山东大学 软件 学院

高级程序设计语言 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202000300125 | 姓名：贾星宇 | | 班级： 2020级5班 |
| 实验题目：继承和多态 | | | |
| 实验学时：4 | | 实验日期：2020年11月23日 | |
| 实验目的：  1．掌握继承和多态的概念与实现方法。  2．掌握如何从已有的类中派生子类并继承父类。  3．掌握方法的覆盖和重载。 | | | |
| 硬件环境：  Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11GHz | | | |
| 软件环境：  IntelliJ IDEA Community;  Java jdk1.8.0\_261 | | | |
| 实验步骤与内容：  设计一个通用的排序算法，作为验证，自定义学生类，矩形类，可以利用该排序算法对学生按照ID以及Name进行排序；对矩形按照面积进行排序。  一个通用的排序程序只要知道数组中任意两个元素如何比较大小就可以完成排序。因此，设计一个父类，包含 int compare(object a, object b)方法，其他所有子类按照自己的方式实现compare。 | | | |
| 结论分析与体会：   1. 继承是从现有类派生新类的过程，其目的之一是重用现有软件。 2. 继承在父类和子类之间建立是一种“is-a”的关系。 3. Protected可见性提供了允许继承的最大可能的封装性。 4. 使用super调用父类的构造方法。 5. 子类方法可覆盖她所继承的父类的方法，一个类的子类还可以是一个或多个其他类的父类。 6. 所有的java类都直接或间接地有object类派生。 7. 构造方法不能被继承，如果没有显式调用父类的构造方法，系统将自动调用父类的默认构造方法。如果想要显式调用，使用super();（也可能有参数）作为子类构造方法的第一个语句。 8. 当子类和父类中具有相同的方法名，且方法所带有的参数数目和类型都相同时，我们说子类方法覆盖了父类方法。如果想要在子类对象中调用父类的方法，可使用super.方法名（参数）来实现。 9. 变量决定了要使用的方法，而对象决定了使用方法的哪一个定义。 10. Java程序的每一个类都继承toString方法和equals方法。 11. 抽象类代表一种概念，子类基于这种概念来定义方法，因此抽象类不能实例化。 12. 由抽象类派生的子类必须覆盖所有父类的抽象方法，否则该子类仍然是抽象类。   13.一个多态性引用能够随时间变化而指向不同类型的对象。  对于多态性引用，方法调用与方法定义代码的绑定在运行时执行。  基本代码：  class Compare {  public static int compare(Object a,Object b){  return 0;  } }  class Student extends Compare {  String name;  int id;   public Student(String name, int id) {  this.name = name;  this.id = id;  }   public static int compare(Object stu1,Object stu2) {  if ((((Student)stu1).name).equals(((Student)stu2).name)) {  if (((Student)stu1).id < ((Student)stu2).id) return -1;  else if (((Student)stu1).id > ((Student)stu2).id) return 1;  else return 0;  } else return (((Student)stu1).name).compareTo(((Student)stu2).name);  }   public static void sorting(Student[] list) {  int min;  Student temp;  for (int i = 0; i < list.length; i++) {  min = i;  for (int j = i + 1; j < list.length; j++) {  if (*compare*(list[i],list[j]) < 0) min = j;  }  temp = list[min];  list[min] = list[i];  list[i] = temp;  }  }   public String toString() {  return name + "\t" + id;  }   public static void main(String[] args) {  Student[] students = new Student[5];  students[0] = new Student("jason", 1001);  students[1] = new Student("seven", 1002);  students[2] = new Student("audin", 1003);  students[3] = new Student("jason", 1002);  students[4] = new Student("mia", 1005);  *sorting*(students);  for (Student student : students) System.*out*.println(student);  } }  class Shape extends Compare {  int length, wide, sqr;  public Shape(int length, int wide) {  this.length = length;  this.wide = wide;  sqr = length \* wide;  }   public String toString(){  String result=(""+sqr);  return result;  }   public int getSqr() {  return sqr;  }   public static void sorting(Shape[] list){  int min;  Shape temp;  for (int i=0;i<list.length;i++){  min = i;  for (int j=i+1;j<list.length;j++){  if (list[j].sqr<list[min].sqr) min=j;  }  temp = list[min];  list[min] = list[i];  list[i] = temp;  }  }   public static void main(String[] args) {  Shape[] shape = new Shape[5];  shape[0] = new Shape(1,2);  shape[1] = new Shape(2,4);  shape[2] = new Shape(1,1);  shape[3] = new Shape(5,9);  shape[4] = new Shape(2,4);  *sorting*(shape);  for (Shape shapes : shape) System.*out*.println(shapes.getSqr());  } } | | | |