# 实验报告

**姓名**： 胡启明 **学号**： 164547 **日期：**

**实验名称**： **实验一 类和对象编程 成绩：**

|  |
| --- |
| **一、实验目的及要求**  **【实验目的】**  **1.如何创建类和对象。**  **2.如何为定义的类编写相应的方法。**  **3.如何通过属性访问对象中的数据**  **【实验要求】**  **1.写出程序，并调试程序，要给出测试数据和实验结果。**  **2.整理上机步骤，总结经验和体会。**  **3.完成实验日志和上交程序。**  **二、实验环境**  **Windows XP操作系统，Visual Studio 2005及以上版本**  **三、实验源程序及关键代码解释**  **实验1\_1**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Exp\_1\_1  {  class Time  {  // 定义变量  private int hour;  private int minute;  private int second;  // 定义包含默认值的构造方法  public Time(int hour=0,int minute=0,int second=0)  {  this.hour = hour;  this.minute = minute;  this.second = second;  }  // 定义属性和对应访问器，访问器返回格式化后的字符串  public string Hour{  get  {  return string.Format("hour:{0}",hour);  }  }  public string Minute  {  get  {  return string.Format("minute:{0}", minute);  }  }  public string Second  {  get  {  return string.Format("second:{0}", second);  }  }  // 定义自增方法  public void HourIncrement()  {  hour = ++hour >= 24 ? 0 : hour;  }  public void MinuteIncrement()  {  minute++;  if(minute >= 60)  {  HourIncrement();  minute = 0;  }  }  public void SecondIncrement()  {  second++;  if (second >= 60)  {  MinuteIncrement();  second = 0;  }  }  // 定义展示方法  public string getTime()  {  // 返回同时显示时分秒的格式化后的字符串  return string.Format("{0:D2}:{1:D2}:{2:D2}", hour, minute, second);  }    }  class Program  {  static void Main(string[] args)  {  Time time = new Time(23,59,59);  // 同时显示时分秒  Console.WriteLine(time.getTime());  // 分别显示时分秒  Console.WriteLine(time.Hour);  Console.WriteLine(time.Minute);  Console.WriteLine(time.Second);  // 自增1秒，嵌套调用了自增1分和自增1小时  time.SecondIncrement();  // 同时显示时分秒  Console.WriteLine(time.getTime());  // 分别显示时分秒  Console.WriteLine(time.Hour);  Console.WriteLine(time.Minute);  Console.WriteLine(time.Second);  Console.ReadKey();  }  }  }  **实验1\_2**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Exp\_1\_2  {  class Student  {  // 定义变量  private string number;  private string name;  private char sex;  // 定义带默认参数的构造方法  public Student(string number="",string name="",char sex = 'M')  {  Number = number;  this.name = name;  // 这样就不用再次判断sex的输入范围是否符合要求了  Sex = sex;  }  // 定义属性和对应访问器  public string Number  {  set  {  number = value;  }  get  {  return number;  }  }  // Name属性只具有get访问器，则该属性为只读属性  public string Name  {  get  {  return name;  }  }  public char Sex  {  // 当输入性别的值不为F或M中的任意一个的时候就默认设置为男性  set  {  if (value != 'M' && value != 'F')  {  sex = 'M';  }else  {  sex = value;  }  }  get  {  return sex;  }  }  // 显示学生信息  public string GetStudent()  {  return string.Format("学号：{0}\n姓名：{1}\n性别：{2}\n", number, name, sex == 'M' ? "男" : "女");  }  }  class Program  {  static void Main(string[] args)  {  // 要求用户按要求格式输入数据  Console.WriteLine("请以空格为分割在同一行分别输入学号 姓名 性别：");  // 获取分割后的字符串数组  string[] inputs = Console.ReadLine().Split(' ');  // 建立Student对象  Student student = new Student(inputs[0], inputs[1], inputs[2][0]);  Console.WriteLine(student.GetStudent());  // 修改学号和性别  student.Number = "145685";  student.Sex = 'F';  Console.WriteLine(student.GetStudent());  // 输入非法性别  student.Sex = 'Q';  Console.WriteLine(student.GetStudent());  Console.ReadKey();  }  }  }  **实验1\_3**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Exp\_1\_3  {  // 这部分类中只放变量和属性  partial class Student  {  // 定义成员变量  private string number;  private string name;  private double score;  // 定义静态变量  private static int studentCount;  private static double scoreSum;  private static double scoreAverage;  // 定义属性和对应访问器  public string Number  {  set  {  number = value;  }  get  {  return number;  }  }  public string Name  {  set  {  name = value;  }  get  {  return name;  }  }  public double Score  {  set  {  score = value;  }  get  {  return score;  }  }  public static int StudentCount  {  set  {  studentCount = value;  }  get  {  return studentCount;  }  }  public static double ScoreSum  {  set  {  scoreSum = value;  }  get  {  return scoreSum;  }  }  public static double ScoreAverage  {  set  {  scoreAverage = value;  }  get  {  return scoreAverage;  }  }  }  // 这部分类中只放各种方法  partial class Student  {  // 定义构造方法  public Student(string number="",string name="",double score = 0)  {  Number = number;  Name = name;  Score = score;  // 处理静态变量  studentCount++;  scoreSum += score;  scoreAverage = scoreSum / studentCount;  }  // 定义显示学生信息的方法  public string GetStudent()  {  return string.Format("学号：{0}\n 姓名：{1}\n 成绩：{2}\n", number, name, score);  }  // 定义统计数据显示方法  public static string GetAnalyse()  {  return string.Format("学生人数：{0}\n学生总成绩：{1}\n学生平均成绩：{2:f}\n", studentCount, scoreSum, scoreAverage);  }    }  class Program  {  static void Main(string[] args)  {  Student[] students = new Student[3];  Console.WriteLine("对于每一个同学请以空格为分割在每一行分别输入学号 姓名 成绩：");  for(int i = 0; i < 3; i++)  {  // 获取分割后的字符串数组  string[] inputs = Console.ReadLine().Split(' ');  students[i] = new Student(inputs[0], inputs[1], Convert.ToDouble(inputs[2]));  }  Console.WriteLine("输入的学生信息有: ");  foreach (Student stu in students)  {  Console.WriteLine(stu.GetStudent());  }  Console.WriteLine("学生数据统计为: ");  Console.WriteLine(Student.GetAnalyse());  Console.ReadKey();  }  }  }  **四、实验调试过程及运行结果**  **实验1\_1**    **实验1\_2**  输入：  164547 胡启明 M    **实验1\_3**  输入：  164503 张三 92  164504 李四 85  164505 王五 98    **五、实验总结：**  **1. C#中的partial标识符很好用，可以使代码变得整洁；**  **2. C#中的属性和访问器也让代码变得整洁；**  **3. C#中的形参默认值很好用，是C/C++里面没有的特性；**  **4. C#中的模板字符串一般好用，没有感觉多强大；**  **5. C#中无法格式化输入，这一点上并不如Java或者C做的好，不过想想似乎也没必要，反正用户的输入使用Convert转换不了抛异常就完事了，横竖都是异常，就是C#还稍微麻烦一些，而且数据的类型转换方法全部都放到Convert类中了，和Java放到对应类型的包装类中有所不同，而且C#似乎没有基本类型和对应包装类的区别，反正都可以调用对应的方法，这点有点重复。** |

# 实验报告

**姓名**： 胡启明 **学号**： 164547 **日期：**

**实验名称**： **实验二 继承与多态编程 成绩：**

|  |
| --- |
| **一、实验目的及要求**  **【实验目的】**  1.理解继承的含义，掌握派生类的定义方法和实现；  2.理解虚函数在类的继承层次中的作用，虚函数的引入对程序运行时的影响，能够对使用虚函数的简单程序写出程序结果。  3.编写体现类的继承性（成员变量，成员方法，成员变量隐藏）的程序；  4.编写体现类多态性（成员方法重载，构造方法重载）的程序。  **【实验要求】**  1.写出程序，并调试程序，要给出测试数据和实验结果。  2.整理上机步骤，总结经验和体会。  3.完成实验日志和上交程序。  **二、实验环境**  **Windows XP操作系统，Visual Studio 2005及以上版本**  **三、实验源程序及关键代码解释**  **实验2\_1\_1**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Exp\_2\_1  {  /// <summary>  /// 定义变量 isbn(书号)、title（标题）、author(作者)、press（出版社）、price（价格）  /// 以及对应属性  /// </summary>  partial class Book  {  private readonly string isbn;  private string title;  private string author;  private string press;  private float price;  public string Isbn{  get  {  return isbn;  }  }  public string Title  {  set  {  title = value;  }  get  {  return title;  }  }  public string Author  {  set  {  author = value;  }  get  {  return author;  }  }  public string Press  {  set  {  press = value;  }  get  {  return press;  }  }  public float Price  {  set  {  price = value;  }  get  {  return price;  }  }  }  partial class Book  {  public Book(string isbn,string title,string author,string press,float price)  {  this.isbn = isbn;  this.title = title;  this.author = author;  this.press = press;  this.price = price;  }  // 使用形参列表调用自身的其他重载的构造方法  public Book(string isbn): this(isbn, "未知", "未知", "未知", 0) {}  public void Show()  {  Console.WriteLine("书号：{0}\n 标题：{1}\n 作者：{2}\n 出版社：{3}\n 价格：{4}\n ",  isbn,title,author,press,price);  }  }  class Program  {  static void Main(string[] args)  {  Book book1 = new Book("kt750911", "卡通书01", "张三", "河工大出版社", 30f);  Book book2 = new Book("kt750913");  book1.Show();  book2.Show();  Console.ReadKey();  }  }  }  **实验2\_1\_2**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Exp\_2\_1\_2  {  /// <summary>  /// 定义圆，作为球体，圆柱体和圆锥体的基类  /// </summary>  class Circle  {  public double Radius { get; set; }  public Circle(double radius)  {  Radius = radius;  }  public double GetArea()  {  return Math.PI \* Math.Pow(Radius, 2);  }  }  /// <summary>  /// 定义“体”的接口，声明一个返回该物体体积的抽象方法  /// </summary>  interface Body  {  double GetVolume();  }  // 球体，圆柱体和圆锥体的类的定义  class Sphere : Circle, Body  {  public Sphere(double radius) : base(radius)  {  }  public double GetVolume()  {  return (4 / 3) \* Math.PI \* Math.Pow(Radius, 3);  }  }  class Cylinder : Circle, Body  {  public double Height { get; set; }  public Cylinder(double radius, double height) : base(radius)  {  Height = height;  }  public double GetVolume()  {  return GetArea() \* Height;  }  }  class Cone : Circle, Body  {  public double Height { get; set; }  public Cone(double radius, double height) : base(radius)  {  Height = height;  }  public double GetVolume()  {  // 这种分数一定要加f，切记切记  return (1f / 3) \* GetArea() \* Height;  }  }  class Program  {  /// <summary>  /// 采用多态的方式打印某物体的体积  /// </summary>  /// <param name="body">继承了Body接口的物体的类</param>  static void PrintVolumeOf(Body body)  {  Console.WriteLine("该物体的体积为{0:f}", body.GetVolume());  }  static void Main(string[] args)  {  Console.WriteLine("请输入球体的半径： ");  Sphere sphere = new Sphere(Convert.ToDouble(Console.ReadLine()));  Console.WriteLine("请以空格为分割输入圆柱体的底面半径和高： ");  string[] tmp = Console.ReadLine().Split(' ');  Cylinder cylinder = new Cylinder(Convert.ToDouble(tmp[0]), Convert.ToDouble(tmp[1]));  Console.WriteLine("请以空格为分割输入圆锥体的底面半径和高： ");  tmp = Console.ReadLine().Split(' ');  Cone cone = new Cone(Convert.ToDouble(tmp[0]), Convert.ToDouble(tmp[1]));  // 打印物体的体积  PrintVolumeOf(sphere);  PrintVolumeOf(cylinder);  PrintVolumeOf(cone);  Console.ReadKey();  }  }  }  **实验2\_1\_3**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Exp\_1\_1\_3  {  /// <summary>  /// 这个类的成员包括书名、作者、馆藏数量  /// </summary>  partial class Card  {  public string BookName { get; set; }  public string Author { get; set; }  public int Number { get; set; }  }  /// <summary>  /// store书的入库处理，show显示图书信息  /// </summary>  partial class Card  {  public Card(string bookName,string author,int number=0)  {  BookName = bookName;  Author = author;  Number = number;  }    public void Store()  {  Number++;  }  public void Show()  {  Console.WriteLine(" 书名：{0}\n 作者：{1}\n 馆藏数量：{2}\n ", BookName, Author, Number);  }  }  class Program  {  private const int SORT\_BY\_BOOK\_NAME = 1;  private const int SORT\_BY\_AUTHOR = 2;  private const int SORT\_BY\_NUMBER = 3;  private static List<Card> cardList;  /// <summary>  /// 将图书按某一顺序展示  /// 先浅拷贝一下原列表  /// 然后使用List的Sort方法传入使用箭头函数表示的比较委托进行排序  /// 最后遍历列表调用Card对象的Show方法依次输出  /// </summary>  /// <param name="sortingFlag">排序标志字段</param>  static void ShowBooks(int sortingFlag=0)  {  List<Card> tempList = new List<Card>(cardList);  switch (sortingFlag)  {  case SORT\_BY\_BOOK\_NAME:  Console.WriteLine("\n按书名排序显示： ");  tempList.Sort((c1,c2)=>c1.BookName.CompareTo(c2.BookName));  break;  case SORT\_BY\_AUTHOR:  Console.WriteLine("\n按作者排序显示： ");  tempList.Sort((c1, c2) => c1.Author.CompareTo(c2.Author));  break;  case SORT\_BY\_NUMBER:  Console.WriteLine("\n按馆藏数量排序显示： ");  tempList.Sort((c1, c2) => c1.Number.CompareTo(c2.Number));  break;  default:  Console.WriteLine("\n按默认排序显示： ");  break;  }  foreach (Card card in tempList)  {  card.Show();  }  }  static void Main(string[] args)  {  cardList = new List<Card>();  Console.WriteLine("请输入您要入库的图书总数： ");  int n = Convert.ToInt16(Console.ReadLine());  Console.WriteLine("请以空格为分隔符在每行输入图书的书名 作者 馆藏数量： ");  for (int i = 0; i < n; i++)  {  string[] tmp = Console.ReadLine().Split(' ');  cardList.Add(new Card(tmp[0], tmp[1], Convert.ToInt16(tmp[2])));  }  ShowBooks();  ShowBooks(SORT\_BY\_BOOK\_NAME);  ShowBooks(SORT\_BY\_AUTHOR);  ShowBooks(SORT\_BY\_NUMBER);    Console.ReadKey();  }  }  }  **实验2\_2\_1**  using System;  using System.Globalization;  namespace Exp\_2\_2\_1  {  /// <summary>  /// 声明雇员的抽象类，包含三个基本属性  /// 以及工资计算的抽象函数GetSalary  /// 并实现了基本的展示方法并使用virtual修饰符声明为虚函数  /// 派生类中使用override修饰符重写该方法  /// 这样通过基类对象调用到的Show方法就是派生类的Show方法了  ///  /// 在该类中同时定义了日期的输入格式，方便在输入时调用  /// </summary>  abstract class Employee  {  public static DateTimeFormatInfo DATE\_FORMAT;  public Employee(string name, string address, DateTime birthday)  {  DATE\_FORMAT = new DateTimeFormatInfo();  DATE\_FORMAT.ShortDatePattern = "yyyy-MM-dd";  Name = name;  Address = address;  Birthday = birthday;  }  public string Name { get; set; }  public string Address { get; set; }  public DateTime Birthday { get; set; }  public abstract double GetSalary();  public virtual void Show()  {  Console.WriteLine(" 姓名：{0}\n 地址：{1}\n 出生日期：{2}\n 本月工资：{3:C2}",  Name, Address, Birthday.ToString("D"),GetSalary());  }  }  /// <summary>  /// 程序员类  /// 工资=底薪+提成  /// 底薪使用静态变量和属性维护，保证该类的全部对象底薪字段一致  /// 实现了抽象基类中的工资计算方法GetSalary  /// 并覆盖了基类中的虚方法Show，说明了该员工所处岗位  /// </summary>  class Programmer:Employee  {  private static double baseSalary = 2000;  public Programmer(string name, string address,  DateTime birthday,double bonus) : base(name, address, birthday)  {  Bonus = bonus;  }  public static double BaseSalary {  get  {  return baseSalary;  }  set  {  baseSalary = value;  }  }  public double Bonus { get; set; }  public override double GetSalary()  {  return BaseSalary + Bonus;  }  public override void Show()  {  Console.WriteLine("\n 员工岗位：程序员");  base.Show();  }  }  class Administrator : Employee  {  private static double baseSalary = 5000;  public Administrator(string name, string address,  DateTime birthday,double bonus) : base(name, address, birthday)  {  Bonus = bonus;  }  public static double BaseSalary  {  get  {  return baseSalary;  }  set  {  baseSalary = value;  }  }  public double Bonus { get; set; }  public override double GetSalary()  {  return BaseSalary + Bonus;  }  public override void Show()  {  Console.WriteLine("\n 员工岗位：高层管理");  base.Show();  }  }  /// <summary>  /// 秘书类  /// 工资=固定工资  /// 固定工资使用静态变量和属性维护  /// 也同样实现了GetSalary方法  /// </summary>  class Secretary : Employee  {  private static double salary = 3000;  public Secretary(string name, string address, DateTime birthday) : base(name, address, birthday)  {  }  public static double Salary  {  get  {  return salary;  }  set  {  salary = value;  }  }  public override double GetSalary()  {  return Salary;  }  public override void Show()  {  Console.WriteLine("\n 员工岗位：秘书");  base.Show();  }  }    class Cleaner : Employee  {  private static double salary = 1000;  public Cleaner(string name, string address, DateTime birthday) : base(name, address, birthday)  {  }  public static double Salary  {  get  {  return salary;  }  set  {  salary = value;  }  }  public override double GetSalary()  {  return Salary;  }  public override void Show()  {  Console.WriteLine("\n 员工岗位：清洁工");  base.Show();  }  }    class Program  {  static void ShowDetailOf(Employee employee)  {  employee.Show();  }  static void Main(string[] args)  {  Console.WriteLine("请以空格为分隔符输入程序员的姓名 地址 出生日期(yyyy-MM-dd) 本月提成：");  string[] tmp = Console.ReadLine().Split(' ');  Programmer programmer = new Programmer(tmp[0], tmp[1], Convert.ToDateTime(tmp[2],Employee.DATE\_FORMAT), Convert.ToDouble(tmp[3]));  Console.WriteLine("请以空格为分隔符输入高层管理的姓名 地址 出生日期(yyyy-mm-dd) 本月提成：");  tmp = Console.ReadLine().Split(' ');  Administrator administrator = new Administrator(tmp[0], tmp[1], DateTime.Parse(tmp[2], Employee.DATE\_FORMAT), Convert.ToDouble(tmp[3]));  Console.WriteLine("请以空格为分隔符输入秘书的姓名 地址 出生日期(yyyy-mm-dd)：");  tmp = Console.ReadLine().Split(' ');  Secretary secretary = new Secretary(tmp[0], tmp[1], DateTime.Parse(tmp[2], Employee.DATE\_FORMAT));  Console.WriteLine("请以空格为分隔符输入清洁工的姓名 地址 出生日期(yyyy-mm-dd)：");  tmp = Console.ReadLine().Split(' ');  Cleaner cleaner = new Cleaner(tmp[0], tmp[1], DateTime.Parse(tmp[2], Employee.DATE\_FORMAT));  ShowDetailOf(programmer);  ShowDetailOf(administrator);  ShowDetailOf(secretary);  ShowDetailOf(cleaner);  Console.ReadKey();  }  }  }  **四、实验调试过程及运行结果**  **实验2\_1\_1**    **实验2\_1\_2**  **输入：**  30  30 50  30 50    **实验2\_1\_3**  **输入：**  挪威的森林 村上春树 2  人性的优点 卡耐基 1  揭秘Angular2 广发证券互联网金融技术团队 3  海边的卡夫卡 村上春树 2  人性的弱点 卡耐基 3  Java编程思想 陈昊鹏 1        **实验2\_2\_1**  **输入：**  张三 天津 1996-01-02 300  李四 北京 1997-03-12 500  王五 深圳 1994-11-23  马六 杭州 1995-08-14    **五、实验总结：**  **1. 实验1\_1\_2中计算圆锥体积的时候1/3忘记添加f，导致结果一直为0，断点调试了一段时间才解决，这个坑要谨记。**  **2. 当使用基类对象调用和派生类中使用new覆盖的同名方法时，是无法直接调用到的，会发现调用到的还是基类的方法，因为是使用基类对象的参考调用的，自然会优先在基类中开始寻找该方法，如果不声明为virtual的话肯定就直接调用基类中的同名方法了，new起到的作用仅仅是提示开发者基类中有同名方法被覆盖了，并不能起到virtual的作用。**  **3. 好吧，现在发现C#的格式化输出有时候还是挺好用的，比如表示货币时的C2…**  **4. 想要格式化转换日期也是可以有现成的方法可以调用的，不过调用起来不如Java的SimpleDateFormat类方便，但是起码比JS中都没有的方便…** |

# 实验报告

**姓名**： 胡启明 **学号**： 164547 **日期：**

**实验名称**： **实验三 接口编程 成绩：**

|  |
| --- |
| **一、实验目的及要求**  **【实验目的】**  1. 了解什么是接口，接口和抽象类的异同，掌握接口的方法实现，接口方法的重定义。  **【实验要求】**  1. 写出程序，并调试程序，要给出测试数据和实验结果。  2. 整理上机步骤，总结经验和体会。  3. 完成实验日志和上交程序。  **二、实验环境**  **Windows XP操作系统，Visual Studio 2005及以上版本**  **三、实验调试过程及运行结果**  1. 本程序中的接口包含方法的构成是哪些？  对象的比较方法：int CompareTo(IComparable comp);  2. 实现接口的类包含哪些元素？  构造方法：public TimeSpan()、public TimeSpan(uint initialSeconds)  覆盖的接口的比较方法：public int CompareTo(IComparable comp)  3. 类实现接口方法的参数如何变换实现的？  TimeSpan compareTime = (TimeSpan)comp;  通过强制类型转换将实现了将接口类型的对象转换为本类型对象。  4. 给出程序的输出结果。    **四、实验总结：对上机实践结果进行分析，问题回答及反思，心得体会及改进。**  接口的好处就是不管什么类型的对象，只要实现了统一的接口，就能将该对象传入接收这一接口类型参数的方法中去做统一处理，实现了面向对象中的多态的观点。 |

# 实验报告

**姓名**： 胡启明 **学号**： 164547 **日期：**

**实验名称**： **实验四 委托编程 成绩：**

|  |
| --- |
| **一、实验目的及要求**  **【实验目的】**  1.了解委托的概念，掌握委托的使用；  **【实验要求】**  1. 理解委托应用的程序；  2. 写出委托应用程序；  3. 调试程序，要给出测试数据和实验结果。  4. 整理上机步骤，总结经验和体会。  5. 完成实验日志和上交程序。  **二、实验环境**  **Windows XP操作系统，Visual Studio 2005及以上版本**  **三、实验源程序及关键代码解释**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Exp\_4\_1  {  // 声明委托，可接受所有类似double(double,double)形式的方法类型  public delegate double Calculation(double param1,double param2);  /// <summary>  /// 定义计算器类  /// 进行两个操作数的运算  /// 定义了两个double类型成员变量，以及对应构造方法  /// 以及求和和求均值运算的静态方法  /// 和最后的统一计算方法（相当于点击等号键）  /// </summary>  class Calculator  {  public double Param1 { get; set; }  public double Param2 { get; set; }  public Calculator(double param1, double param2)  {  Param1 = param1;  Param2 = param2;  }  /// <summary>  /// 两数求和  /// </summary>  /// <param name="param1">求和的第一个操作数</param>  /// <param name="param2">求和的第二个操作数</param>  /// <returns>两数之和</returns>  public static double Sum(double param1,double param2)  {  return param1 + param2;  }  /// <summary>  /// 两数求均值  /// </summary>  /// <param name="param1">求均值的第一个操作数</param>  /// <param name="param2">求均值的第二个操作数</param>  /// <returns>两数的均值</returns>  public static double Aver(double param1, double param2)  {  return (param1 + param2) / 2;  }  /// <summary>  /// 计算方法，使用传入的Calculation委托类型的方法计算自身的Param1和Param2并返回计算结果  /// </summary>  /// <param name="Cal"></param>  /// <returns>通过相应类型方法的委托的计算结果</returns>  public double DoCalculate(Calculation Cal)  {  return Cal(Param1, Param2);  }  }  class Program  {    static void Main(string[] args)  {  while (true)  {  Console.WriteLine("请以空格为分隔在同一行写入两个操作数: ");  string[] temp = Console.ReadLine().Split(' ');  Calculator calculator = new Calculator(Convert.ToDouble(temp[0]), Convert.ToDouble(temp[1]));  // 进行求和计算  Console.WriteLine("两操作数之和为: {0}", calculator.DoCalculate(Calculator.Sum));  // 进行求均值计算  Console.WriteLine("两操作数的均值为: {0}\n", calculator.DoCalculate(Calculator.Aver));  }    }  }  }  **四、实验调试过程及运行结果**  数据输入：  5 9  15 28  33 49  1265 8538    **五、实验总结：对上机实践结果进行分析，问题回答及反思，心得体会及改进。**  通过本实验对委托有了如下理解，就是可以将方法当做参数进行传递，然后在参数所在的方法中进行统一调用，实现多态。而Java中想要实现相似操作则只能通过接口或抽象类的方式进行，C#的这种方式就相当于简化了只有一个方法的接口的实现和调用。 |

# 实验报告

**姓名**： 胡启明 **学号**： 164547 **日期：**

**实验名称**： **实验五 异常处理编程 成绩：**

|  |
| --- |
| **一、实验目的及要求**  **【实验目的】**   1. 理解什么是异常及异常可能产生的原因； 2. 掌握异常抛出和捕获语句； 3. 理解专有catch块的使用；   **【实验要求】**  1. 调试程序，要给出测试数据和实验结果。  2. 整理上机步骤，总结经验和体会。  3. 完成实验日志和上交程序。  **二、实验环境**  **Windows XP操作系统，Visual Studio 2005及以上版本**  **三、实验源程序及关键代码解释**  using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Exp\_5\_1  {  class Meteorologist  {  private int[] rainfall;  public Meteorologist()  {  // 初始化12个数据  rainfall = new int[] { 145, 25, 389, 64, 625, 364, 485, 383, 199, 1103, 1511, 2123 };  }  // 定义一个访问器用来访问rainfall数组  public int this[int index]  {  set  {  rainfall[index] = value;  }  get  {  return rainfall[index];  }  }  /// <summary>  /// 通过index返回rainfall降雨量数值，如超出索引范围则给出提示  /// </summary>  /// <param name="index">降雨量索引</param>  /// <returns>对应序号的降雨量，如抛出异常则返回0</returns>  public int GetRainfall(int index)  {  int item = 0;  try  {  item = this[index];  }  catch (IndexOutOfRangeException e)  {  Console.WriteLine("序号超出访问范围！降雨量索引范围0~{0},您的输入:{1}", rainfall.Length-1, index);  }  return item;  }  }  class Program  {  static void Main(string[] args)  {  Meteorologist mete = new Meteorologist();  while (true)  {  Console.WriteLine("请输入您要访问的降雨量观测序号: ");  int index = Convert.ToInt16(Console.ReadLine());  Console.WriteLine("序号为{0}的记录降雨量为{1}\n", index, mete.GetRainfall(index));  }  }  }  }  **四、实验调试过程及运行结果**    **五、实验总结：对上机实践结果进行分析，问题回答及反思，心得体会及改进。**  这里的异常处理和Java基本无异，使用也很得心应手。 |

# 实验报告

**姓名**： 胡启明 **学号**： 164547 **日期：**

**实验名称**： **实验一 文本编辑器的设计 成绩：**

|  |
| --- |
| **一、实验目的及要求**  **【实验目的】**   1. 熟悉VisualC#.NET的可视化界面。   2、掌握控件：文本类控件、按钮类控件、框架类控件、列表框、视图类控件、滚动类控件、工具栏和状态栏控件、定时器控件、菜单控件、对话框类控件的使用。  **【实验要求】**  1、开发环境：Visual Studio2005及以上  2、实现文本编辑器功能  **二、实验环境**  **Windows XP操作系统，Visual Studio 2005及以上版本**  **三、实验源程序及关键代码解释**  **四、实验调试过程及运行结果**  **五、实验总结：对上机实践结果进行分析，问题回答及反思，心得体会及改进。** |