Main class: abstract

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Banking

{

internal abstract class Account: IAccount

{

public string AccountNumber { get; set; }

public string AccountHolder { get; set; }

public double Balance { get; protected set; }

public Account(string accountNumber, string accountHolder)

{

AccountNumber = accountNumber;

AccountHolder = accountHolder;

Balance = 0;

}

public void Deposit(double amount)

{

Balance += amount;

}

public abstract void Withdraw(double amount); //abstract method declaration

public void Transfer(IAccount toAccount, double amount)

{

Withdraw(amount); //withdraw amount from current account

toAccount.Deposit(amount); //deposit in the to-account

}

public virtual void PrintAccountDetails() //virtual method implementations that can be overriden

{

Console.WriteLine($"Account Number: {AccountNumber}");

Console.WriteLine($"Account Holder: {AccountHolder}");

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

}

}

}

Checkbalance:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Banking

{

internal class TransferAmount: Account

{

public TransferAmount(string accountNumber, string accountHolder)

: base(accountNumber, accountHolder) { }

public override void Withdraw(double amount)

{

Balance -= amount; // Allow overdraft for simplicity

}

}

}

Current Account:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Banking

{

internal class CurrentAccount: Account

{

public CurrentAccount(string accountNumber, string accountHolder)

: base(accountNumber, accountHolder) { }

public override void Withdraw(double amount)

{

if (Balance >= amount)

{

Balance -= amount;

}

else

{

throw new InvalidOperationException("Insufficient funds.");

}

}

}

}

Interface:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Banking

{

internal interface IAccount

{

string AccountNumber { get; set; }

double Balance { get; }

void Deposit(double amount);

void Withdraw(double amount);

void Transfer(IAccount toAccount, double amount);

void PrintAccountDetails();

}

}

Pogram.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Text;

using System.Threading.Tasks;

namespace Banking

{

internal class Program

{

static void Main(string[] args)

{

//while(true) {

// int option = Convert.ToInt32(Console.ReadLine());

// switch(option)

// {

// case 1:

// }

//create new account

Console.WriteLine("Create Acount:");

Console.WriteLine("Enter name:");

string name = Console.ReadLine();

IAccount savings = new CurrentAccount("12345", name);

savings.PrintAccountDetails();

//case 2:

// Deposit to savings account

Console.WriteLine("Deposit Amount:");

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

savings.Deposit(dep);

savings.PrintAccountDetails();

//case 3:

// Transfer from savings to another account

Console.WriteLine("Transfer amount to:");

Console.WriteLine("Amount:");

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

Console.WriteLine("Account No:");

string accNo = Console.ReadLine();

Console.WriteLine("Account Holder Name:");

string accName = Console.ReadLine();

IAccount checking = new TransferAmount(accNo, accName);

savings.Transfer(checking, amount);

Console.WriteLine("Rs. {0} is transfered to {1}", amount, accNo);

Console.WriteLine();

Console.WriteLine("Your balance:" + savings.Balance);

//savings.PrintAccountDetails();

// Withdraw from current account

Console.WriteLine("Amount to withdraw:");

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

savings.Withdraw(withdraw);

Console.WriteLine("Your balance:" + savings.Balance);

// Print account details

//checking.PrintAccountDetails();

Console.WriteLine();

}

}

}