ASSIGNMENT 8

1)

using System;

using System.Reflection;

using System.IO;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Reflection

{

internal class Program

{

static void FieldInvestigation(Type T)

{

Console.WriteLine("----Fields-----");

FieldInfo[] fieldInfos = T.GetFields();

foreach (FieldInfo fieldInfo in fieldInfos)

{

Console.WriteLine(fieldInfo.Name);

}

}

static void MethodInvestigation(Type T)

{

Console.WriteLine("-----Methods-----");

MethodInfo[] methodInfos = T.GetMethods();

foreach (MethodInfo methodInfo in methodInfos)

{

Console.WriteLine(methodInfo.Name);

}

}

static void Main(string[] args)

{

Console.WriteLine("Enter the name of the explore");

string typName = Console.ReadLine();

Type T = Type.GetType(typName);

FieldInvestigation(T);

MethodInvestigation(T);

Console.ReadKey();

}

}

}

2)

(a)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Reflection1

{

[AttributeUsage(AttributeTargets.Class | AttributeTargets.Constructor | AttributeTargets.Method, AllowMultiple = true)]

class InformationAttribute : Attribute

{

public string InformationString { get; set; }

}

[Information(InformationString = "Class")]

public class ProjectDetails

{

private string ProjectName;

private string Description;

private string ClientDetail;

private string StartDate;

private string EndDate;

public string projectname

{

get { return ProjectName; }

set { ProjectName = value; }

}

public string description

{

get { return Description; }

set { Description = value; }

}

public string clientdetails

{

get { return ClientDetail; }

set { ClientDetail = value; }

}

public string startdate

{

get { return StartDate; }

set { StartDate = value; }

}

public string enddate

{

get { return EndDate; }

set

{

EndDate = value;

}

}

[Information(InformationString = "Constructor")]

public ProjectDetails(string ProjectName, string Description, string ClientDetails, string StartDate, string EndDate)

{

this.ProjectName = ProjectName;

this.Description = Description;

this.ClientDetail = ClientDetails;

this.StartDate = StartDate;

this.EndDate = EndDate;

}

[Information(InformationString = "Method")]

public void Display()

{

Console.WriteLine("Project Name:{0} ", ProjectName);

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

Console.WriteLine("Client Details : {0}", ClientDetail);

Console.WriteLine("Start Date: {0}", StartDate);

Console.WriteLine("End Date: {0}", EndDate);

}

}

public class Program

{

public static void Main(string[] args)

{

ProjectDetails PD = new ProjectDetails("C#", " To Create Appliclatons", "Name:ABC\nPhone.No:1234567890\nEmail:Abc@xyz.com", "07-07-2022", "07-07-2024");

PD.Display();

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

PD.projectname = "C#";

PD.description = "To Create Applications";

PD.clientdetails = "Name:ABC\nPhone.No:1234567890\nEmail:Abc@xyz.com";

PD.startdate = "07-07-2022";

PD.enddate = "07-07-2024";

Console.WriteLine("project name :" + PD.projectname);

Console.WriteLine("description:" + PD.description);

Console.WriteLine("client details: " + PD.clientdetails);

Console.WriteLine("start date: " + PD.startdate);

Console.WriteLine("end date: " + PD.enddate);

}

}

}

(b).

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

[System.AttributeUsage(System.AttributeTargets.Class, AllowMultiple = true)]

public class HdfcAccount : System.Attribute

{

public string name;

int accountNum;

public double balance = 0;

public HdfcAccount(string namae, double firstDep)

{

name = namae;

balance += firstDep;

}

public double getBal()

{

return balance;

}

public string getName()

{

return this.name;

}

public void deposit(double addAmt)

{

balance += addAmt;

}

public bool withdraw(double outAmt)

{

bool chk = true;

if (outAmt <= balance)

{

balance -= outAmt;

}

else if (outAmt > balance)

{

chk = false;

}

return chk;

}

}

public class IcIcIAccount : System.Attribute

{

public string name;

int accountNum;

public double balance = 0;

public IcIcIAccount(string namae, double firstDep)

{

name = namae;

balance += firstDep;

}

public double getBal()

{

return balance;

}

public string getName()

{

return this.name;

}

public void deposit(double addAmt)

{

balance += addAmt;

}

public bool withdraw(double outAmt)

{

bool chk = true;

if (outAmt <= balance)

{

balance -= outAmt;

}

else if (outAmt > balance)

{

chk = false;

}

return chk;

}

}

class Program

{

static void Main(string[] args)

{

List<HdfcAccount> bank = new List<HdfcAccount>();

while (true)

{

Console.Write("1. Set up account\n2. Check balance\n3. Deposit\n4. Withdraw\n\n5. Terminate\n\nSelect Function: ");

string selGo = Console.ReadLine().ToUpper();

Console.WriteLine("\n");

if (selGo == "1")

{

Console.Write("Enter name: ");

string name = Console.ReadLine();

Console.Write("Enter initial balance: ");

double init = double.Parse(Console.ReadLine());

bank.Add(new HdfcAccount(name, init));

Console.WriteLine("Bank Account Added!");

Console.ReadLine();

}

else if (selGo == "2")

{

Console.Write("Enter name: ");

string nmChk = Console.ReadLine();

for (int ix = 0; ix < bank.Count; ix++)

{

if (bank[ix].name == nmChk)

{

Console.WriteLine("Account Found!\nName: {0}\nBalance: {1}", bank[ix].name, bank[ix].balance);

}

}

Console.ReadLine();

}

else if (selGo == "3")

{

Console.Write("Enter name: ");

string nmChk = Console.ReadLine(), nmNow = null;

int accNum = -1;

for (int ix = 0; ix < bank.Count; ix++)

{

if (bank[ix].name == nmChk)

{

nmNow = nmChk;

accNum = ix;

}

}

if (accNum != -1)

{

Console.Write("Amount to Deposit: ");

bank[accNum].deposit(double.Parse(Console.ReadLine()));

Console.WriteLine("Amount successfully deposited!");

}

else { Console.WriteLine("Account not found!"); }

Console.ReadLine();

}

else if (selGo == "4")

{

Console.Write("Enter name: ");

string nmChk = Console.ReadLine(), nmNow = null;

int accNum = -1;

for (int ix = 0; ix < bank.Count; ix++)

{

if (bank[ix].name == nmChk)

{

nmNow = nmChk;

accNum = ix;

}

}

if (accNum != -1)

{

Console.Write("Amount to Withdraw: ");

bool ok = bank[accNum].withdraw(double.Parse(Console.ReadLine()));

if (ok)

{

Console.WriteLine("Amount successfully withdrawn!");

if (bank[accNum].balance == 0)

{

bank[accNum] = null;

Console.WriteLine("Account closed!");

}

}

else { Console.WriteLine("Insufficient funds!"); }

}

else { Console.WriteLine("Account not found!"); }

Console.ReadLine();

}

else if (selGo == "5")

{

break;

}

else

{

Console.WriteLine("Invalid Syntax!");

Console.ReadLine();

}

Console.Clear();

}

List<IcIcIAccount> account = new List<IcIcIAccount>();

while (true)

{

Console.Write("1. Set up account\n2. Check balance\n3. Deposit\n4. Withdraw\n\n5. Terminate\n\nSelect Function: ");

string selGo = Console.ReadLine().ToUpper();

Console.WriteLine("\n");

if (selGo == "1")

{

Console.Write("Enter name: ");

string name = Console.ReadLine();

Console.Write("Enter initial balance: ");

double init = double.Parse(Console.ReadLine());

account.Add(new IcIcIAccount(name, init));

Console.WriteLine("Bank Account Added!");

Console.ReadLine();

}

else if (selGo == "2")

{

Console.Write("Enter name: ");

string nmChk = Console.ReadLine();

for (int ix = 0; ix < bank.Count; ix++)

{

if (bank[ix].name == nmChk)

{

Console.WriteLine("Account Found!\nName: {0}\nBalance: {1}", bank[ix].name, bank[ix].balance);

}

}

Console.ReadLine();

}

else if (selGo == "3")

{

Console.Write("Enter name: ");

string nmChk = Console.ReadLine(), nmNow = null;

int accNum = -1;

for (int ix = 0; ix < bank.Count; ix++)

{

if (account[ix].name == nmChk)

{

nmNow = nmChk;

accNum = ix;

}

}

if (accNum != -1)

{

Console.Write("Amount to Deposit: ");

account[accNum].deposit(double.Parse(Console.ReadLine()));

Console.WriteLine("Amount successfully deposited!");

}

else { Console.WriteLine("Account not found!"); }

Console.ReadLine();

}

else if (selGo == "4")

{

Console.Write("Enter name: ");

string nmChk = Console.ReadLine(), nmNow = null;

int accNum = -1;

for (int ix = 0; ix < bank.Count; ix++)

{

if (account[ix].name == nmChk)

{

nmNow = nmChk;

accNum = ix;

}

}

if (accNum != -1)

{

Console.Write("Amount to Withdraw: ");

bool ok = account[accNum].withdraw(double.Parse(Console.ReadLine()));

if (ok)

{

Console.WriteLine("Amount successfully withdrawn!");

if (account[accNum].balance == 0)

{

account[accNum] = null;

Console.WriteLine("Account closed!");

}

}

else { Console.WriteLine("Insufficient funds!"); }

}

else { Console.WriteLine("Account not found!"); }

Console.ReadLine();

}

else if (selGo == "5")

{

break;

}

else

{

Console.WriteLine("Invalid Syntax!");

Console.ReadLine();

}

}

}

}