**任务06 类的继承、抽象类和接口**

**认真阅读下列要求，完成代码编写**

1. 提交任务时，将本次任务中的子任务同时进行提交。
2. 提交代码时，请将代码放入Java代码框中提交。
3. 命名规范：

类名首字母大写,如果类名由多个单词组成,每个单词的首字母都要大写。

格式： Student、String、FileReader

变量名、方法名首字母小写，如果名称由多个单词组成，每个单词(第一个单词除外)的首字母都要大写。格式：index、studentNo、studentName

命名过程不能出现中文或拼音。 (影响成绩评定)

1. 提交代码时要保证代码简洁及对齐，可以合理的进行缩进及换行。(影响成绩评定)
2. 注意访问权限修饰符的使用。
3. **每个题涉及到的所有代码，要全部展示在题号的下面，即使此题是其他题的延伸。**

**题目06\_1**

**在画图软件中，可以画出不同大小或颜色的圆形、矩形等几何图形。几何图形之间有许多共同的特征，如它们可以是用某种颜色画出来的，可以是填充的或者不填充的。此外还有些不同的特征，比如，圆形都有半径，可以根据半径计算圆形的面积和周长，矩形都有宽和高，可以根据宽高来计算矩形的面积和周长。**

**1、编写Java程序**

**(1)使用继承机制，分别设计实现抽象基类图形类，派生类圆形类、正方形类、长方形类，要求：**

**①抽象图形类中有属性：画笔颜色、图形是否填充，有方法获取图形面积、获取图形周长；**

**②使用构造方法为其所有成员属性赋初值；**

**③重写toString()方法打印所有属性的信息，并根据文字描述合理设计类的其他成员属性和方法。**

**(2)设计实现画板类，要求：**

**①画一个红色、无填充、长和宽分别为10.0与5.0的长方形，求它的面积和周长，并将这个对象的所有属性信息打印到控制台；**

**②画一个绿色、有填充、半径为3.0的圆形，求它的面积和周长，并将这个对象的所有属性信息打印到控制台；**

**③画一个黄色、无填充、边长为4.0的正方形，求它的面积和周长，并将这个对象的所有属性信息打印到控制台。**

**2、问答题：阐述使用抽象类和不使用抽象类的区别。**

**题目06\_2**

**基于上题背景，设计实现以下程序：**

**1、设计Comparable接口，接口中设计compareTo()方法，用来比较对象，方法的返回值类型设计为int。**

**2、在图形类中实现compareTo()方法，用来比较两个图形的面积大小。**

**3、在测试类中，创建图形类的数组，数组中存放多个圆形、正方形、长方形对象，使用compareTo()方法找出数组中面积最大的图形。**

**提示：比较对象时，可以将compareTo()方法的返回值设置为1、0、-1，来代表比较结果，大、相等或者小。**

**题目06\_3**

**读下面关于文件读写的程序。要求**

**1、给每行代码写注释。**

**2、自己修改程序，完成读取文件后，在原文件的每一行后加上自己的学号和姓名，修改后写入到另一个文件中。**

**package** com.baidu.file;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.Writer;

**public** **class** Demo {

**public** **static** **void** main(String[] args) {

System.***out***.println(*readAndWriteFile*("F:\\old.txt"));

}

**public** **static** **boolean** readAndWriteFile(String name) {

**boolean** b = **false**;

**try** {

Writer w = **new** FileWriter(**new** File("F:\\result.txt"));

FileReader fr = **new** FileReader(name);

BufferedReader bf = **new** BufferedReader(fr);

String str;

**while** ((str = bf.readLine()) != **null**) {

w.write(str + "Lucy2017001" + "\r\n");

}

b = **true**;

bf.close();

fr.close();

w.close();

} **catch** (IOException e) {

e.printStackTrace();

}

**return** b;

}

}