**C# Assignment-2**

**1,2,3,4**

using System;

using System.Collections.Generic;

using System.Collections;

using System.Linq;

using System.Text;

namespace Asignment2

{

public abstract class Account

{

int accountNo;

int accountBalance;

string accountPassword;

public string bankName;

public int \_accountNo { set; get; }

public int \_accountBalance { set; get; }

public string \_accountPassword { set; get; }

public Account() { }

public Account(int accountNo, int accountBalance, string accountPassword, string bankName)

{

this.accountNo = accountNo;

this.accountBalance = accountBalance;

this.accountPassword = accountPassword;

this.bankName = bankName;

}

public abstract void withdraw(long amount);

virtual public void displayAccount()

{

Console.WriteLine("accountNo: " + accountNo);

Console.WriteLine("accountBalance: " + accountBalance);

Console.WriteLine("accountPassword: " + accountPassword);

Console.WriteLine("bankName: " + bankName);

}

}

class SavingsAccount : Account

{

long minimumBalance;

public long MinimumBalance { get => minimumBalance; set => minimumBalance = value; }

public override void displayAccount()

{

Console.WriteLine("The minimum balance is:" + MinimumBalance);

}

public override void withdraw(long amount)

{

Console.WriteLine("Enter the amount to be withdrawn:");

amount = long.Parse(Console.ReadLine());

Console.WriteLine("the balance Amount after withdrawn is :" + (\_accountBalance - amount));

if (amount > \_accountBalance)

{

throw new InsufficientFundException("InsufficientFundException");

}

if (amount < 0)

{

InvalidAmountException e1 = new InvalidAmountException("negative number");

throw e1;

}

}

}

class CurrentAccount : Account

{

long overdraftlimitAmount;

public long OverdraftlimitAmount { get => overdraftlimitAmount; set => overdraftlimitAmount = value; }

public override void displayAccount()

{

Console.WriteLine("The Overdraftlimit Amount is:" + OverdraftlimitAmount);

}

public override void withdraw(long amount)

{

Console.WriteLine("Enter the amount to be withdrawn:");

amount = long.Parse(Console.ReadLine());

Console.WriteLine("the overdraft limit Amount after withdrawn is :" + OverdraftlimitAmount);

}

}

public interface ATM

{

void withdraw(int accountNumber, double amount);

void changePassword(int accountNumber, String oldPassword, String newPassword);

void checkBalance();

}

class SbiAtm : ATM

{

public void withdraw(int accountNumber, double amount) { }

public void changePassword(int accountNumber, String oldPassword, String newPassword) { }

public void checkBalance() { }

}

class IciciAtm : ATM

{

public void withdraw(int accountNumber, double amount) { }

public void changePassword(int accountNumber, String oldPassword, String newPassword) { }

public void checkBalance() { }

}

class InvalidAmountException : Exception

{

public InvalidAmountException() { }

public InvalidAmountException(string message) : base(message) { }

}

class InsufficientFundException : Exception

{

public InsufficientFundException() { }

public InsufficientFundException(string message) : base(message)

{

}

}

public interface AccountDao

{

void addAnAccount(Account account);

void withdraw(int accountNumber, double amount);

void checkBalance();

void changePassword(int accountNumber, String oldPassword, String newPassword);

// void List<Account> viewAllAccounts();

void getAccountDetails(int accountNumber);

}

class InMemoryAccountDaoImpl

{

public void addAnAccount(Account account)

{

ArrayList al = new ArrayList();

al.Add(account);

}

public void withdraw(int accountNumber, double amount)

{

ArrayList al2 = new ArrayList();

}

}

class Program

{

static void Main(string[] args)

{

SavingsAccount s = new SavingsAccount();

s.MinimumBalance = 5000;

s.displayAccount();

CurrentAccount c = new CurrentAccount();

c.OverdraftlimitAmount = 10000;

c.displayAccount();

}

}

}

**5**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApp1

{

class Bank

{

int accountNo;

double accountBalance;

string accountPassword;

public static string bankName = "SBI";

public int \_number

{

get { return accountNo; }

set { accountNo = value; }

}

public double \_balance

{

get { return accountBalance; }

set { accountBalance = value; }

}

public string \_password

{

get { return accountPassword; }

set { accountPassword = value; }

}

public string \_name

{

get { return bankName; }

set { bankName = value; }

}

public Bank()

{

}

public Bank(int accountNo, double accountBalance, string accountPassword)

{

this.accountNo = accountNo;

this.accountBalance = accountBalance;

this.accountPassword = accountPassword;

}

}

class info

{

public void display(Bank b1)

{

Console.WriteLine(b1.\_number);

Console.WriteLine(b1.\_balance);

Console.WriteLine(b1.\_password);

Console.WriteLine(Bank.bankName);

}

}

class InvalidAmountException : Exception

{

public InvalidAmountException() { }

public InvalidAmountException(double balance) : base(String.Format("Amount less than minimum balance,current balance is {0}", balance))

{

}

}

class InsufficientFundException : Exception

{

public InsufficientFundException() { }

public InsufficientFundException(double balance) : base(String.Format("Amount less than minimum funds in the current Account"))

{

}

}

interface ATM

{

void withdraw(int accountNo, double amount);

string changePassword(int accountNo, string oldPassword, string newPassword);

double checkBalance();

}

class SbiAtm : ATM

{

int accountNumber;

double balance;

//string pass;

//SavingsAccount sv = new SavingsAccount(accountNumber);

public int AccountNumber { get => accountNumber; set => accountNumber = value; }

public double Balance { get => balance; set => balance = value; }

//public string Pass { get => pass; set => pass = value; }

public SbiAtm()

{

}

public SbiAtm(int acc, double amt)

{

this.AccountNumber = acc;

this.balance = amt;

}

public void withdraw(int accountNo, double amount)

{

if (this.AccountNumber != accountNo || amount >= this.balance)

{

Console.WriteLine("Insufficient Funds");

}

else

{

Console.WriteLine(this.balance - amount);

}

}

public string changePassword(int accountNo, string oldPassword, string newPassword)

{

//string p = "Password Change is not possible";

if (accountNo == this.AccountNumber)

{

return newPassword;

}

else

{

return oldPassword;

}

}

public double checkBalance()

{

return this.Balance;

}

}

class IciciAtm : ATM

{

int accountNumber;

double balance;

//string pass;

public int AccountNumber { get => accountNumber; set => accountNumber = value; }

public double Balance { get => balance; set => balance = value; }

//public string Pass { get => pass; set => pass = value; }

public IciciAtm()

{

}

public IciciAtm(int acc, double amt)

{

this.AccountNumber = acc;

this.balance = amt;

}

public void withdraw(int accountNo, double amount)

{

if (this.AccountNumber != accountNo || amount >= this.balance)

{

Console.WriteLine("Insufficient Funds");

}

else

{

Console.WriteLine(this.balance - amount);

}

}

public string changePassword(int accountNo, string oldPassword, string newPassword)

{

//string p = "Password Change is not possible";

if (accountNo == this.AccountNumber)

{

return newPassword;

}

else

{

return oldPassword;

}

}

public double checkBalance()

{

return this.Balance;

}

}

abstract class Account

{

int accountNo;

double accountBalance;

string accountPassword;

double amount;

public static string bankName = "SBI";

public int \_number

{

get { return accountNo; }

set { accountNo = value; }

}

public double \_balance

{

get { return accountBalance; }

set { accountBalance = value; }

}

public string \_password

{

get { return accountPassword; }

set { accountPassword = value; }

}

public string \_name

{

get { return bankName; }

set { bankName = value; }

}

public Account()

{

}

//public abstract double withdraw(double amount);

public Account(int accountNo, double accountBalance, string accountPassword)

{

this.accountNo = accountNo;

this.accountBalance = accountBalance;

this.accountPassword = accountPassword;

}

public virtual void display()

{

Console.WriteLine(this.\_number);

Console.WriteLine(this.\_balance);

Console.WriteLine(this.\_password);

Console.WriteLine(Account.bankName);

}

public abstract double withdraw(double amount);

}

class SavingsAccount : Account

{

double minBalance;

public double MinBalance { get => minBalance; set => minBalance = value; }

public SavingsAccount(int accountNo, double accountBalance, string accountPassword, double minBalance) : base(accountNo, accountBalance, accountPassword)

{

this.MinBalance = minBalance;

}

public SavingsAccount()

{

}

public override void display()

{

base.display();

Console.WriteLine(this.MinBalance);

}

public override double withdraw(double amount)

{

return this.\_balance - amount;

}

}

class CurrentAccount : Account

{

double overdraftLimit;

public double OverdraftLimit { get => overdraftLimit; set => overdraftLimit = value; }

public CurrentAccount(int accountNo, double accountBalance, string accountPassword, double overdraftLimit) : base(accountNo, accountBalance, accountPassword)

{

this.OverdraftLimit = overdraftLimit;

}

public CurrentAccount()

{

}

public override void display()

{

base.display();

Console.WriteLine(this.OverdraftLimit);

}

public override double withdraw(double amount)

{

return this.\_balance - amount;

}

}

class Program

{

static void Main(string[] args)

{

Bank b = new Bank();

Bank b2 = new Bank(245678343, 55.90, "pratyusha@123");

Bank.bankName = "SBI";

b.\_number = 356778634;

b.\_balance = 50045.89;

b.\_password = "pratyusha@123";

info i1 = new info();

i1.display(b);

i1.display(b2);

SavingsAccount s = new SavingsAccount(245673451, 50000, "abc@123", 500);

CurrentAccount c = new CurrentAccount(567834291, 55000, "xyz@456", 10000);

//s.MinBalance = 100;

//Console.WriteLine("SavingsAccount Details are:");

//s.display();

//Console.WriteLine("Savings Account Balance is:");

//Console.WriteLine(s.withdraw(200));

//Console.WriteLine("CurrentAccount Details are:");

//c.display();

//Console.WriteLine("Current Account Balance is:");

//Console.WriteLine(c.withdraw(3000));

//ATM i;

//i = new SbiAtm(344010,8000);

//SbiAtm sb1 = new SbiAtm(1000,12000);

//sb.AccountNumber = 12890;

////sb.withdraw(10000, 110000);

//sb1.withdraw(1000, 10000);

//Console.WriteLine(sb.changePassword(1234, "abcd", "xyz"));

//Console.WriteLine("Balance:"+sb.checkBalance());

//Console.WriteLine("Interface Balance:"+i.checkBalance());

//Console.WriteLine(sb1.changePassword(5678, "abcd", "xyz"));

SavingsAccount sv = new SavingsAccount(123, 150000, "abc@123", 5000);

SavingsAccount sv1 = new SavingsAccount(789, 50000, "xyz#45", 2500);

InMemoryAccountDaoImpl i = new InMemoryAccountDaoImpl();

i.AddanAccount(sv);

i.AddanAccount(sv1);

i.withdraw(123, 5000);

i.checkBalance();

try

{

if (s.withdraw(1000) < 0)

{

throw new InvalidAmountException();

}

}

catch (InvalidAmountException e)

{

Console.WriteLine(e.Message);

}

try

{

if (c.withdraw(100000) < 0)

{

throw new InsufficientFundException();

}

}

catch (InsufficientFundException e)

{

Console.WriteLine(e.Message);

}

}

}

class global

{

public static List<SavingsAccount> l = new List<SavingsAccount>();

}

interface AccountDao

{

void AddanAccount(SavingsAccount account);

void withdraw(int accountNo, double amount);

void checkBalance();

void changePassword(int acntNumber, String oldPassword, String newPassword);

}

class InMemoryAccountDaoImpl : AccountDao

{

public void AddanAccount(SavingsAccount account)

{

global.l.Add(account);

}

public void withdraw(int accountNumber, double amount)

{

foreach (var i in global.l)

{

if (i.\_number == accountNumber)

{

if (i.\_balance >= amount)

{

double t = i.\_balance - amount;

Console.WriteLine("Balance in the account is:" + t);

}

else

{

Console.WriteLine("Insufficient Funds");

}

}

}

}

public void checkBalance()

{

foreach (var i in global.l)

{

Console.WriteLine($"Balance for {i}" + i.\_balance);

}

}

public void changePassword(int acntNumber, String oldPassword, String newPassword)

{

foreach (var i in global.l)

{

if (i.\_number == acntNumber)

{

if (i.\_password == oldPassword)

{

i.\_password = newPassword;

Console.WriteLine("The password is updated");

}

else

{

Console.WriteLine("The password mismatch");

}

}

}

}

public List<SavingsAccount> viewAllAccounts()

{

return global.l;

}

public void getAccountDetails(int accountNumber)

{

foreach (var i in global.l)

{

if (i.\_number == accountNumber)

{

Console.WriteLine(i.\_number + "," + i.MinBalance + "," + i.\_balance + "," + i.\_password);

}}}}

**6**

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Text;

namespace Asignment2\_6thq\_ado\_

{

class Program

{

class InvalidAmountException : Exception

{

public InvalidAmountException(string message) : base(message)

{

}

}

class InsufficientFundException : Exception

{

public InsufficientFundException(string message) : base(message)

{

}

}

interface ATM

{

abstract public void withdraw(int accountNumber, double amount);

abstract public bool changePassword(int accountNumber, String oldPassword, String newPassword);

abstract public void checkBalance();

}

class SbiAtm : ATM

{

public void withdraw(int accountNumber, double amount)

{

}

public bool changePassword(int accountNumber, String oldPassword, String newPassword)

{

return true;

}

public void checkBalance()

{

}

}

class IciciAtm : ATM

{

public void withdraw(int accountNumber, double amount)

{

}

public bool changePassword(int accountNumber, String oldPassword, String newPassword)

{

return true;

}

public void checkBalance()

{

}

}

abstract class Account

{

int accountNo;

double accountBalance;

string accountPassword;

static string bankName = "SBI BANK";

public static string BankName { get => bankName; set => bankName = value; }

public int \_accountNo

{

get => accountNo;

set => accountNo = value;

}

public double \_accountBalance

{

get => accountBalance;

set => accountBalance = value;

}

public string \_accountPassword

{

get => accountPassword;

set => accountPassword = value;

}

public Account()

{

}

public Account(int accountNo, double accountBalance, string accountPassword)

{

this.accountNo = accountNo;

this.accountBalance = accountBalance;

this.accountPassword = accountPassword;

}

virtual public void displayAccount()

{

Console.WriteLine("Account Number : {0}", accountNo);

Console.WriteLine("Account Balance : {0}", accountBalance);

Console.WriteLine("Bank Name : {0}", BankName);

}

abstract public void withdraw(double WithdrawAmount);

}

class SavingsAccount : Account

{

public static double minimumBalance = 500;

override public void displayAccount()

{

Console.WriteLine("Account Number : {0}", \_accountNo);

Console.WriteLine("Account Balance : {0}", \_accountBalance);

Console.WriteLine("Minumum Balance : {0}", minimumBalance);

Console.WriteLine("Bank Name : {0}", BankName);

}

public SavingsAccount()

{

}

public SavingsAccount(int accountNo, double accountBalance, string accountPassword) : base(accountNo, accountBalance, accountPassword)

{

}

override public void withdraw(double WithdrawAmount)

{

try

{

if (WithdrawAmount < 1)

{

throw new InvalidAmountException("Invalid Amount");

}

else if (\_accountBalance < WithdrawAmount)

{

throw new InsufficientFundException("Insufficient Balance");

}

else

{

\_accountBalance -= WithdrawAmount;

Console.WriteLine("Transaction Successfull");

Console.WriteLine("Debited : {0}", WithdrawAmount);

this.displayAccount();

}

}

catch (Exception e)

{

Console.WriteLine("Transaction Failed");

Console.WriteLine(e.Message);

}

}

}

class CurrentAccount : Account

{

public static double overdraftLimitAmount = 20000;

override public void displayAccount()

{

Console.WriteLine("Account Number : {0}", \_accountNo);

Console.WriteLine("Account Balance : {0}", \_accountBalance);

Console.WriteLine("overdraftLimitAmount: {0}", overdraftLimitAmount);

Console.WriteLine("Bank Name : {0}", BankName);

}

public CurrentAccount()

{

}

public CurrentAccount(int accountNo, double accountBalance, string accountPassword) : base(accountNo, accountBalance, accountPassword)

{

}

override public void withdraw(double WithdrawAmount)

{

try

{

if (WithdrawAmount < 1)

{

throw new InvalidAmountException("Invalid Amount");

}

else if (\_accountBalance < WithdrawAmount)

{

throw new InsufficientFundException("Insufficient Balance");

}

else

{

\_accountBalance -= WithdrawAmount;

Console.WriteLine("Current Balance : {0}", \_accountBalance);

Console.WriteLine("overdraftLimitAmount : {0}", overdraftLimitAmount);

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

interface AccountDao

{

abstract void addAnAccount(Account account);

abstract void withdraw(int accountNumber, double amount);

abstract void checkBalance(int accountNumber);

abstract void changePassword(int accountNumber, String oldPassword, String newPassword);

abstract void viewAllAccounts();

abstract void getAccountDetails(int accountNumber);

}

class InMemoryAccountDaoImpl : AccountDao

{

List<Account> AllAccounts = new List<Account>();

public string connectionString = "Data Source=DESKTOP-HH0PVJC;Initial Catalog = AccountInfo; Integrated Security = true";

public void addAnAccount(Account account)

{

// AllAccounts.Add(account);

using (SqlConnection con = new SqlConnection(this.connectionString))

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("CreateSavingsAccount", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@accountNo", account.\_accountNo);

cmd.Parameters.AddWithValue("@accountBalance", account.\_accountBalance);

cmd.Parameters.AddWithValue("@accountPassword", account.\_accountPassword);

int result = cmd.ExecuteNonQuery();

if (result > 0)

{

Console.WriteLine("Transaction Successfull");

Console.WriteLine("Account Added");

}

else

{

Console.WriteLine("Transaction Failed");

}

con.Close();

}

catch (Exception e)

{

Console.WriteLine("Transaction Failed");

Console.WriteLine(e.Message);

}

}

}

public void withdraw(int accountNumber, double amount)

{

using (SqlConnection con = new SqlConnection(this.connectionString))

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("SP\_DebitAmount", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@accountNo", accountNumber);

cmd.Parameters.AddWithValue("@amount", amount);

int result = cmd.ExecuteNonQuery();

if (result > 0)

{

Console.WriteLine("Transaction Successfull");

Console.WriteLine("Amount Debited");

}

else

{

Console.WriteLine("Transaction Failed");

}

con.Close();

}

catch (Exception e)

{

Console.WriteLine("Transaction Failed");

Console.WriteLine(e.Message);

}

}

}

public void checkBalance(int accountNumber)

{

using (SqlConnection con = new SqlConnection(this.connectionString))

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("SP\_AccountBalance", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@accountNo", accountNumber);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

Console.WriteLine("Transaction Successfull");

Console.WriteLine("Account Balance : {0}", dr[0]);

}

con.Close();

}

catch (Exception e)

{

Console.WriteLine("Transaction Failed");

Console.WriteLine(e.Message);

}

}

}

public void changePassword(int accountNumber, String oldPassword, String newPassword)

{

using (SqlConnection con = new SqlConnection(this.connectionString))

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("SP\_ChangePassword", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@accountNo", accountNumber);

cmd.Parameters.AddWithValue("@accountOldPassword", oldPassword);

cmd.Parameters.AddWithValue("@accountNewPassword", newPassword);

cmd.Parameters.Add("@status", SqlDbType.Int).Direction = ParameterDirection.Output;

int result = cmd.ExecuteNonQuery();

if (result > 0)

{

int status = Convert.ToInt32(cmd.Parameters["@status"].Value);

if (status == 1)

{

Console.WriteLine("Transaction Successfull -- Passwordchanged");

}

else

{

Console.WriteLine("Transaction UnSuccessfull -- Invalid Old Password");

}

}

else

{

Console.WriteLine("Transaction Failed");

}

con.Close();

}

catch (Exception e)

{

Console.WriteLine("Transaction Failed");

Console.WriteLine(e.Message);

}

}

}

public void viewAllAccounts()

{

Console.WriteLine("==== All ACCOUNT DETAILS ====");

using (SqlConnection con = new SqlConnection(connectionString))

{

con.Open();

SqlCommand cmd = new SqlCommand("Select \* from Account", con);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

Console.WriteLine("Account Number : {0}", dr[0]);

Console.WriteLine("Account Balance : {0}", dr[1]);

Console.WriteLine("Account Password : {0}", dr[2]);

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

}

con.Close();

}

Console.WriteLine("==== =================== ====");

}

public void getAccountDetails(int accountNumber)

{

using (SqlConnection con = new SqlConnection(this.connectionString))

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("SP\_GetAccountDetailsByID", con);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@accountNo", accountNumber);

SqlDataReader dr = cmd.ExecuteReader();

while (dr.Read())

{

Console.WriteLine("==== ACCOUNT DETAILS ====");

Console.WriteLine("Account Number : {0}", dr[0]);

Console.WriteLine("Account Balance : {0}", dr[1]);

Console.WriteLine("Account Password : {0}", dr[2]);

Console.WriteLine("==== =============== ====");

}

con.Close();

}

catch (Exception e)

{

Console.WriteLine("Transaction Failed");

Console.WriteLine(e.Message);

}

}

}

}

public static void Main()

{

//Account A0 = new Account(0, 2648, "SBI");

//Account.BankName = "SBI";

//A0.displayAccount();

//Account SA = new SavingsAccount(1,2000, "OGSS");

//SA.displayAccount();

//SA.withdraw(-500);

//SA.displayAccount();

//Console.WriteLine("#########################################################");

//Console.WriteLine("#########################################################");

//Account CA = new CurrentAccount(2, 2002, "SBI");

//CA.displayAccount();

//CA.withdraw(40000);

//CA.displayAccount();

InMemoryAccountDaoImpl InMemoryAccountDaoImplOnj = new InMemoryAccountDaoImpl();

bool shallContinue = true;

int choosen;

while (shallContinue)

{

Console.WriteLine("==== 1:addAnAccount -- 2:withdraw -- 3:checkBalance -- 4:changePassword -- 5:viewAllAccounts -- 6:getAccountDetails -- 7:Exit ==== ");

Console.Write("Enter Option : ");

choosen = Convert.ToInt32(Console.ReadLine());

switch (choosen)

{

case 1:

Account tempAccount = new SavingsAccount();

//Console.Write("Enter Type (S/C) : ");

Console.Write("Enter Account Number : ");

tempAccount.\_accountNo = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Balance : ");

tempAccount.\_accountBalance = Convert.ToDouble(Console.ReadLine());

Console.Write("Enter Password : ");

tempAccount.\_accountPassword = Convert.ToString(Console.ReadLine());

InMemoryAccountDaoImplOnj.addAnAccount(tempAccount);

break;

case 2:

Console.Write("Enter Account Number : ");

int accountNumber = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Amount : ");

double amount = Convert.ToDouble(Console.ReadLine());

InMemoryAccountDaoImplOnj.withdraw(accountNumber, amount);

break;

case 3:

Console.Write("Enter Account Number : ");

int accountNumber2 = Convert.ToInt32(Console.ReadLine());

InMemoryAccountDaoImplOnj.checkBalance(accountNumber2);

break;

case 4:

Console.Write("Enter Account Number : ");

int accountNumber4 = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Old Password : ");

string OldaccountPassword = Convert.ToString(Console.ReadLine());

Console.Write("Enter New Password : ");

string NewaccountPassword = Convert.ToString(Console.ReadLine());

InMemoryAccountDaoImplOnj.changePassword(accountNumber4, OldaccountPassword, NewaccountPassword);

break;

case 5:

InMemoryAccountDaoImplOnj.viewAllAccounts();

break;

case 6:

Console.Write("Enter Account Number : ");

int accountNumber6 = Convert.ToInt32(Console.ReadLine());

InMemoryAccountDaoImplOnj.getAccountDetails(accountNumber6);

break;

case 7:

shallContinue = false;

break;

default:

Console.WriteLine("Please Enter a Valid Option");

break;

}

}

}

}

}

**SQL part of 6th question**

create database AccountInfo

use AccountInfo;

create table Account

(

acno int,

minb int,

amt int,

acT varchar(30),

psw varchar(10)

);

select \* from Account;