## 实验五 实现继承和抽象类

### 实验目的

编写一个单一继承的C#程序

编写一个抽象类，并用一个类继承这个抽象类

### 第一部分 上机训练

1、在学校教务管理系统中，有三种用户，一种是系统管理员，一种是学生，一种是教师，编写C#类实现该继承关系。系统管理员/学生/教师都有用户名和密码属性，并都有Login和显示自身信息的方法，另外，学生拥有班级和成绩两个特殊属性以及查询成绩这一方法，教师拥有部门和工资两个特殊属性以及查询工资这一方法，系统管理员可以添加学生和教师信息。由于教师和学生个人信息不同，所以学生和教师重载基类的DisplayInfo方法。（本例和教材一致，再次出现的目的是通过反复训练，理解继承的作用）

1）类的继承关系可以表示如下：



2）创建控制台应用程序，命名为SchoolManagement

3）创建名为User的基类，基类定义了用户名和密码属性，以及Login和DisplayInfo方法

using System;

using System.Collections.Generic;

using System.Text;

namespace SchoolManagement

{

public class User

{

private string userName;//用户名

public string Name

{

get { return userName; }

set { userName = value; }

}

private string password;//密码

public string Password

{

get { return password; }

set { password = value; }

}

private string userType;//用户类型

public string UserType

{

get { return userType; }

set { userType = value; }

}

public void Login()

{

if (userName.Equals("1") && password.Equals("1"))

{

DisplayPersonInfo();

}

}

public virtual void DisplayPersonInfo() //注意virtual关键字

{

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

Console.WriteLine("用户类型:{0}", userType);

}

}

}

2）创建名为Student的派生类，具备班级和成绩两个特殊属性，以及查询成绩的特殊方法,覆盖基类的显示个人信息方法

using System;

using System.Collections.Generic;

using System.Text;

namespace SchoolManagement

{

public class Student : User

{

private string classID;

public string ClassID

{

get { return classID; }

set { classID = value; }

}

private float stuScore;

public float StuScore

{

get { return stuScore; }

set { stuScore = value; }

}

public Student()

{

UserType = "学生";

}

public override void DisplayPersonInfo()

{

base.DisplayPersonInfo();

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

}

public void CheckScore()

{

Console.WriteLine("成绩:{0}", this.stuScore);

}

}

}

3）创建名为Teacher的派生类，具备部门和工资两个特殊属性，以及查询工资的特殊方法,覆盖基类的显示个人信息方法。

using System;

using System.Collections.Generic;

using System.Text;

namespace SchoolManagement

{

public class Teacher : User

{

private string department;//部门

public string Department

{

get { return department; }

set { department = value; }

}

private float salary;//工资

public float Salary

{

get { return salary; }

set { salary = value; }

}

public Teacher()

{

UserType = "教师";

}

public override void DisplayPersonInfo()

{

base.DisplayPersonInfo();

Console.WriteLine("部门:" + this.department);

}

public void checkSalary()

{

Console.WriteLine("工资:" + this.salary);

}

}

}

4）创建名为SysManager的派生类，具备插入学生信息和插入教师信息两个方法

using System;

using System.Collections.Generic;

using System.Text;

namespace SchoolManagement

{

public class SysManager : User

{

public SysManager()

{

UserType = "管理员";

}

public Student InsertStudent()

{

Student stu = new Student();

stu.Name = "张郭郭";

stu.UserType = "教师";

stu.ClassID = "310730";

stu.StuScore = 70;

return stu;

}

public Teacher InsertTeacher()

{

Teacher teacher = new Teacher();

teacher.Name = "李沙沙";

teacher.Department = "计算机系";

teacher.Salary = 5000;

return teacher;

}

}

}

5）在Program.cs的Main方法中调用以上类，测试效果如下：

using System;

using System.Collections.Generic;

using System.Text;

namespace SchoolManagement

{

class Program

{

static void Main(string[] args)

{

//系统管理员登录

SysManager sys = new SysManager();

sys.Name = "1";

sys.Password = "1";

sys.Login();

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

//系统管理员录入学生和教师信息

Teacher teacher = sys.InsertTeacher();

Student stu = sys.InsertStudent();

//显示教师信息，查询工资

teacher.DisplayPersonInfo();

teacher.checkSalary();

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

//显示学生信息，查询成绩

stu.DisplayPersonInfo();

stu.CheckScore();

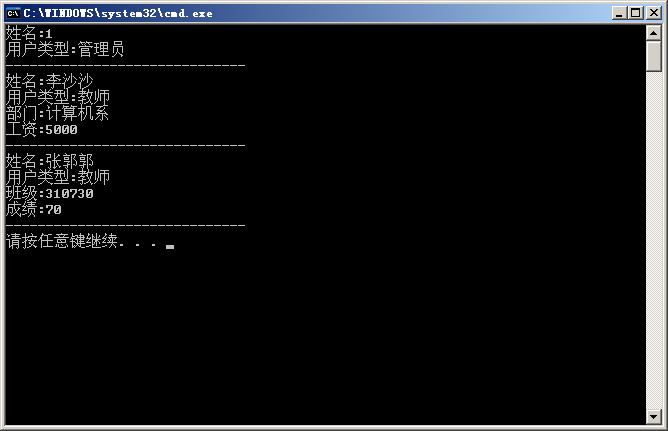
Console.WriteLine("------------------------------");

}

}

}

6）最终运行效果如下



### 实战提高

定义一个车（vehicle）基类，具有数据成员 maxspeed, weight（均为 int  型）， 函数成员 run(), stop()，由此派生出自行车（bicycle）类、汽车（motorcar）类。其中， bicycle  类新增数据成员高度（height），motorcar  类新增数据成员座位数（seatnum）属性。再从 bicycle和 motorcar  派生出摩托车（motorcycle）类，并在主程序中测试这个类。（每个类都要求定义构造函数和析构函数）（提示： ① 注意把 vehicle 设置为虚基类； ② run(), stop()函数体，通过输出字符串 run, stop 简单模拟。）

**代码和运行结果图请表示在下面：**

**代码如下：**

**#Vehicle.cs**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace MySoftwareOnDotnetFramework.Vehicle**

**{**

**public class Vehicle**

**{**

**private int maxspeed;**

**private int weight;**

**public Vehicle() { }**

**public Vehicle(int maxspeed,int weight)**

**{**

**this.maxspeed = maxspeed;**

**this.weight = weight;**

**}**

**public int Maxspeed**

**{**

**get**

**{**

**return maxspeed;**

**}**

**set**

**{**

**maxspeed = value;**

**}**

**}**

**public int Weight**

**{**

**get**

**{**

**return weight;**

**}**

**set**

**{**

**weight = value;**

**}**

**}**

**public virtual void run()**

**{**

**Console.WriteLine("A vehicle is running.");**

**}**

**public virtual void stop()**

**{**

**Console.WriteLine("The vehicle stopped.");**

**}**

**}**

**}**

**#Bicycle.cs**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace MySoftwareOnDotnetFramework.Vehicle**

**{**

**public class Bicycle : Vehicle**

**{**

**private int height;**

**public Bicycle() { }**

**public Bicycle(int maxspeed,int weight,int height):base(maxspeed,weight)**

**{**

**this.height = height;**

**}**

**public int Height**

**{**

**get**

**{**

**return height;**

**}**

**set**

**{**

**height = value;**

**}**

**}**

**public override void run()**

**{**

**base.run();**

**Console.WriteLine("A bicycle is running.");**

**}**

**public override void stop()**

**{**

**base.stop();**

**Console.WriteLine("The bicycle stopped.");**

**}**

**}**

**}**

**#Motorcycle.cs**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**namespace MySoftwareOnDotnetFramework.Vehicle**

**{**

**public class Motorcycle:Bicycle**

**{**

**private int seatnum;**

**public Motorcycle() { }**

**public Motorcycle(int maxspeed,int weight,int height,int seatnum) : base(maxspeed, weight, height)**

**{**

**this.seatnum = seatnum;**

**}**

**public int Seatnum**

**{**

**get**

**{**

**return seatnum;**

**}**

**set**

**{**

**seatnum = value;**

**}**

**}**

**public override void run()**

**{**

**base.run();**

**Console.WriteLine("A motorcycle is running.");**

**}**

**public override void stop()**

**{**

**base.stop();**

**Console.WriteLine("The motorcycle stopped.");**

**}**

**}**

**}**

**#Program.cs**

**static void Main(string[] args)**

**{**

**Motorcycle motorcycle = new Motorcycle(100, 200, 26, 2);**

**motorcycle.run();**

**motorcycle.stop();**

**Console.ReadLine();**

**}**

**运行结果如下：**

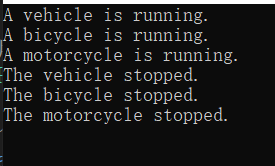


图5-1 运行结果

### 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 | 5分钟 |
| 完成实战提高模块花费的时间 | 25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 |