

# Object Oriented Programming Term Project

**Submitted by:**

**Group members: Awais (01-134211-014)**

**Ali Faizan Mehar (01-134211-010)**

**Submitted to: Sir Imran Ahmed Siddique**

**Object Oriented Programming**

# BANK MANAGEMENT SYSTEM

Department of Computer Science Bahria University Islamabad

## CONTENTS

[Abstract 4](#_bookmark0)

[Introduction 4](#_bookmark1)

[What basically bank management system is? 4](#_bookmark2)

[Necessities 4](#_bookmark3)

Main theme of project Error! Bookmark not defined.

[Details of the project 5](#_bookmark4)

[Explanation of Options 5](#_bookmark5)

[UML Diagram 6](#_bookmark6)

[Source Code 6](#_bookmark6)

[Output 14](#_bookmark7)

[Learning Outcomes 16](#_bookmark8)

## ABSTRACT

In this problem, we’re basically creating the whole scenario of Bank Management system by object oriented programming. In the system we will be providing a user to make their own account by the help of few key words; to modify its own account, check account details, Balance Inquiry, deposit amount, withdraw money transfer amount and to pay utility bills.

## INTRODUCTION:

### WHAT BASICALLY BANK MANAGEMENT SYSTEM IS?

There are many definitions of **Bank Management System (BMS).** In general, bank management refers to the process of managing the Bank’s statutory activity. Bank management is characterized by the specific object of management - financial relations connected with banking activities and other relations, also connected with implementation of management functions in banking.

**Bank** is the place where customers feel the sense of safety for their property. In the bank, customers deposit and withdraw their money. Transaction of money also is a part where customer takes shelter of the bank. Now to keep the belief and trust of customers, there is the positive need for management of the bank, which can handle all this with comfort and ease. Smooth and efficient management affects the satisfaction of the customers and staff members, indirectly. And of course, it **encourages management committee** in taking some needed decision for future enhancement of the bank. Now a days, managing a bank is tedious job upto certain **limit**. So software that reduces the work is essential. Also today’s world is a genuine computer world and is getting faster and faster day-by-day.

## NECESSITIES:

### **Computers** are included in almost all kind of works and jobs everyone come across in the routine. The availability of the software’s for almost every process or every system has taken the world in its top-gear and fastens the **day-to-day** life. So, we have tried our best to develop the **software program** for the Bank Management System where all the tasks to manage the bank system are performed easily and efficiently. It manages all the transactions like new account entry, deposit as well as withdraw entry, transaction of money for various processes, loan entry, **managing bills** cash or cheque, etc.

Thus, above features of this software will save transaction time and therefore increase the efficiency of the system. Requirements definition and management is recognized as a necessary step in the delivery of successful system s and software projects, **discipline** is also required by standards, regulations, and quality improvement initiatives. Creating and managing requirements is a challenge of **IT**, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. Organization need to effectively define and manage requirements to ensure they are meeting needs of the customer, while **proving compliance** and staying on the schedule and within budge. The impact of a poorly expressed requirement can bring a business out of compliance or even cause injury or death. Requirements definition and management is an activity that can deliver a high, fast return on investment**.**

**Thus, considering above necessities, the software for bank management has became necessary which would be useful in managing the bank more efficiently.**

## INVESTIGATION AND STRENGTH:

### The Bank Management is determined by the following characteristics:

* Management expertise in strategic analysis, **planning**, policy development and management functions.
* **Quality of planning.**

### Risk management (credit, interest rate and currency risks).

* Liquidity management.

### Management of human resources.

* Creation of control systems: **audit and internal audit** , monitoring of profitability and risks liquidity;
* Integrated automation of workflow, accounting, current analysis and control, strategic planning.

## DETAILS OF THE PROJECT:

### EXPLANATION OF OPTIONS:

The project is divided into 9 modules.

**Create new account:** It will led the user to make a new account if he/she is new in this.

## **Modify Account:** It will have two more branches to whether to modify your account title or account type.

**Check Account details:** For details the system will provide account number, account title, account type and its current balance.

## **Balance Inquiry:** This will only provide current balance after if any depositing and withdrawel have been made.

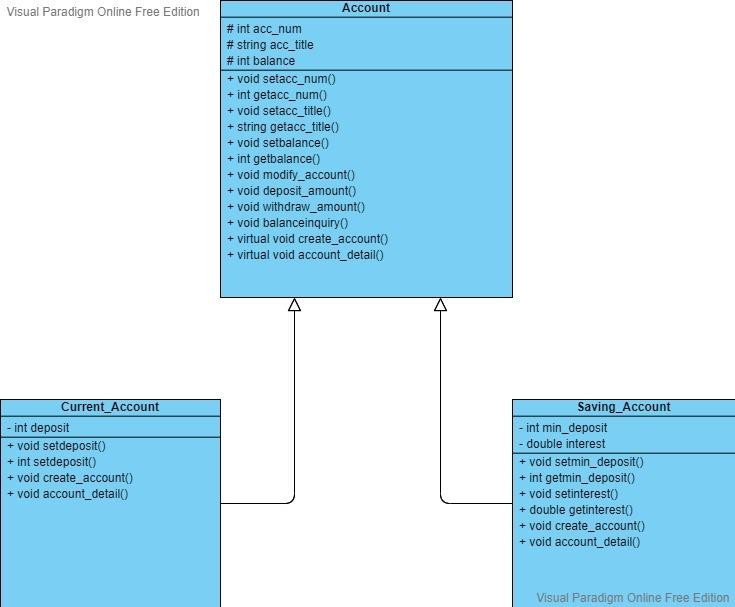
**Deposit Amount:** Give user, to deposit its money.

## **Withdraw:** It will led the user to withdraw the money (To withdraw is to take something back or remove yourself from a situation).

**Pay Utility Bills:** Led user to pay bills.

## **Exit:** And last is to exit, if you want don’t want to use the system anymore.

### UML DIAGRAM



### SOURCE CODE:

#include<iostream> #include<conio.h> #include<string>

using namespace std;

class Account //base class

{

protected:

int acc\_num; string acc\_title; int balance;

public:

Account(int an = 0, string n = " ", int b = 0) //parameterized constructor

{

acc\_num = an; acc\_title = n; balance = b;

}

//set & get functions of class

void setacc\_num(int an)

{

acc\_num = an;

}

int getacc\_num()

{

return acc\_num;

}

void setacc\_title(string n)

{

acc\_title = n;

}

string getacc\_title()

{

return acc\_title;

}

void setbalance(int b)

{

balance = b;

}

int getbalance()

{

return balance;

}

//function to modify account void modify\_account()

{

modification

cout << "\n \*\*Previous Detail\*\* \n"; //show previous detail before account\_detail();

char choice;

cout << "\n\n \*\*\*\*Modify Account\*\*\*\*\n\n" << endl; cout << " Do you want to modify Account Title (y/n): "; cin >> choice;

if (choice == 'y')

{

cout << " Enter New Account Title: "; cin >> acc\_title;

cout << "\n \*\*\*Account Title Updated\*\*\*\n" << endl;

}

cout << " Do you want to modify Account Balance (y/n): "; cin >> choice;

if (choice == 'y')

{

cout << "\n Do you want to deposit cash or withdraw cash (d/w): "; cin >> choice;

if (choice == 'd')

{

}

else

{

}

}

int d;

cout << " Enter Deposit Amount: "; cin >> d;

deposit\_amount(d);

cout << "\n \*\*\*Balance Updated\*\*\*\n" << endl;

int w;

cout << " Enter Withdraw Amount: "; cin >> w;

withdraw\_amount(w);

cout << "\n \*\*\*Balance Updated\*\*\*\n" << endl;

modification

}

cout << "\n \*\*New Detail\*\* \n"; //sho new detail after account\_detail();

//function to deposit amount void deposit\_amount(int deposit)

{

balance += deposit;

}

//function to withdraw amount void withdraw\_amount(int withdraw)

{

balance -= withdraw;

}

//function to check balance void balanceinquiry()

{

cout << "\n Available Balance in Account: " << balance << endl;

}

//virtual functions

virtual void create\_account() = 0; virtual void account\_detail() = 0;

};

class Current\_Account : public Account //derived class

{

int deposit;

public:

Current\_Account(int an = 0, string n = " ", int b = 0, int d = 0) :Account(an, n, b) //parameterized constructor

{

deposit = d;

}

//set & get functions

void setdeposit(int d)

{

deposit = d;

}

int getdeposit()

{

return deposit;

}

//funtion to create new account void create\_account()

{

cout << " Enter Account Number: "; cin >> acc\_num;

cout << " Enter Account Title: "; cin >> acc\_title;

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

balance += deposit;

cout << "\n\n \*\*\*Account Created\*\*\*\n\n" << endl;

}

//function to check account detail void account\_detail()

{

cout << "\n \*\*\*Current Account Detail\*\*\*\n" << endl; cout << " Account Number: " << acc\_num << endl; cout << " Account Title: " << acc\_title << endl; cout << " Balance: " << balance << endl;

}

};

class Saving\_Account : public Account

{

int min\_deposit; double interest;

public:

Saving\_Account(int an = 0, string n = " ", int b = 0, int md = 0, int i = 0)

:Account(an, n, b) //parameterized constructor

{

min\_deposit = md; interest = i;

}

//set & get functions

void setmin\_deposit(int md)

{

min\_deposit = md;

}

int getmin\_deposit()

{

return min\_deposit;

}

void setinterest(int i)

{

interest = i;

}

double getinterest()

{

return interest;

}

//funtion to create new account void create\_account()

{

cout << " Enter Account Number: "; cin >> acc\_num;

cout << " Enter Account Title: "; cin >> acc\_title;

cout << " Enter Initial Deposit Amount (Not less than 100k): "; cin >> min\_deposit;

balance += min\_deposit; interest = 0.1\*balance;

cout << "\n\n \*\*\*Account Created\*\*\*\n\n" << endl;

}

//function to check account detail void account\_detail()

{

cout << "\n \*\*\*Saving Account Detail\*\*\*\n" << endl; cout << " Account Number: " << acc\_num << endl; cout << " Account Title: " << acc\_title << endl; cout << " Balance: " << balance << endl;

interest = 0.1\*balance;

cout << " Annual Interest: " << interest << endl;

}

};

//functions prototype

bool find\_acc(Account \*p[], int size, int acc); int index(Account \*p[], int size, int acc);

int main()

{

Account \* ptr[100]; //pointer to base class

//storing some data in array

ptr[0] = new Current\_Account(999, "John", 10000, 5500); ptr[1] = new Current\_Account(888, "Ali", 45500); ptr[2] = new Saving\_Account(777, "Abdullah", 150000); ptr[3] = new Saving\_Account(666, "Watson", 100000);

int i = 4; int option; do

{

//main menu

cout << "\n \n"; cout << "\tBANK MANAGEMENT SYSTEM";

cout << "\n \n"; cout << "\n (1) Create New Account";

cout << "\n (2) Modify Account";

cout << "\n (3) Check Account Detail";

cout << "\n (4) Balance Inquiry"; cout << "\n (5) Deposit Amount"; cout << "\n (6) Withdraw Amount"; cout << "\n (7) Pay Utility Bills"; cout << "\n (8) Exit";

cout << "\n\n Select your option: "; cin >> option;

int n;

switch (option)

{

case 1: //create new account

cout << "\n Choose Account Type (c/s): "; char type;

cin >> type;

if (type == 'c')

{

}

else

{

}

break;

ptr[i] = new Current\_Account; //dynamic binding ptr[i]->create\_account();

i++;

ptr[i] = new Saving\_Account; //dynamic binding ptr[i]->create\_account();

i++;

case 2: //modify account

cout << "\n Enter Account Number: "; cin >> n;

if (find\_acc(ptr, i, n))

{

}

else

{

}

break;

ptr[index(ptr, i, n)]->modify\_account();

cout << "\n \*\*\*Account Not Found\*\*\*\n" << endl;

case 3: //check account details

cout << "\n Enter Account Number: "; cin >> n;

if (find\_acc(ptr, i, n))

{

}

else

{

ptr[index(ptr, i, n)]->account\_detail();

cout << "\n \*\*\*Account Not Found\*\*\*\n" << endl;

}

break;

case 4: //Balance Inquiry

cout << "\n Enter Account Number: "; cin >> n;

if (find\_acc(ptr, i, n))

{

}

else

{

}

break;

ptr[index(ptr, i, n)]->balanceinquiry();

cout << "\n \*\*\*Account Not Found\*\*\*\n" << endl;

case 5: //deposit amount

cout << "\n Enter Account Number: "; cin >> n;

if (find\_acc(ptr, i, n))

{

cout << " Enter Deposit Amount: "; int dep;

cin >> dep;

ptr[index(ptr, i, n)]->deposit\_amount(dep);

cout << "\n New Detail after Deposit "; ptr[index(ptr, i, n)]->account\_detail();

}

else

{

}

break;

cout << "\n \*\*\*Account Not Found\*\*\*\n" << endl;

case 6: //withdraw amount

cout << "\n Enter Account Number: "; cin >> n;

if (find\_acc(ptr, i, n))

{

cout << " Enter Withdraw Amount: "; int wid;

cin >> wid;

ptr[index(ptr, i, n)]->withdraw\_amount(wid);

cout << "\n New Detail after Withdraw "; ptr[index(ptr, i, n)]->account\_detail();

}

else

{

cout << "\n \*\*\*Account Not Found\*\*\*\n" << endl;

}

break;

case 7: //pay utility bills

cout << "\n Enter Bill Number: "; int billnum;

cin >> billnum;

cout << " Enter Bill Amount: "; int billamt;

cin >> billamt;

cout << "\n \*\*\*Bill Payed\*\*\*" << endl;

}

} while (option != 8); delete ptr[100];

\_getch(); return 0;

}

//function definitions

//function to find account record in array bool find\_acc(Account \*p[], int size, int acc)

{

bool found = false;

for (int i = 0; i <= size; i++)

{

if (p[i]->getacc\_num() == acc)

{

found = true; break;

}

}

return found;

}

//function to fetch index number of required account in an array int index(Account \*p[], int size, int acc)

{

for (int i = 0; i <= size; i++)

{

if (p[i]->getacc\_num() == acc)

{

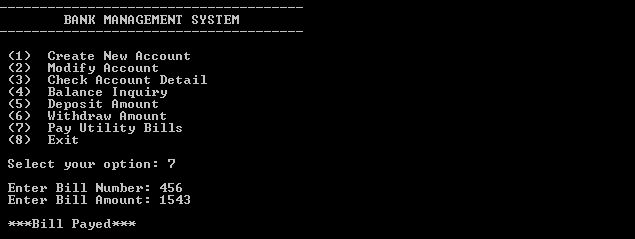
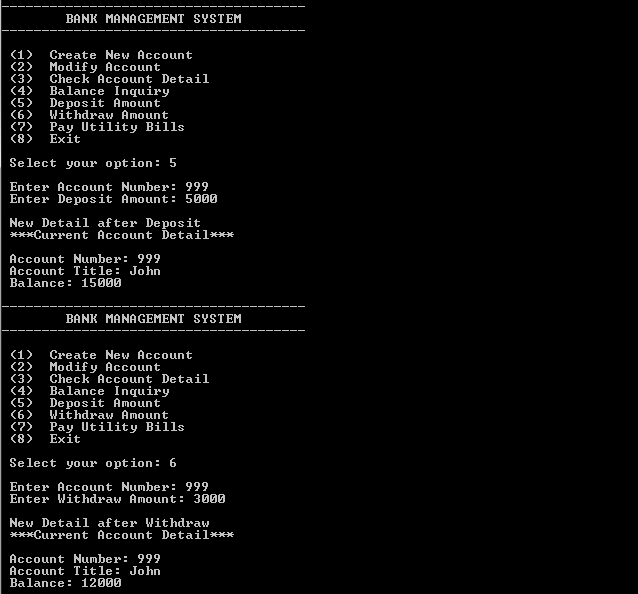
return i; break;

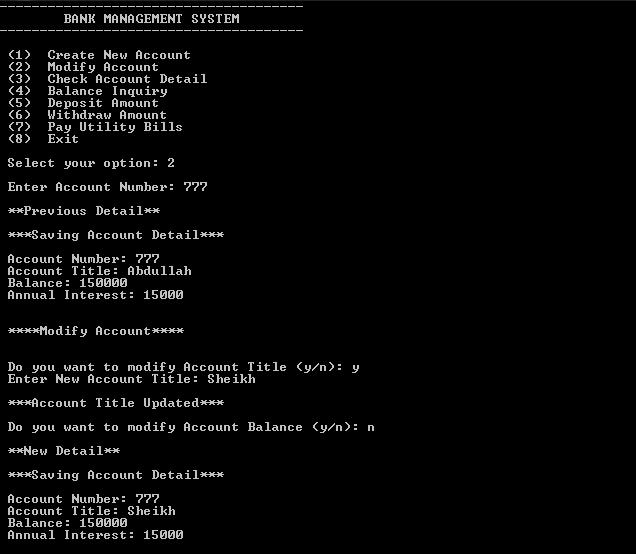
}

}

}

### OUTPUT:





## LEARNING OUTCOMES:

### This activity brings us to abstract thinking, originality in analysis to formulate suitable programming models of the activity. The activity involves us to creative use of programming principles and research- based knowledge in novel ways.

1. In this project we get to learn about management system of bank and by exploring how the other tools of object oriented programming work and got there well explained idea too.
2. Get to know how classes would work in big projects and how functions ease our work by sharing different task to different group members.

**THE END**