the details.

Data abstraction is a programming (and design) technique that relies on the separation of interface and implementation.

Let's take one real life example of a TV, which you can turn on and off, change the channel, adjust the volume, and add external components such as speakers, VCRs, and DVD players, BUT you do not know its internal details, that is, you do not know how it receives signals over the air or through a cable, how it translates them, and finally displays them on the screen.

Thus, we can say a television clearly separates its internal implementation from its external interface and you can play with its interfaces like the power button, channel changer, and volume control without having zero knowledge of its internals.

In C++, classes provide great level of **data abstraction**. They provide sufficient public methods to the outside world to play with the functionality of the object and to manipulate object data, i.e., state without actually knowing how class has been implemented internally.

For example, your program can make a call to the **sort()** function without knowing what algorithm the function actually uses to sort the given values. In fact, the underlying implementation of the sorting functionality could change between releases of the library, and as long as the interface stays the same, your function call will still work.

Data abstraction provides two important advantages:

* Class internals are protected from inadvertent user-level errors, which might corrupt the state of the object.
* The class implementation may evolve over time in response to changing requirements or bug reports without requiring change in user-level code.

By defining data members only in the private section of the class, the class author is free to make changes in the data. If the implementation changes, only the class code needs to be examined to see what affect the change may have. If data is public, then any function that directly access the data members of the old representation might be broken.

**In the program:**

**private**: *//-*

**int** AccountNo;

string Name;

**float** Deposit;

**char** Type;

**public**: *//+*

*//Extras*

**void** createAccount();

**void** showAccounts(); *//show on the screen*

*//Sets*

**char** setType(**char** type);

**void** setDeposit(**float** deposit);

**void** setAccountNo(**int** accountNo);

*//Gets*

**char** getType()**const**;

string getName()**const**;

**float** getDeposit()**const**;

**int** getAccountNo()**const**;

In that sample show how to use abstraction in C++. The customer or the person who took that program, he can not access the private values by not using set or get functions.

## PSEUDO CODE

## Start

## Write the options

## Switch-Case

## If decision is 1 go to 13

## If decision is 2 go to 16

## If decision is 3 go to 20

## If decision is 4 go to 22

## If decision is 5 go to 24

## If decision is 6 go to 28

## If decision is 7 go to 31

## If decision is 8 go to

## If decision is 9 go to

## Create the new account

## Take the information

## Open the file and write on it

## Research the account number

## If you find go to 19

## Write couldn’t fine!

## Write on the screen the information

## Find the account number and take money

## Give the negative parameter for that

## Find the account number and give money

## Give the positive parameter for account

## Ask the password

## If it is correct go to 27

## Send error message

## Achieve all lines and show them

## Create the new text

## Write the same thing when you do not find the parameter (Account No)

## Remove the old one and rename the new one

## Show the test and take points

## Show exit massage

## Finish

**Implementation**

The main purpose of using implementation as I mention in the slides that not writing the same codes again and again. We should write one time then save the codes afterward, we need to use them when we need. Implementation in that program:

**int** ARUBANK::getAccountNo()**const**{

**return** AccountNo;

}

**float** ARUBANK::getDeposit()**const**{

**return** Deposit;

}

string ARUBANK::getName()**const**{

**return** Name;

}

**char** ARUBANK::getType()**const**{

**return** Type;

}

**char** ARUBANK::setType(**char** type){

Type=type;

}

**void** ARUBANK::setDeposit(**float** deposit){

Deposit=deposit;

}

**void** ARUBANK::setAccountNo(**int** accountNo){

AccountNo=accountNo;

}

**void** ARUBANK::showAccounts(){ *//on screen*

cout<<setw(18)<<"Account Number"<<setw(18)<<"Account Holder"<<setw(18)<<"Type of Account"<<setw(18)<<"Initial Amount"<<endl;

cout<<setw(18)<<AccountNo<<setw(18)<<Name<<setw(18)<<Type<<setw(18)<<Deposit<<endl<<endl<<endl;

}

**void** ARUBANK::createAccount(){

cout<<"Enter The Account No: "; cin>>AccountNo;

cout<<"Enter The Name of The Account Holder :"; cin>>Name;

**do**{

cout<<"Enter The Type of Account [C/S] :"; cin>>Type;

}**while**(Type!='C' && Type!='c' && Type!='s' && Type!='S'); *//not any wrong option for type*

**if**(Type!='C' || Type!='S') Type=toupper(Type); *//transfer to upper letter.*

**do**{

cout<<"Enter Your Initial Amount: "; cin>>Deposit;

}**while**(Deposit<=0); *//not any wrong option for Deposit*

## The Body of the ARUBANK SYSTEM



**Start**

Giving the money

How many?

* Check the account

**Showing the options**

Creating the account

* Account number
* Account holder name
* Type and Money

Changing the information by withdraw money

Switch Case

Case 1,2,3,4,5,6,7,8,9

#### Decisions

After giving money check account again

Look at again

* sources
* method

Account number is correct massage

Is open function work?

If work, just keep going asking

If all account information true

Show that in the file

* generally

Sleep() ; To see clearly

If exit button on , show good bye message?

Close the files all of them

FINISH

If the current money is not bigger than the money that you want to take, it must be error for that.

If doesn’t exist, the file would give ERROR

If the Error continue in all step do not show anything.

Start again

Delete the accounts

If they want to delete

* Don’t ask the task for that

Good bye message

* Sleep use here
* Good Bye message

If the file couldn’t open

Send error message again

Do not give his information

Say good bye

Say press key

If he press

Finish