Java程序设计编程作业-1

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
        ├─complex
        │    ├─Complex.java (待实现)
        │    ├─*.java (如果有多个源文件)
        │    └─Test.java
        ├─rational
        │    ├─Rational.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`。

```
└─hw1
    ├─README.pdf
    │
    └─src
        ├─complex
        │    ├─Complex.java (待实现)
        │    ├─*.java (如果有多个源文件)
        │    └─Test.java
        ├─rational
        │    ├─Rational.java (待实现)
        │    └─Test.java
        ...
        │
        └─testutil
```

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全部测例**。
- 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: Complex （40”）

在包`complex`中实现公有的功能类`Complex`，进行复数的算术运算。该类所要求的接口见以下**伪代码**。

1. 用两个公有的成员变量表示复数的实部和虚部。

```
/**
* 分别表示复数的实部和虚部。
*/
double realPart, imaginaryPart;
```

2. 构造函数，参数分别是实部和虚部。

```
/**
* 构造函数。
* @param real 实部。
* @param imag 虚部。
*/
Complex(double real, double imag);
```

3. 复数的字符串形式表示，公有函数。根据书写传统，需要满足以下规则：

- 实部和虚部均保留三位小数，用i表示虚数。
- 复数为0时输出0.000。虚部非零但实部为零时只显示虚部，虚部为零但实部非零时只显示实部。
- 实部为负数时显示负号，为正数时不显示正号。
- 虚部为正数和负数都需要输出符号。除非实部为0，此时正虚部不显示正号，负虚部显示负号。

示例输出：0.000; 1.000; -1.000; 1.000+2.000i; -1.000-2.000i; -2.000i; 2.000i。

```
/**
* 复数的字符串输出X+Yi。
* @return 字符串形式的复数，保留三位小数，例如1.000+1.000i。
*/
String toString();
```

4. 复数的四则运算，公有函数。

```
/**
* 复数加法c=a+b，其中a是本对象，b是另一个Complex对象。
* @param b 另一个Complex对象。
* @return 计算结果c。
*/
Complex add(Complex b);

/**
* 复数减法c=a-b，其中a是本对象，b是另一个Complex对象。
* @param b 另一个Complex对象。
* @return 计算结果c。
*/
Complex sub(Complex b);

/**
* 复数乘法c=a*b，其中a是本对象，b是另一个Complex对象。
* @param b 另一个Complex对象。
* @return 计算结果c。
*/
Complex mul(Complex b);

/**
* 复数除法c=a/b，其中a是本对象，b是另一个Complex对象。
* @param b 另一个Complex对象。
* @return 计算结果c。
*/
Complex div(Complex b);
```

测例数据范围：

- 作为函数参数输入的复数$X+Yi$满足：$|X| \leq 1000$，$|Y| \leq 1000$。
- 0不作为除法的除数。

Problem 2: Rational （35“）

在包rational 中实现公有的功能类 Rational，用来执行分数的算术运算。该类所要求的接口见以下**伪代码**。

1. 用两个公有的成员变量表示分数的分子和分母。一个Rational对象的分子和分母必须总处于**约分**的形式。

```
/**
* 分别表示分子和分母。
*/
int numerator, denominator;
```

2. 构造函数，参数分别是分子和分母。构造对象时对输入的分子和分母**约分**。例如，若给定的分数为2/4(即分子为2，分母为4)，那么要把它约减为1/2，然后存储在相应的成员变量中，即分子为1，分母为2。

```
/**
* 构造函数。
* @param num 分子。
* @param den 分母。
*/
Rational(int num, int den);
```

3. 分数的类型转换，公有函数。转化为字符串x/y的形式，需要进行**约分**，整数的字符串形式省略分母，例如5/3; 1。分数也支持转化为实数类型，例如6/5转化为1.2。

```
/**
* 分数的字符串输出x/y，需要进行约分。
* @return 字符串形式的分数，例如5/3，1。
*/
String toString();

/**
* 分数转化为实数。
* @return 分数等价的实数。
*/
double toDouble();
```

4. 分数的四则运算，公有函数。要求运算结果进行**约分**。

```
/**
* 分数加法a <- a+b。其中a是本对象，b是另外一个Rational对象。计算结果存储在本对象a中。
* @param b 另外一个Rational对象。
*/
void add(Rational b);

/**
* 分数减法a <- a-b。其中a是本对象，b是另外一个Rational对象。计算结果存储在本对象a中。
* @param b 另外一个Rational对象。
*/
void sub(Rational b);

/**
* 分数乘法a <- a*b。其中a是本对象，b是另外一个Rational对象。计算结果存储在本对象a中。
* @param b 另外一个Rational对象。
```

```
*/
void mul(Rational b);

/**
* 分数除法a <- a/b。其中a是本对象，b是另外一个Rational对象。计算结果存储在本对象a中。
* @param b 另外一个Rational对象。
*/
void div(Rational b);
```

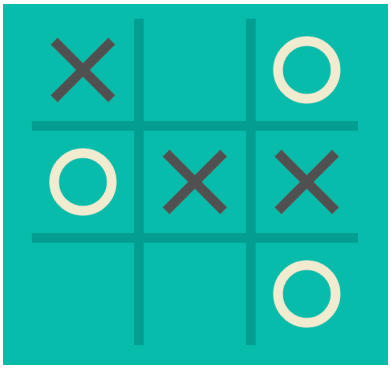
测例数据范围：

- 作为函数参数输入的分数X/Y满足：$1 \leq X \leq 1000$，$1 \leq Y \leq 1000$。
- 测例中出现的分数，包括中间结果，全为正数。

Problem 3: TicTacToe （25"）

在包 tictactoe 中实现公有功能类TicTacToe ， 模拟Tic-Tac-Toe游戏。

如下图所示，Tic-Tac-Toe游戏包含一个 3×3 的棋盘，共两个玩家(id分别为1/2)，交替在棋盘上标记。玩家1标记`circle`，玩家2标记`cross`。首先将三个相同标记连成一条直线（行/列/对角线）的玩家获得胜利。



该类所要求的接口见以下**伪代码**。

1. 定义一个构造函数，用来初始化一个空棋盘。

```
/**
* 构造函数。新建一个空棋盘。
*/
TicTacToe();
```

2. 定义 place 函数，公有函数，用来模拟某个玩家对某个位置进行了标记。该函数需要返回一个整数，表示该操作后棋盘的状态：

- 如果此时没有玩家胜出，返回0;
- 如果此时玩家1胜出，返回1;
- 如果此时玩家2胜出，返回2;
- 如果这一步操作在一个已经被标记过的位置，则忽略这次操作，返回3。

某位玩家胜出，当且仅当某行、某列或某条对角线上的三个位置都是该玩家的标记。

```
/**
* 进行一步游戏，玩家在棋盘上进行操作，函数返回操作的结果。
* 函数判断玩家操作是否非法（在已经标记的位置上重复标记）。若玩家操作非法，则函数不执行操作。
* 否则函数执行玩家的操作，然后判断是否有玩家获胜，游戏是否继续。
* @param player 玩家id，1或2。
* @param row 操作的行数，范围{0,1,2}。
* @param column 操作的列数，范围{0,1,2}。
* @return 一步操作的结果。返回0:游戏继续； 1:玩家1获胜； 2:玩家2获胜； 3:玩家操作非法。
*/
int place(int player, int row, int column);
```

测例数据范围：

- 保证`place`函数的参数在上述规范的合理范围内。