

**MERU UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF COMPUTING AND INFORMATICS**

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE**

**UNIT: OBJECT ORIENTED PROGRAMMING 1**

**UNIT CODE: CIT 3153**

**GROUP MEMBERS**

1. VERONICAH ANZIMBU – CT201/0017/18
2. THOMAS MUTINDA– CT201/0052/18
3. VICTOR MAINA – CT201/0037/18
4. LYDIA KANGARA– CT201/0032/18
5. MARTIN MWENDA – CT201/0049/18
6. MARTIN LANG’AT– CT201/0041/18
7. SEAN GIKENO – CT201/0048/18
8. ELKANA SANG– CT201/0058/18

**IDENTIFICATION OF AN APPROPIATE PROBLEM AND A VIABLE SOLUTION**

Bank tellers get a hard time helping huge number of customers to perform accounting related tasks such as deposits, withdrawals and creation of bank accounts. We hereby write a simple program to help reduce congestion in banks by allowing customers to create accounts, check account balances, to perform deposits and withdrawals all by themselves without teller intervention.

**PROGRAMMING CODE TO SOLVE THE PROBLEM**

#include<iostream>

using namespace std;

//This class stores the general account features such as account name, balance and password.

class Account

{

public:char acc\_name[10];

int acc\_balance;

char password;

};

/\*This class contains banking related functions such as depositing, withdrwal

and creation of accounts.\*/

class Banker:private Account

{

public:void create\_account()

{

char firstname[10];

char secondname[10];

int id\_no;

cout<<"ENTER YOUR FIRST AND SECOND NAME "<<endl;

cin>>firstname>>secondname;

cout<<"ENTER YOUR ID NUMBER "<<endl;

cin>>id\_no;

cout<<"ENTER YOUR PASSWORD "<<endl;

cin>>password;

//acc\_name=name;

acc\_balance=0;

cout<<"Successfully Created Account "<<firstname<<" "<<secondname<<" Thank You"<<endl;

}

void deposit()

{

int amount;

cout<<"ENTER AMOUNT TO DEPOSIT: ";

cin>>amount;

if (amount<0)

cout<<"AMOUNT CAN'T BE NEGATIVE: ";

else

acc\_balance+=amount;

cout<<"Balance is "<<amount<<endl;

}

void show\_balance()

{

cout<<"Your account balance is: Kshs"<<acc\_balance<<endl;

}

void withdraw()

{

int amount;

cout<<"Current account balance is"<<acc\_balance<<endl;

cout<<"ENTER AMOUNT TO WITHDRAW"<<endl;

cin>>amount;

if (acc\_balance>=amount)

{

acc\_balance-=amount;

cout<<"Current account balance is"<<acc\_balance<<endl;

}

else

{

cout<<"Sorry! Your account balance is too low"<<endl;

}

}

};

int main()

{

//We create an object bk to access the members of class Banker.

Banker bk;

char input;

cout<<"Step 1.Create an account"<<endl;

bk.create\_account();

cout<<"PLEASE CHOOSE AN OPTION FROM THE MENU OR ENTER 'q' TO QUIT"<<endl;

do

{

cout<<"1.Deposit money to an account"<<endl;

cout<<"2.Withdraw from your account"<<endl;

cout<<"3.Show balance of your account"<<endl;

cin>>input;

/\*The switch below enables us to call the necessary functions depending on the

user's options\*/

switch(input)

{

case '1':

bk.deposit();

break;

case '2':

bk.withdraw();

break;

case '3':

bk.show\_balance();

break;

case 'q':

case 'Q':

cout<<"Thank you customer for using our services"<<endl;

break;

default:

cout<<"invalid character selected"<<endl;

}

//The sentinel below enables the user to terminate the program when entering q.

}while((input!='q')&&(input!='Q'));

return 0;

}

**PROGRAMMING CONCEPTS**

Programming concepts involved include:

1. Encapsulation-We have put the member functions such as deposit, create\_account and member data such as password, acc\_name into their appropriate classes Banker and Account.
2. Single Inheritance- We have the derived class banker inheriting privately from class account in order to access members’ class account.
3. Sentinel-We allow the user to keep on performing tasks and terminate by entering the character q to quit.
4. Switch- We used the switch to call the appropriate functions depending on the user’s choice.