BANK MANAGEMENT SYESTEM

BY:

NC ADEEL ZAFAR

Department of Computer Software Engineering

Military College of Signals, National University of Sciences and Technology, Islamabad

BANK MANAGEMENT SYSTEM

The Bank management system is an application for maintaining a person's account in a bank. The system provides the access to customer, officers and manager to create an account, deposit/withdraw the cash from his account and also view the report of all the accounts.

INTRODUCTION:

Objectives:

1. To provide manager an easy interface to view reports of the accounts.
2. To facilitate officers in creating, maintaining and deleting customer accounts.
3. To accelerate the withdrawing and depositing routines of the customers.

Project Specifications:

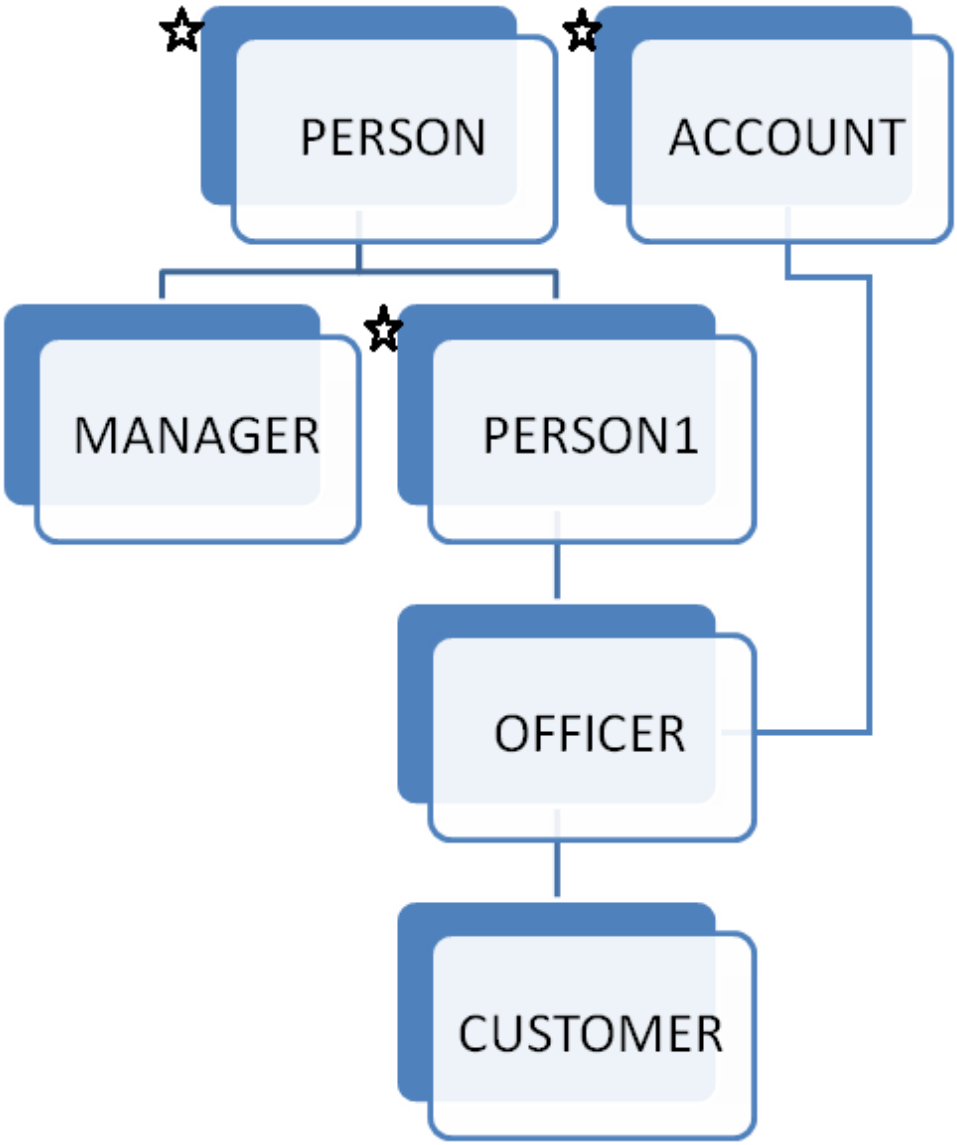
- Tools used: Borland, Microsoft Visual Studio 2008.
- Development Language: C++

Implementation Techniques:

- Classes
- Inheritance
- Polymorphism
- Pointers

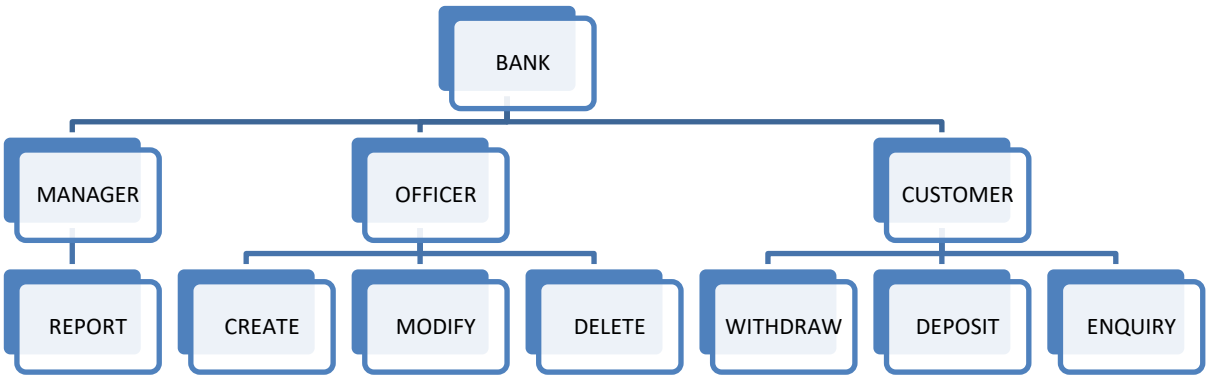
IMPLEMENTATION DETAILS:

Following is the Hierarchy of classes of our Bank Management System:



Note: The classes with star are abstract.

Following are the functions of our Bank Management System:



File handling:

Files are used to store data in a relatively permanent form on any form of secondary storage (like on floppy, hard disk, tape). Files can hold huge amount of data when needed. When filing is not used the ordinary variables are kept in main memory which is temporary and rather limited in size and washes away when the execution is ended.

Streams:

A stream is a general name given to flow of data. Different streams are used to represent different kinds of data flow. Each stream is associated with particular class, which contains

member functions and definitions for dealing with that particular kind of data flow. C++ provides the following classes to perform output and input of characters to/from files:

- **ofstream:** Stream class to write on files
- **ifstream:** Stream class to read from files
- **stream:** Stream class to both read and write from/to files

BASE CLASSES:

There are two base classes in the program which are:

- **Person**
- **Account**

These classes will be the base for derived classes which will inherit its functions and member functions.

Class Person:-

This class contain name and cnic as protected data members. It contains constructors and dataEntry() and print() as member functions. This class is absolute because dataEntry function is pure virtual. So objects of this class can't be created.

Declaration code of this class is :

```
class person

{

protected:

        char name[20];

        char cnic[20];

public:

        person();

        person(char[],char[]);

        virtual void dataEntry()=0;

        void print();

};
```

Here dataEntry() takes input name and cnic and print() gives the output of name and cnic.

Class Account:

This class contain data members for account no. , account type and balance. It contains constructors, print(), retacno(), retbal(), rettype() as member functions. This class is absolute because dataEntry function is pure virtual. So objects of this class can't be created. Concept of pointers is also used in this class.

Declaration Code of this class is :

```
class account
{
protected:
    int acno;
    char type;
    float balance;
public:
    account();
    account(int,char,float);
    virtual void dataEntry()=0;
    void print();
    int* retacno();
    float* retbal();
    char* rettype();
};
```

Here dataEntry() takes the input of the data members from the users and print() gives the output() of the data members. retacno(), retbal() and retype() function returns the account number , balance and account type respectively.

DERIVED CLASSES:

Following are the derived classes in the program.

- **Person1**
- **Manager**
- **Officer**
- **Customer**

Class Person1:-

This class is derived from the class person. So it posses all data members and functions of person. Besides this it also contains extra protected data members for address, profession and age. It contains constructors and dataEntry() and print() as member functions. This class is absolute because dataEntry function is pure virtual. So objects of this class can't be created.

Declaration code of this class is :

```
class person1:public person
{
protected:
    char addr[20];
    char prof[20];
    int age;
public:
    person1();
    person1(char[],char[],char[],char[],int);
    virtual void dataEntry()=0;
    void print();
};
```

Here dataEntry() and print() functions are used to take input and give output of the data members. Fucntion overriding is also used in both of these classes.

Class Manager:

This class is derived from class person so it contains all the data members and functions of person. Besides this it contains an extra data member for password. It contains constructors , dataEntry(), print() and verify() as member functions.

Declaration Code of this class is :

```
class manager:public person
{
private:
    int pass;
public:
    manager();
    manager(int,char[],char[]);
    void dataEntry();
    void print();
    bool verify(manager);
};
```

Here dataEntry() and print() takes input() and gives output(). Function overriding is used in both of these functions. verify() gives the verification of the password if the entered password is same as passed through the constructor or not.

Class Officer:

This class is derived from two classes i.e. account and person1. It contains all the functionality of the classes and contains a private data member for password. It contains constructors, dataEntry(), create(), modify(), report() , check() and verify() as member functions.

Declaration code of this class is :

```
class officer: public person1, public account
{
private:
```

```
        int pass;
public:
        officer();
        officer(char [], char[] , char [], char [], int , int , char , float ,int );
   officer(int);
        void dataEntry();
        void create();
        void modify();
        void report();
        void check();
         bool verify(officer);
};
```

Here

- dataEntry() : Takes input for the password.
- create(): Takes input for name, cnic, address, profession, age, account no. , account type and balace.
- modify(): Takes the same input again as mentioned in create() to overwrite a particular account.
- report(): Gives the report of all the accounts.
- check(): Gives output of the particular account no.
- verify(): Verifies if the entered password in the main() matches the one passed through constructor or not.

Class Customer:

This class is derived from the class officer. It possesses all the member functions of officer except password because its private in officer. It contains constructors , deposit() , draw() and check() as member functions. Concept of pointers is also used in this class.

Declaration code of this class is :

```
class customer:public officer
```

```
{
public:
        customer();
        customer(char [], char[] , char [], char [],int , int , char , float ,int);
        void deposit(float*);
        void draw(float*);
        void check();
};
```

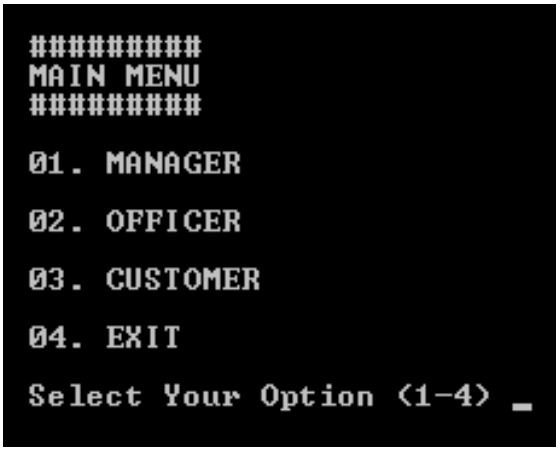
Here

- deposit(): Takes float as a pointer and update it into the balance record.
- draw():Takes float as a pointer and update it into the balance record.
- check(): It calls the check() of officer to view details of the single record.

main():

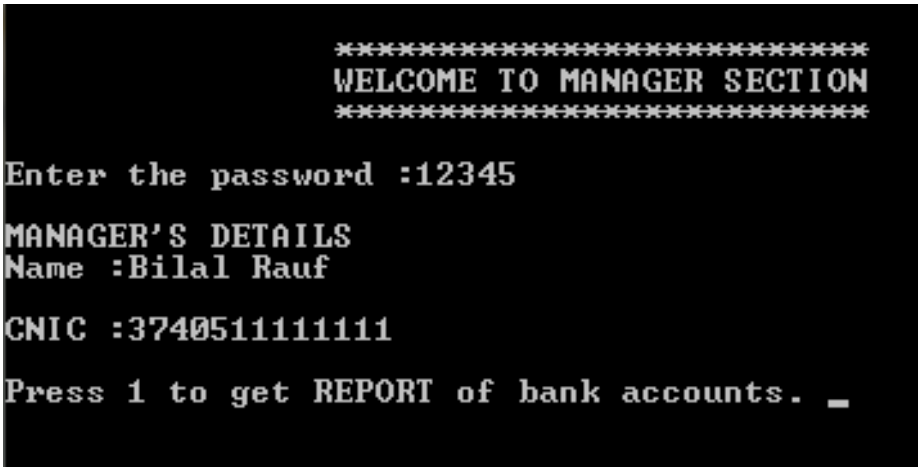
In main(), we have created menu driven program.

Menu will be displayed as:



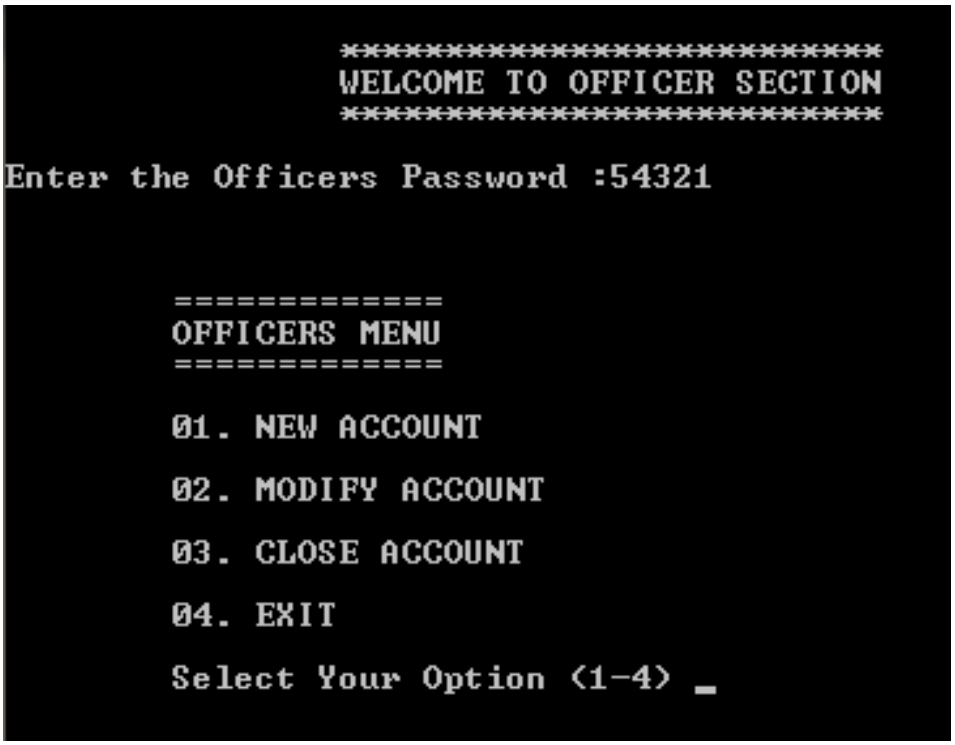
1.Manager:

If we select manager, then it will ask for password. Then after the verification of password, it will further call the function display_all() which further operates by creating object of officer and calling function check() of it.



2. Officer:

If we select Officer , then after the verification of the password, following menu will be opened:

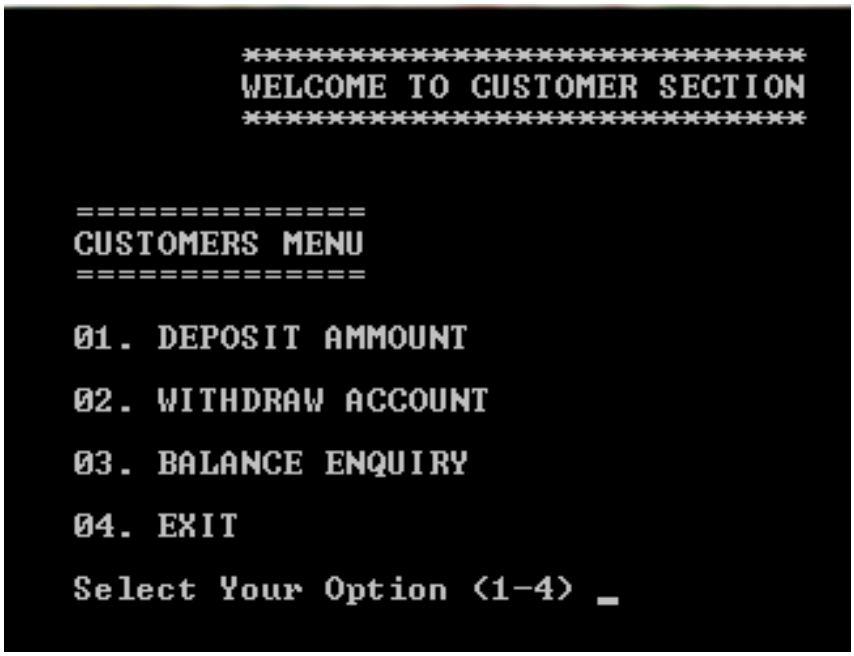


Now if we select new account, it will call write_account() which will make object of officer and saves the information by calling function create() of it.

If we select modify or close account , it will firstly take input for the account no. and then call modify_account() and delete_account() respectively.

3.Customer:

If we select customer, then following menu will be shown:



Now if we select deposit amount or withdraw amount, it will call deposit_withdraw() function which will create objects of customer and call respective functions to deposit or withdraw amount from a particular amount.

If we select balance enquiry, then it will call display_sp() function which further will create objects of class customer which will call function check of class customer.

RESULTS AND CONCLUSIONS:

This program runs fine and have almost no bugs. It can be a useful application for bankers. It can be a handful to customers as well as officers and managers of the bank. So in short, this is a simple but effective application.

Through this project, we could learn that how to use classes, inheritance, polymorphism and pointers in object oriented paradigm.