**新建项目**

**计算机学院（软件学院）实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学生姓名** | 徐可可 | **学号** | 171530425 | **实验成绩** |  |
| **专业** | 软件工程 | **班级** | 软工四班 | **实验日期** | 19年11月06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** | 25 | **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**  I7 7700HQ+16G内存+48TB分布式硬盘的笔记本电脑、Windows 10 1903政府版、Visual Studio 2019 企业版、.NET Framework 4.7.2 | | | | | |
| **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）** | | | | | |

|  |
| --- |
| **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 在这里插入图片描述  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） 在这里插入图片描述 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？ 在这里插入图片描述  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。 在这里插入图片描述 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）。 在这里插入图片描述 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) 在这里插入图片描述  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。 在这里插入图片描述  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。 在这里插入图片描述 |
| **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）**  委托和事件在.NET Framework中的应用非常广泛，然而，较好地理解委托和事件对很多接触C#时间不长的人来说并不容易。  c#委托和事件它们就像是一道槛儿，过了这个槛的人，觉得真是太容易了，而没有过去的人每次见到委托和事件就觉得心里憋得慌，浑身不自在。  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。  委托类型的声明：   1. 以delegate关键字开头。 2. 返回类型+委托类型名+参数列表 |

|  |
| --- |
| **七、附录（包括作品、流程图、源程序及命令清单等）**  **本实验报告同步更新于：**<https://blog.csdn.net/COCO56/article/details/102928850>  在我的CSDN博文中查看代码更美观并且复制代码更方便。 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  * Program.cs * 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); * } * } * }  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） **Program.cs** 2. using System; 3. //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） 4. namespace \_2 5. { 6. abstract class Animal 7. { 8. public abstract void cry(); 9. } 10. class Cat : Animal 11. { 12. public override void cry() 13. { 14. Console.WriteLine("喵喵！"); 15. } 16. } 17. class Dog : Animal 18. { 19. public override void cry() 20. { 21. Console.WriteLine("汪汪!"); 22. } 23. } 24. class Program 25. { 26. static void Main(string[] args) 27. { 28. Cat cat = new Cat(); 29. Dog dog = new Dog(); 30. cat.cry(); 31. dog.cry(); 32. Console.ReadLine(); 33. } 34. } 35. } 36. 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？ **Program.cs** 37. using System; 38. using System.Collections.Generic; 39. namespace \_2\_2 40. { 41. class Program 42. { 43. static void Main(string[] args) 44. { 45. List<Teacher> list = new List<Teacher>(); 46. list.Add(new Teacher(1, "张三", 10000,100)); 47. list.Add(new Teacher(2, "张三", 10000, 101)); 48. list.Add(new Teacher(3, "宋江", 10000,101)); 49. list.Add(new Teacher(4, "李四", 5000,102)); 50. list.Add(new Teacher(5, "王五", 1000,103)); 51. foreach (Teacher t in list) 52. Console.WriteLine(t); 53. Console.WriteLine("排序后："); 54. list.Sort();//IComparable 55. foreach (Teacher t in list) 56. Console.WriteLine(t); 57. Console.ReadKey(); 58. } 59. } 60. public class Teacher : IComparable<Teacher> 61. { 62. public int Id; 63. public string Name; 64. public decimal Salary; 65. public int PopularityValues; 66. public Teacher(int id, string name, decimal salary, int popularityValues) 67. { 68. this.Name = name; 69. this.Id = id; 70. this.Salary = salary; 71. this.PopularityValues = popularityValues; 72. } 73. public override string ToString() 74. { 75. return "id:" + this.Id + "\tname:" + this.Name + "\tsalary:" + this.Salary + "\tpopularityValues:"+this.PopularityValues; 76. } 77. public int CompareTo(Teacher other) 78. { 79. if (this.Salary == other.Salary) 80. { 81. if (this.Name == other.Name) 82. return (int)(this.PopularityValues - other.PopularityValues); 83. return this.Name.CompareTo(other.Name); 84. } 85. return (int)(this.Salary - other.Salary); 86. } 87. } 88. }  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。 **Program.cs** 2. using System; 3. using System.Collections; 4. //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。 5. namespace \_3 6. { 7. class Program 8. { 9. static void Main(string[] args) 10. { 11. //创建一个集合 12. ArrayList list = new ArrayList(); 13. //向集合添加一些数字 14. list.AddRange(new int[] { 1, 8, 7, 4, 5, 6, 3, 2, 9 }); 15. //新建sum变量来存储和 16. int sum = 0; 17. //新建一个max来存储最大值 18. int max = (int)list[0]; 19. //新建一个min来存储最小值 20. int min = (int)list[1]; 21. //通过一个循环来赋值 22. for (int i = 0; i < list.Count; i++) 23. { 24. //list[i]是object类型，通过里氏转换法强转成int类型 25. if ((int)list[i] > max) 26. { 27. max = (int)list[i]; 28. } 29. if ((int)list[i] < min) 30. { 31. min = (int)list[i]; 32. } 33. sum += (int)list[i]; 34. } 35. //分别输出max,min,sum和avg 36. Console.WriteLine("max="+max); 37. Console.WriteLine("min=" + min); 38. Console.WriteLine("sum=" + sum); 39. Console.WriteLine("avg=" + sum / list.Count); 40. Console.ReadKey(); 41. } 42. } 43. } 44. 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）。 **Program.cs** 45. using System; 46. using System.Collections; 47. //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）。 48. namespace \_3\_2 49. { 50. class Program 51. { 52. static void Main(string[] args) 53. { 54. Random r = new Random(); 55. int cnt = 0; 56. ArrayList al = new ArrayList(); 57. while (cnt<10) 58. { 59. cnt += 1; 60. al.Add(r.Next()); 61. } 62. foreach (var i in al) 63. { 64. Console.WriteLine(i); 65. } 66. Console.ReadKey(); 67. } 68. } 69. } 70. 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) **MyList.cs** 71. using System; 72. namespace \_3\_3 73. { 74. class MyList 75. { 76. private Student[] array; 77. private int count; 78. public MyList(int size) 79. { 80. if (size >= 0) 81. { 82. array = new Student[size]; 83. } 84. } 85. public MyList() 86. { 87. array = new Student[0]; 88. } 89. //Capacity属性获取容量大小 90. public int Capacity 91. { 92. get { return array.Length; } 93. } 94. //Count属性访问元素个数 95. public int Count 96. { 97. get { return count; } 98. } 99. //Add（）方法添加元素 100. public void Add(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. array[Count] = item; 116. count++; 117. } 118. //索引器（通过一个Index查找数组中的某个元素） 119. public Student GetItem(int Index) 120. { 121. if (Index >= 0 && Index < Count) 122. { 123. return array[Index]; 124. } 125. else 126. { 127. throw new Exception("索引超出范围"); 128. } 129. } 130. //[index] 访问元素 131. public Student this[int index] 132. { 133. get 134. { 135. return GetItem(index); 136. } 137. set 138. { 139. if (index >= 0 && index < Count) 140. { 141. array[index] = value; 142. } 143. else 144. { 145. throw new Exception("索引超出范围"); 146. } 147. } 148. } 149. public Student this[string name] 150. { 151. get 152. { 153. int index = IndexOf(name); 154. return GetItem(index); 155. } 156. set 157. { 158. int index = IndexOf(name); 159. if (index >= 0 && index < Count) 160. { 161. array[index] = value; 162. } 163. else 164. { 165. throw new Exception("索引超出范围"); 166. } 167. } 168. } 169. //Insert（）插入元素 170. public void Insert(int index, Student item) 171. { 172. if (Count == Capacity) 173. { 174. if (Capacity == 0) 175. { 176. array = new Student[4]; 177. } 178. else 179. { 180. var newArray = new Student[Capacity \* 2]; 181. Array.Copy(array, newArray, Count); 182. array = newArray; 183. } 184. } 185. for (int j = count - 1; j >= index; j--) 186. { 187. array[j + 1] = array[j]; 188. } 189. array[index] = item; 190. count++; 191. } 192. //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） 193. public int IndexOf(Student item) 194. { 195. for (int i = 0; i < count; i++) 196. { 197. if(array[i].Equals(item)) 198. { 199. return i; 200. } 201. } 202. return -1; 203. } 204. public int IndexOf(string name) 205. { 206. for (int i = 0; i < count; i++) 207. { 208. if (array[i].Name == name) 209. { 210. return i; 211. } 212. } 213. return -1; 214. } 215. //从后往前搜索，搜到满足条件就停止没有找到返回-1 216. public int LastIndexOf(Student item) 217. { 218. for (int i = count - 1; i >= 0; i--) 219. { 220. if (array[i].Equals(item)) 221. { 222. return i; 223. } 224. } 225. return -1; 226. } 227. //对列表中的元素进行从小到大的排序 228. public void Sort() 229. { 230. for (int j = 0; j < count - 1; j++) 231. { 232. for (int i = 0; i < count - 1 - j; i++) 233. { 234. if (array[i].CompareTo(array[i + 1]) > 0) 235. { 236. Student temp = array[i]; 237. array[i] = array[i + 1]; 238. array[i + 1] = temp; 239. } 240. } 241. } 242. } 243. } 244. }   **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);  }  }  } 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。 **Program.cs** 2. using System; 3. //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。 4. namespace \_4 5. { 6. class Program 7. { 8. static void Main(string[] args) 9. { 10. string str = "aBc"; 11. Console.WriteLine(str); 12. Console.WriteLine("\n全转成大写的："); 13. string u = str.ToUpper(); 14. Console.WriteLine(u); 15. Console.WriteLine("\n全转成小写的："); 16. string l = str.ToLower(); 17. Console.WriteLine(l); 18. Console.WriteLine("\n添加引号："); 19. string q = "\""+str +"\""; 20. Console.WriteLine(q); 21. Console.ReadKey(); 22. } 23. } 24. }  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。 **Program.cs** 2. using System; 3. using System.Reflection; 4. //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。 5. namespace \_5 6. { 7. class Program 8. { 9. static void Main(string[] args) 10. { 11. Type t = typeof(SimpleClass); 12. object o = Activator.CreateInstance(t, "test"); 13. MethodInfo mi = t.GetMethod("getValue"); 14. mi.Invoke(o, null); 15. Console.ReadKey(); 16. } 17. public class SimpleClass 18. { 19. public string Value; 20. public SimpleClass(string v) 21. { 22. Value = v; 23. Console.WriteLine("正在初始化对象，已为Value成功赋值。"); 24. } 25. public void getValue() 26. { 27. Console.WriteLine("Value=" + Value); 28. } 29. } 30. }   } |
|  |