## 实验四 创建静态类与静态方法

### 实验目的

掌握this的使用

掌握静态成员和静态方法的应用

掌握构造方法的使用

### 第一部分 上机训练

1、个人银行业务主要由存款、取款、查询余额、转帐组成，一般银行为我们办理个人业务时，需要我们实现存入一部分钱方可为我们开户，创建一个PersonalAccount类，实现以上功能。

1）创建一个名为Bank的控制台应用程序

2）对于一个软件系统，我们需要显示其版本号，软件名称，注册码等信息，这些信息可以用静态成员变量保存。因此首先我们可以建一个GlobalSettings类存储这些一般信息，类的创建方法和实验四相同，注意不要忘记类名前加public，GlobalSettings类的代码如下：

using System;

using System.Collections.Generic;

using System.Text;

namespace Bank

{

public class GlobalSettings

{

private static string softwareName = "创新银行个人业务管理系统";

public static string SoftwareName

{

get { return GlobalSettings.softwareName; }

set { GlobalSettings.softwareName = value; }

}

private static string softwareVersion = "1.0版";

public static string SoftwareVersion

{

get { return GlobalSettings.softwareVersion; }

set { GlobalSettings.softwareVersion = value; }

}

private static string softwareCompany = "淮安信息职业技术学院计算机系创新实验室--宋桂岭";

public static string SoftwareCompany

{

get { return GlobalSettings.softwareCompany; }

set { GlobalSettings.softwareCompany = value; }

}

private static string softwareDescribe = "本系统的创意、著作权均归作者本人所有，任何单位和个人如果引用本系统的任何内容，\n作者保留追究其法律责任的权利，特此声明。";

public static string SoftwareDescribe

{

get { return GlobalSettings.softwareDescribe; }

set { GlobalSettings.softwareDescribe = value; }

}

}

}

3）创建PersonalAccount类，利用构造方法实现开户，利用成员方法实现余额查询、存入、提取、转帐等功能，完整代码如下：

using System;

using System.Collections.Generic;

using System.Text;

namespace Bank

{

public class PersonalAccount

{

private double accountRemainValue;//余额

private string accountName;//姓名

public double AccountRemainValue

{

get { return accountRemainValue; }

set { accountRemainValue = value; }

}

public string AccountName

{

get { return accountName; }

set { accountName = value; }

}

#region 开户

//利用构造方法实现开户

public PersonalAccount(string accountName, double accountRemainValue)

{

//注意this用法

this.accountName = accountName;

this.accountRemainValue = accountRemainValue;

}

#endregion

#region 存、取、转帐

public void Deposit(double value)

{

this.accountRemainValue += value;

}

public void WithDraw(double value)

{

this.accountRemainValue -= value;

}

public void Transfer(PersonalAccount toAccount, double value)

{

this.accountRemainValue -= value;

toAccount.AccountRemainValue += value;

}

#endregion

#region 显示余额

public void DisplayRemainAccount()

{

Console.WriteLine("尊敬的{0}先生/女士，您好，您的帐户余额为{1:0.00}元", accountName, AccountRemainValue);

}

#endregion

}

}

4）在Program.cs中输入如下代码，实现系统功能：

using System;

using System.Collections.Generic;

using System.Text;

namespace Bank

{

class Program

{

static void Main(string[] args)

{

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

Console.WriteLine();

Console.WriteLine(" " + GlobalSettings.SoftwareName + GlobalSettings.SoftwareVersion);

Console.WriteLine();

Console.WriteLine(" " + "制作单位：" + GlobalSettings.SoftwareCompany);

Console.WriteLine();

Console.WriteLine(" " + GlobalSettings.SoftwareDescribe);

Console.WriteLine();

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

//开户

PersonalAccount myAccount = new PersonalAccount("关羽", 200);

PersonalAccount sheAccount = new PersonalAccount("关夫人", 30000);

//业务

bool op = true;

while (op)

{

Console.WriteLine();

Console.WriteLine("请输入您要办理的业务：1.存款2.提款3.转帐4.退出");

string cmd = Console.ReadLine();

switch (cmd)

{

case "1":

HandleDesposit(myAccount);

break;

case "2":

HandleWithDraw(myAccount);

break;

case "3":

HandleTransfer(myAccount, sheAccount);

break;

case "4":

op = false;

break;

}

}

}

static void HandleDesposit(PersonalAccount account)

{

Console.Write("请输入存款金额:");

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

account.AccountRemainValue += value;

Console.WriteLine("存款操作完成!");

account.DisplayRemainAccount();

}

static void HandleWithDraw(PersonalAccount account)

{

Console.Write("请输入取款金额:");

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

if (account.AccountRemainValue < value)

{

Console.WriteLine("帐户余额不足，取款操作失败！");

return;

}

account.AccountRemainValue -= value;

Console.WriteLine("取款操作完成!");

account.DisplayRemainAccount();

}

static void HandleTransfer(PersonalAccount fromAccount, PersonalAccount toAccount)

{

Console.Write("请输入转帐金额:");

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

if (fromAccount.AccountRemainValue < value)

{

Console.WriteLine("帐户余额不足，取款操作失败！");

return;

}

fromAccount.AccountRemainValue -= value;

toAccount.AccountRemainValue += value;

Console.WriteLine("转帐操作完成!");

fromAccount.DisplayRemainAccount();

toAccount.DisplayRemainAccount();//实际中你是看不到对方的余额的，所以运行结果有点奇怪，不是吗？

}

}

}

### 第二部分 实战提高

设计一个控制台应用程序，定义一个Box类，该类表示一个立方体，完成该类运算符重载等相关功能。具体要求如下：

（1）Box类中定义3个私有字段（代表长、宽、高）及相应的构造函数

（2）Box类中重载ToString方法输出长宽高的值。

（3）Box类中重载一元运算符“-”。

（4）Box类中重载二元运算符“+”和“-”。

（5）Box类中重载二元运算符“>”和“<”。

（6）Box类中重载二元运算符“==”和“！=”。

（7）在Main方法中测试上述运算符。

**代码和运行结果图请表示在下面：**

代码如下：

#Box.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MySoftwareOnDotnetFramework.Box

{

public class Box

{

private double length;

private double width;

private double height;

public double Length

{

set

{

length = value;

}

get

{

return length;

}

}

public double Width

{

set

{

width = value;

}

get

{

return width;

}

}

public double Height

{

set

{

height = value;

}

get

{

return height;

}

}

public double getVolume()

{

return length \* width \* height;

}

public static double operator-(Box box) //重载一元运算符“-”

{

return -box.getVolume();

}

public static double operator+(Box b1,Box b2) //重载“+”

{

return b1.getVolume() + b2.getVolume();

}

public static double operator-(Box b1,Box b2) //重载二元运算符“-”

{

return b1.getVolume() - b2.getVolume();

}

public static bool operator<(Box b1,Box b2) //对大于和小于的重载

{

return (b1.getVolume() < b2.getVolume()) ? true : false;

}

public static bool operator>(Box b1,Box b2)

{

return (b1.getVolume() > b2.getVolume()) ? true : false;

}

public static bool operator==(Box b1,Box b2) //对“==”和“!=”的重载

{

return b1.getVolume().Equals(b2.getVolume()) ? true : false;

}

public static bool operator!=(Box b1,Box b2)

{

return b1.getVolume().Equals(b2.getVolume()) ? false : true;

}

public override string ToString()

{

return "length: " + length + " width: " + width + " height: " + height + " volume: " + this.getVolume();

}

}

}

#Program.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace MySoftwareOnDotnetFramework.Box

{

class Program

{

static void Main(string[] args)

{

Box box1 = new Box(); //初始化区域

box1.Length = 1.5;

box1.Width = 6.9;

box1.Height = 5.7;

Box box2 = new Box();

box2.Length = 3.2;

box2.Width = 1.1;

box2.Height = 4.4;

Console.WriteLine("toString的重载：");

Console.WriteLine(box1.ToString());

Console.WriteLine(box2.ToString());

Console.WriteLine();

Console.WriteLine("一元运算符“-”的重载：");

Console.WriteLine("-box1: "+-box1);

Console.WriteLine();

Console.WriteLine("二元运算符“+”的重载：");

Console.WriteLine("box1 + box2: " + (box1 + box2));

Console.WriteLine();

Console.WriteLine("二元运算符“-”的重载：");

Console.WriteLine("box1 - box2: " + (box1 - box2));

Console.WriteLine();

Console.WriteLine("二元运算符“<”和“>”的重载：");

Console.WriteLine("box1 < box2: " + (box1 < box2));

Console.WriteLine("box1 > box2: " + (box1 > box2));

Console.WriteLine();

Console.WriteLine("二元运算符“==”和“!=”的重载: ");

Console.WriteLine("box1 == box2: " + (box1 == box2));

Console.WriteLine("box1 != box2: " + (box1 != box2));

Console.ReadLine();

}

}

}

运行结果如下：

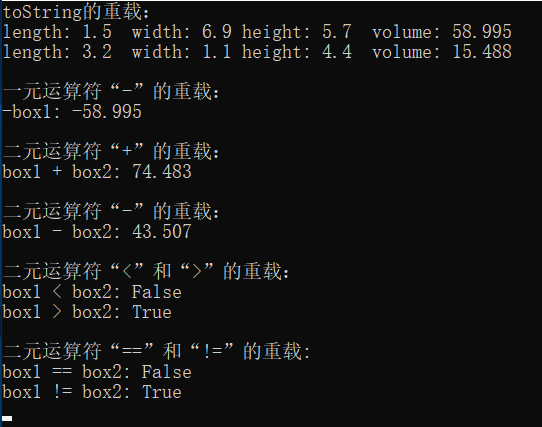


图4-1 重载运行结果

### 第三部分 知识总结

|  |  |
| --- | --- |
| **学生总结** | 请将问题答案写在下面空白处 |
| 完成上机训练模块花费的时间 | 20 分钟 |
| 完成实战提高模块花费的时间 | 40分钟 |
| 本次实验对你而言的难点 |  |
| 本次实验的收获 |  |
| 希望老师强化讲解的知识要点 |  |
| 希望老师帮助回答的问题 |  |
| 本次实验自评分数  （五分制） | 4.8 |