Java程序设计编程作业-3

作业说明

请按照本文档的题目要求和代码框架完成编程作业。代码框架的示例结构如下所示，在`src`文件夹目录下，每一个题目对应一个文件夹(package)。在package内部，通常题目要求实现若干个类(class)，分别对应一个java源代码文件（未完整实现）。此外，每个题目有一个包含main函数的公有类`Test.java`，功能是对题目中待实现的类进行测试。最后，在`src`文件夹目录下有一个`testutil`包，是测试时所用到的工具，同学无需阅读或更改。

```
└─hw1
    ├─README.pdf
    │
    └─src
        ├─problem1
        │    ├─Class1.java (待实现)
        │    ├─Class2.java (待实现)
        │    └─Test.java
        ├─problem2
        │    ├─Class1.java (待实现)
        │    └─Test.java
        ...
        │
        └─testutil
```

同学们完成作业的建议步骤如下：

1. 按照文档，在代码框架中完成题目所要求的功能类的完整实现。
2. 运行每个题目的`Test`类，根据测试结果进行调试。
3. **严格**按照以下文件**结构**和**命名**提交源代码文件，并打包成 zip 格式上传，文件命名为`学号_姓名.zip`。

```
└─2020123456_张三 (学号_姓名)
    ├─report.pdf/txt/md (实验报告，可选)
    │
    └─src
        ├─problem1 (抽象名，根据实际题名决定，小写)
        │    ├─Class1.java （抽象名，根据实际类名决定，首字母大写）
        │    ├─Class2.java
        │    └─... (其他java源代码)
        │
        ├─problem2
        │    ├─Class1.java
        │    └─... (其他java源代码)
        ...
```

请注意：

- 编程作业为个人作业，请独立完成。如果提交的代码参考了资料或个人，请在report中标注来源，否则视为抄

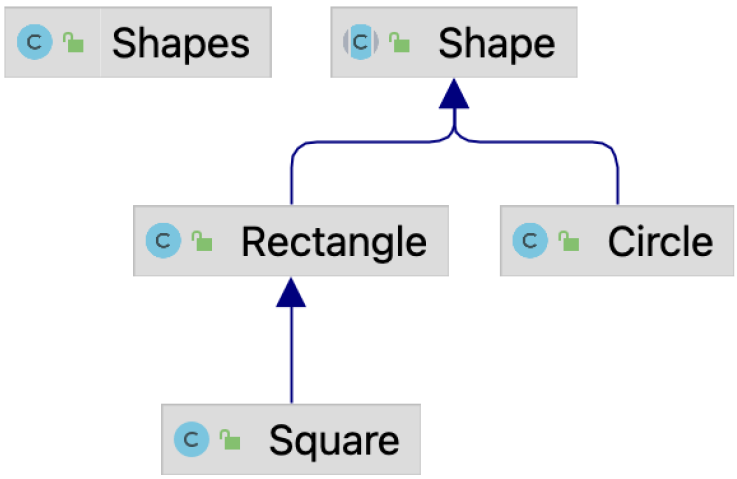
袭。

- 编程作业采用自动化评判，因此不按照规定格式提交作业可能导致错误。
- 请严格按照代码框架和文档要求对成员变量、类、函数、文件命名，函数参数类型、名称、顺序和函数返回值类型请严格按照文档实现，此外类、成员变量和函数访问权限请按照文档要求指定。**Java严格区分大小写，不符合要求的命名会导致错误。**
- 请不要更改代码框架的文件结构和命名，请不要删除代码框架中java源文件开头的包声明`package ...;`。
- 除了题目要求外，可以根据需要实现其他类、函数和成员变量。可以提交其他新增的源代码文件，在新增的**源代码文件开头必须进行包声明**`package ...;`。
- **`Test`类对应的源代码文件以及`testutil`包可以不提交**，同时同学们也可以不改写。无论它们是否被提交或改写，评判过程中会对这两类文件重新覆盖。
- `Test`类包含了测试程序的一些基本测例。无特殊说明的情况下，**作业框架中提供的测例是评判时测例的子集**。最终的评判分数为同学所提交代码在全体测例中通过的比例。同学们可以在`Test`类中根据需要补充测例，进行测试。但是**本次作业提供的测例是评判时的约60%**。
- 实验报告是可选的，内容为除了源代码外额外需要说明的内容，例如参考的资料，复杂题目的实现思路等。**实验报告不影响本次评判。**
- 文档和实验框架的错误、歧义和bug可联系助教wujy22@mails.tsinghua.edu.cn。
- Java API官方在线文档：https://docs.oracle.com/en/java/javase/19/docs/api/

Problem 1: Shape （100”）

在包`shape`中创建基于继承和多态的不同图形的多个功能类。该包所要求的接口见以下**伪代码，本次作业以下要求实现的函数全部为公有函数。**

本次作业共要求实现5个类，其继承关系由以下的类图说明：



请注意，本次作业提供的测例是评判时的约60%，其中`Square`和`Shapes`两个类的测例并没有提供在作业框架中，需要同学在Test.java文件中仿照作业框架的测例自行测试。以下是构造测例的模版说明：

```
TestRunner.runTest(()->{
      Circle c0 = new Circle(1); // 构造对象
      c0.setRadius(2); // 测试成员函数1
      c0.setColor("Blue"); // 测试成员函数2
      System.out.println("Ans: "+c0); // 输出检查结果
      System.out.println("Expected: Circle(Blue)"); // 输出预期结果
      if (c0.toString().equals("Circle(Blue)")) // 比对结果
        System.out.println("Accept");
      else System.out.println("Wrong Answer");
    },1000);
```

1. `Shape`：抽象基类

```
Shape();

/**
* @param c 一个字符串，表示这个图形的颜色
*/
Shape(String c);

/**
* @return 得到图形的颜色，未填充颜色输出"#"
*/
String getColor();

/**
* 填充新的颜色
*/
void setColor(String c);

/**
* @return 表示这个图形是否被着色
*/
boolean isFilled();

/**
* 抽象方法
* @return 得到该图形的面积
*/
double getArea();

/**
* 抽象方法
* @return 得到该图形的周长
*/
double getPerimeter();

/**
```

```
* 可以选择定义为抽象方法或一般成员函数。
* @return 字符串表示，格式为"图形名(颜色)"，例如"Rectangle(Yellow)"。图形名即类名。
*/
String toString();
```

`toString`函数可以任选定义为抽象方法或一般成员函数，但需要支持以下功能：调用`shape.toString`可以返回对应子类的类名+颜色。例如以下代码段：

```
Rectangle rect = new rect();
Shape sh = rect;
System.out.println(sh.toString());
// 示例输出：
// Rectangle(#)
```

类似地，调用`shape.getArea`和`shape.getPerimeter`应该返回对应子类的面积和周长。

2. `Circle`：圆形

```
Circle();

/**
* @param r 半径，保证为正数
*/
Circle(double r);

/**
* @param r 半径，保证为正数
* @param c 颜色
*/
Circle(double r, String c);

/**
* @return 得到半径
*/
double getRadius();

/**
* 设置新的半径
* @param r 半径
*/
void setRadius(double r);

/**
* @return 得到该图形的面积
*/
double getArea();

/**
* @return 得到该图形的周长
```

```
*/
double getPerimeter();

/**
* 可选，基于实现思路选择是否要重写(override)基类Shape中的toString方法。
* @return 字符串表示，格式为"Circle(颜色)"。
*/
String toString();
```

3. `Rectangle`：矩形

```
Rectangle();

/**
* @param h 高，保证为正数
* @param w 宽，保证为正数
*/
Rectangle(double h, double w);

/**
* @param h 高，保证为正数
* @param w 宽，保证为正数
* @param c 颜色
*/
Rectangle(double h, double w, String c);

/**
* @return 得到高
*/
double getHeight();

/**
* @return 得到宽
*/
double getWidth();

/**
* 设置新的高
* @param h 高
*/
void setHeight(double h);

/**
* 设置新的宽
* @param w 宽
*/
void setWidth(double w);

/**
```

```
* @return 得到该图形的面积
*/
double getArea();

/**
* @return 得到该图形的周长
*/
double getPerimeter();

/**
* 可选，基于实现思路选择是否要重写(override)基类Shape中的toString方法。
* @return 字符串表示，格式为"Rectangle(颜色)"。
*/
String toString();
```

4. `Square`：正方形

```
Square();

/**
* @param a 边长，保证为正数
*/
Square(double a);

/**
* @param a 边长，保证为正数
* @param c 颜色
*/
Square(double a, String c);

/**
* @return 得到变长
*/
double getSide();

/**
* 设置新的边长
* @param w 宽
*/
void setSide(double a);

/**
* 可选，基于实现思路选择是否要重写(override)基类Rectangle中对应方法。
* @return 得到该图形的面积
*/
double getArea();

/**
* 可选，基于实现思路选择是否要重写(override)基类Rectangle中对应方法。
```

```
* @return 得到该图形的周长
*/
double getPerimeter();

/**
* 可选，基于实现思路选择是否要重写(override)基类Shape中的toString方法。
* @return 字符串表示，格式为"Square(颜色)"。
*/
String toString();
```

5. `Shapes`：一系列图形的集合

```
/**
* @param s 一系列图形构成的数组，数组中的对象可能是Rectangle, Circle或Square
*/
Shapes(Shape[] s);

/**
* @return 一系列图形集合的总面积
*/
double getArea();

/**
* @return 所有已着色图形的总面积
*/
double getFilledArea();

/**
* @return 输出一系列图形的名称，格式为"[图形1(颜色1),图形2(颜色2),...,图形N(颜色N)]”，按照构造函数的顺序输出
*/
String toString();
```

提示： 实现`Shapes`类的一种方法是利用`Shape`类及其子类的多态特性，亦即虽然用`Shape`引用指向不同子类对象，仍然能够统一调用各子类中的`getArea`和`getColor`等函数，而多态特性确保具体执行哪个子类的方法取决于运行时对象的实际类型。