**新建项目**

**计算机学院（软件学院）实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学生姓名** | 陈亚龙 | **学号** | 171530431 | **实验成绩** |  |
| **专业** | 软件工程 | **班级** | 软工四班 | **实验日期** | 19年11月16日 |
| **课程名称** | C#高级程序设计 | | | **任课教师** | 彭伟国 |
| **实验名称** | 面向对象编程进阶 | | | **实验序号** | 4 |
| **实验地点** | S409 | **实验台号** | 1 | **指导教师** | 彭伟国 |
| 1. **实验目的及要求**   1．会使用类的继承机制派生子类。 2．会使用虚函数、抽象类、接口实现多态。 3．会使用List和Dictionary泛型集合类组织数据。 4．会使用ArryList和Hashtable非泛型集合类组织数据。 5．掌握委托的定义和使用方法。 6．会使用泛型知识设计程序。 7．会使用C#反射机制获取类信息。 | | | | | |
| **二、实验内容（或实验原理、实验拓扑）**  第一次：   1. 写一个Student类和一个Teacher类,他们都有一个打招呼的方法,不同的是Studetn打招呼是说"大家好,我叫XX,我今年XX岁了,我的爱好是XXX",Teacher的打招呼的方法是说"大家好,我叫XX,我今年XX岁了,我已经工作XX年了"? 2. 自己定义一个父类Person,两个子类Teacher和Student类.练习类内部构造器的调用，练习子类调用父类的构造器 3. 自己试试,子类可以自动转父类,父类转子类要通过强转.为了不出错,可以先通过is判断或用as转换   第二次：   1. 定义一个动物类,在动物类中有一个叫的抽象方法. 写两个子类,一个猫一个狗,继承自动物类,并实现相应的抽象方法.（抽象类） 2. 在一个程序中，既想实现按年龄排序，又想实现按姓名排序，还要实现按人气值排序，怎么办？   第三次   1. 创建一个集合，里面添加一些数字，求平均值与和，最大值，最小值，并在控制台显示输出结果。 2. 写一个长度为10的集合，要求在里面随机地存放10个数字，并在控制台上显示输出这10个数字。（提示：用到Random随机数类中的Next方法，首先创建随机数对象，然后调用Next方法）。 3. 定义一个集合类（学生类），不仅可以通过下标查找学生，还可以通过姓名查找（设姓名没有重复)   第四次   1. 将一个字符串数组中每个元素都转换成大写、小写，每个元素两边都加上双引号。   第五次   1. 一个简单的类，包含一个有参数的构造器，一个GetValue的方法，一个Value属性，运用反射知识通过方法的名称来得到方法并且调用之。 | | | | | |
| **三、实验设备与环境**  **Windows7+Visual studio 2012** | | | | | |
| **四、实验设计方案（包括实验步骤、设计思想、算法描述或开发流程等）**  **一.** 构造两个类，一个是学生类student，一个是教师类Teaxher。然后，我们进行类实现和构建。我在Student类中定义了三个私有字段：姓名，年龄，爱好，在Teacher类中也定义了私有字段：姓名，年龄，工作时间。  **二.** 根据题目要求我们需要按年龄排序，又想实现按姓名排序，还要实现按人气值排序。当相等时就往后找，按对应的顺序进行排序。我这里使用一个类paixu，通过继承接口  **三.** 在student类中，使用字典作为是由字段，然后在使用构造函数为其赋值，最后使用共有的方法进行数据的输出，按照题目要求通过学号挺好使用只有姓名需要我们去进行相应的函数实现  **四.** 声明委托，**public delegate string DeleteCon(String T);**  在写声明时要与将要使用的方法的签名相同，然后我们实现相应的函数方法，最后直接调用就可以了。  **五.** 定义一个类，包含一个有参数的构造器，一个GetValue的方法，一个Value属性，运用反射知识通过方法的名称来得到方法并且调用之 | | | | | |

|  |
| --- |
| **五、实验结果（包括设计效果、测试数据、运行结果等）** 第一次  1. 写一个Student类和一个Teacher类,他们都有一个打招呼的方法,不同的是Studetn打招呼是说"大家好,我叫XX,我今年XX岁了,我的爱好是XXX",Teacher的打招呼的方法是说"大家好,我叫XX,我今年XX岁了,我已经工作XX年了"? 2. 自己定义一个父类Person,两个子类Teacher和Student类.练习类内部构造器的调用，练习子类调用父类的构造器 3. 自己试试,子类可以自动转父类,父类转子类要通过强转.为了不出错,可以先通过is判断或用as转换  第二次  1. 定义一个动物类,在动物类中有一个叫的抽象方法. 写两个子类,一个猫一个狗,继承自动物类,并实现相应的抽象方法.（抽象类） 2. 在一个程序中，既想实现按年龄排序，又想实现按姓名排序，还要实现按人气值排序，怎么办？ QQ浏览器截图20191116175853  第三次  1. 创建一个集合，里面添加一些数字，求平均值与和，最大值，最小值，并在控制台显示结果QQ浏览器截图20191116180109。 2. 写一个长度为10的集合，要求在里面随机地存放10个数字，并在控制台上显示输出这10个数字。（提示：用到Random随机数类中的Next方法，首先创建随机数对象，然后调用Next方法）。 3. 定义一个集合类（学生类），不仅可以通过下标查找学生，还可以通过姓名查找（设姓名没有重复)  第四次  1. 将一个字符串数组中每个元素都转换成大写、小写，每个元素两边都加上双引号。 |
| **六、实验小结（包括收获、心得体会、注意事项、存在问题及解决办法、建议等）**  委托的概念：委托是用户自定义的类，它定义了方法的类型。储存的是一系列具有相同参数和返回类型方法的地址列表，调用委托时，此委托列表的所有方法都将被执行。  委托类型的定义：委托类型必须在被用来创建变量以及类型对象之前声明。 |

|  |
| --- |
| **七、附录（包括作品、流程图、源程序及命令清单等）**   * using System; * namespace \_1\_1 * { * class Program * { * static void Main(string[] args) * { * Student stu = new Student() * { * Name = "Coco", * Age = 22, * Hobby = "coding" * }; * stu.hello(); * Teacher t = new Teacher(); * t.Name = "Mr. Smith"; * t.Age = 42; * t.WorkingAge = 15; * t.hello(); * Console.WriteLine(); * //里氏转换：1)子类可以赋值给父类 * Person p = new Student("Join", 18, "Running"); * //Student ss = (Student)p; 有风险 * //2)如果父类中装的是子类对象，那么可以将这个父类转换为子类对象 * //3)子类对象可以调用父类中的成员，但是父类对象永远只能调用自己的成员 * //is用法：表示类型转换 转换成功返回True，否则False * //强制类型转换 由大类型转换为小类型 有风险 * if (p is Student) * { * Console.WriteLine("可以转换，正在进行转换"); * //as用法 表示类型转化 转换成功返回对应的对象 否则返回一个null * Student t2 = p as Student; * t2.hello(); * } * else * { * Console.WriteLine("不可以转换"); * } * Console.WriteLine("\nPress any key to quit."); * Console.ReadKey(); * } * } * } * Person.cs * using System; * namespace \_1\_1 * { * public class Person * { * public string Name; * public double Age; * public Person(string name, double age) * { * this.Name = name; * this.Age = age; * } * public Person() * { * this.Name = "Null"; * this.Age = 0; * } * public void hello() * { * Console.WriteLine("我是人类"); * } * } * } * Student.cs * using System; * namespace \_1\_1 * { * class Student: Person * { * public string Hobby; * public Student(): base() * { * this.Hobby = "Null"; * } * public Student(string name, double age, string hobby) : * base(name, age) * { * this.Hobby = hobby; * } * public void hello() * { * Console.WriteLine("我叫{0}，我今年{1}岁了，我的爱好是{2}。", this.Name, this.Age, Hobby); * } * } * } * Teacher.cs * using System; * namespace \_1\_1 * { * class Teacher : Person * { * public double WorkingAge; * public Teacher() { } * public Teacher(string name, double age, double workingAge) : * base(name, age) * { * this.WorkingAge = workingAge; * } * public void hello() * { * Console.WriteLine("我叫{0}，我今年{1}岁了，我已经工作{2}年了。", this.Name, this.Age, WorkingAge); * } * } * }   二 **Program.cs**   1. using System; 2. //定义一个动物类,在动物类中有一个叫的抽象方法. 写两个子类,一个猫一个狗,继承自动物类,并实现相应的抽象方法.（抽象类） 3. namespace \_2 4. { 5. abstract class Animal 6. { 7. public abstract void cry(); 8. } 9. class Cat : Animal 10. { 11. public override void cry() 12. { 13. Console.WriteLine("喵喵！"); 14. } 15. } 16. class Dog : Animal 17. { 18. public override void cry() 19. { 20. Console.WriteLine("汪汪!"); 21. } 22. } 23. class Program 24. { 25. static void Main(string[] args) 26. { 27. Cat cat = new Cat(); 28. Dog dog = new Dog(); 29. cat.cry(); 30. dog.cry(); 31. Console.ReadLine(); 32. } 33. } 34. }   三.  using System;   1. using System.Collections.Generic; 2. namespace \_2\_2 3. { 4. class Program 5. { 6. static void Main(string[] args) 7. { 8. List<Teacher> list = new List<Teacher>(); 9. list.Add(new Teacher(1, "张三", 10000,100)); 10. list.Add(new Teacher(2, "张三", 10000, 101)); 11. list.Add(new Teacher(3, "宋江", 10000,101)); 12. list.Add(new Teacher(4, "李四", 5000,102)); 13. list.Add(new Teacher(5, "王五", 1000,103)); 14. foreach (Teacher t in list) 15. Console.WriteLine(t); 16. Console.WriteLine("排序后："); 17. list.Sort();//IComparable 18. foreach (Teacher t in list) 19. Console.WriteLine(t); 20. Console.ReadKey(); 21. } 22. } 23. public class Teacher : IComparable<Teacher> 24. { 25. public int Id; 26. public string Name; 27. public decimal Salary; 28. public int PopularityValues; 29. public Teacher(int id, string name, decimal salary, int popularityValues) 30. { 31. this.Name = name; 32. this.Id = id; 33. this.Salary = salary; 34. this.PopularityValues = popularityValues; 35. } 36. public override string ToString() 37. { 38. return "id:" + this.Id + "\tname:" + this.Name + "\tsalary:" + this.Salary + "\tpopularityValues:"+this.PopularityValues; 39. } 40. public int CompareTo(Teacher other) 41. { 42. if (this.Salary == other.Salary) 43. { 44. if (this.Name == other.Name) 45. return (int)(this.PopularityValues - other.PopularityValues); 46. return this.Name.CompareTo(other.Name); 47. } 48. return (int)(this.Salary - other.Salary); 49. } 50. } 51. }   四. **Program.cs**   1. using System; 2. using System.Collections; 3. //1. 创建一个集合，里面添加一些数字，求平均值与和，最大值，最小值，并在控制台显示输出结果。 4. namespace \_3 5. { 6. class Program 7. { 8. static void Main(string[] args) 9. { 10. //创建一个集合 11. ArrayList list = new ArrayList(); 12. //向集合添加一些数字 13. list.AddRange(new int[] { 1, 8, 7, 4, 5, 6, 3, 2, 9 }); 14. //新建sum变量来存储和 15. int sum = 0; 16. //新建一个max来存储最大值 17. int max = (int)list[0]; 18. //新建一个min来存储最小值 19. int min = (int)list[1]; 20. //通过一个循环来赋值 21. for (int i = 0; i < list.Count; i++) 22. { 23. //list[i]是object类型，通过里氏转换法强转成int类型 24. if ((int)list[i] > max) 25. { 26. max = (int)list[i]; 27. } 28. if ((int)list[i] < min) 29. { 30. min = (int)list[i]; 31. } 32. sum += (int)list[i]; 33. } 34. //分别输出max,min,sum和avg 35. Console.WriteLine("max="+max); 36. Console.WriteLine("min=" + min); 37. Console.WriteLine("sum=" + sum); 38. Console.WriteLine("avg=" + sum / list.Count); 39. Console.ReadKey(); 40. } 41. } 42. }   五. **Program.cs**   1. using System; 2. using System.Collections; 3. //2. 写一个长度为10的集合，要求在里面随机地存放10个数字，并在控制台上显示输出这10个数字。（提示：用到Random随机数类中的Next方法，首先创建随机数对象，然后调用Next方法）。 4. namespace \_3\_2 5. { 6. class Program 7. { 8. static void Main(string[] args) 9. { 10. Random r = new Random(); 11. int cnt = 0; 12. ArrayList al = new ArrayList(); 13. while (cnt<10) 14. { 15. cnt += 1; 16. al.Add(r.Next()); 17. } 18. foreach (var i in al) 19. { 20. Console.WriteLine(i); 21. } 22. Console.ReadKey(); 23. } 24. } 25. }   六． **MyList.cs**   1. using System; 2. namespace \_3\_3 3. { 4. class MyList 5. { 6. private Student[] array; 7. private int count; 8. public MyList(int size) 9. { 10. if (size >= 0) 11. { 12. array = new Student[size]; 13. } 14. } 15. public MyList() 16. { 17. array = new Student[0]; 18. } 19. //Capacity属性获取容量大小 20. public int Capacity 21. { 22. get { return array.Length; } 23. } 24. //Count属性访问元素个数 25. public int Count 26. { 27. get { return count; } 28. } 29. //Add（）方法添加元素 30. public void Add(Student item) 31. { 32. if (Count == Capacity) 33. { 34. if (Capacity == 0) 35. { 36. array = new Student[4]; 37. } 38. else 39. { 40. var newArray = new Student[Capacity \* 2]; 41. Array.Copy(array, newArray, Count); 42. array = newArray; 43. } 44. } 45. array[Count] = item; 46. count++; 47. } 48. //索引器（通过一个Index查找数组中的某个元素） 49. public Student GetItem(int Index) 50. { 51. if (Index >= 0 && Index < Count) 52. { 53. return array[Index]; 54. } 55. else 56. { 57. throw new Exception("索引超出范围"); 58. } 59. } 60. //[index] 访问元素 61. public Student this[int index] 62. { 63. get 64. { 65. return GetItem(index); 66. } 67. set 68. { 69. if (index >= 0 && index < Count) 70. { 71. array[index] = value; 72. } 73. else 74. { 75. throw new Exception("索引超出范围"); 76. } 77. } 78. } 79. public Student this[string name] 80. { 81. get 82. { 83. int index = IndexOf(name); 84. return GetItem(index); 85. } 86. set 87. { 88. int index = IndexOf(name); 89. if (index >= 0 && index < Count) 90. { 91. array[index] = value; 92. } 93. else 94. { 95. throw new Exception("索引超出范围"); 96. } 97. } 98. } 99. //Insert（）插入元素 100. public void Insert(int index, Student item) 101. { 102. if (Count == Capacity) 103. { 104. if (Capacity == 0) 105. { 106. array = new Student[4]; 107. } 108. else 109. { 110. var newArray = new Student[Capacity \* 2]; 111. Array.Copy(array, newArray, Count); 112. array = newArray; 113. } 114. } 115. for (int j = count - 1; j >= index; j--) 116. { 117. array[j + 1] = array[j]; 118. } 119. array[index] = item; 120. count++; 121. } 122. //IndexOf（）方法取得一个元素所在列表中的索引位置（从前往后搜索） 123. public int IndexOf(Student item) 124. { 125. for (int i = 0; i < count; i++) 126. { 127. if(array[i].Equals(item)) 128. { 129. return i; 130. } 131. } 132. return -1; 133. } 134. public int IndexOf(string name) 135. { 136. for (int i = 0; i < count; i++) 137. { 138. if (array[i].Name == name) 139. { 140. return i; 141. } 142. } 143. return -1; 144. } 145. //从后往前搜索，搜到满足条件就停止没有找到返回-1 146. public int LastIndexOf(Student item) 147. { 148. for (int i = count - 1; i >= 0; i--) 149. { 150. if (array[i].Equals(item)) 151. { 152. return i; 153. } 154. } 155. return -1; 156. } 157. //对列表中的元素进行从小到大的排序 158. public void Sort() 159. { 160. for (int j = 0; j < count - 1; j++) 161. { 162. for (int i = 0; i < count - 1 - j; i++) 163. { 164. if (array[i].CompareTo(array[i + 1]) > 0) 165. { 166. Student temp = array[i]; 167. array[i] = array[i + 1]; 168. array[i + 1] = temp; 169. } 170. } 171. } 172. } 173. } 174. }   **Person.cs**  using System;  namespace \_3\_3  {  class Person  {  public string Name;  public int ID;  public static int IDCnt = 1;  public Person(string name)  {  this.ID = IDCnt;  IDCnt += 1;  this.Name = name;  }  public void hello()  {  Console.WriteLine("我是人类");  }  }  }  **Program.cs**  using System;  //3. 定义一个集合类（学生类），不仅可以通过下标查找学生，还可以通过姓名查找（设姓名没有重复)  namespace \_3\_3  {  class Program  {  static void Main(string[] args)  {  Student stu = new Student("Coco", 99);  Student stu2 = new Student("Jone", 90);  MyList myList = new MyList();  myList.Add(stu);  myList.Add(stu2);  Console.WriteLine("通过下标查找学生：");  for (int i = 0; i < myList.Count; i++)  {  myList[i].hello();  }  Console.WriteLine("\n\n通过姓名查找学生：");  myList["Jone"].hello();  myList["Coco"].hello();  Console.ReadKey();  }  }  }  **Student.cs**  using System;  namespace \_3\_3  {  class Student : Person  {  public float Score;  public Student(string name, float score) :  base(name)  {  this.Score = score;  }  public new void hello()  {  Console.WriteLine("我的学号是{0}，我叫{1}，我的分数是{2}。", ID, Name, Score);  }  public float CompareTo(Student other)  {  return (Score - other.Score);  }  }  }  四. **Program.cs**   1. using System; 2. //将一个字符串数组中每个元素都转换成大写、小写，每个元素两边都加上双引号。 3. namespace \_4 4. { 5. class Program 6. { 7. static void Main(string[] args) 8. { 9. string str = "aBc"; 10. Console.WriteLine(str); 11. Console.WriteLine("\n全转成大写的："); 12. string u = str.ToUpper(); 13. Console.WriteLine(u); 14. Console.WriteLine("\n全转成小写的："); 15. string l = str.ToLower(); 16. Console.WriteLine(l); 17. Console.WriteLine("\n添加引号："); 18. string q = "\""+str +"\""; 19. Console.WriteLine(q); 20. Console.ReadKey(); 21. } 22. } 23. }   五. **Program.cs**   1. using System; 2. using System.Reflection; 3. //一个简单的类，包含一个有参数的构造器，一个GetValue的方法，一个Value属性，运用反射知识通过方法的名称来得到方法并且调用之。 4. namespace \_5 5. { 6. class Program 7. { 8. static void Main(string[] args) 9. { 10. Type t = typeof(SimpleClass); 11. object o = Activator.CreateInstance(t, "test"); 12. MethodInfo mi = t.GetMethod("getValue"); 13. mi.Invoke(o, null); 14. Console.ReadKey(); 15. } 16. public class SimpleClass 17. { 18. public string Value; 19. public SimpleClass(string v) 20. { 21. Value = v; 22. Console.WriteLine("正在初始化对象，已为Value成功赋值。"); 23. } 24. public void getValue() 25. { 26. Console.WriteLine("Value=" + Value); 27. } 28. } 29. }   } |
|  |