

2019/2020(1) Java 程序设计上机实验 2

班级：软工 1805

学号：201806061219

姓名：王程飞

完成日期：2019.10.17

1、编写一个三角形类 Triangle，能根据 3 个实数构造三角形对象，如果三个实数不满足三角形的条件，则自动构造以最小值为边的等边三角形。随机产生 3 个数，求构造的三角形面积。

分析

注意三角形边长是否符合条件的算法，随机数生成用
`Random`.nextDouble()*100;

代码

```
public class Complex {
    private int real, imagin;
    Complex() {
        real = 0;
        imagin = 0;
    }
    Complex(int real, int imagin){
        this.real = real;
        this.imagin = imagin;
    }
    Complex complexAdd(Complex complex){
        return new Complex(this.real+complex.real,
this.imagin+complex.imagin);
    }

    @Override
    public String toString() {
        if(imagin >= 0) {
            return real+" "+imagin+"i";
        } else {
            return real+" "+imagin+"i";
        }
    }
}

private static void test_01() {
    System.out.println("-----Triangle-----");
    Random random = new Random();
    double a = random.nextDouble()*100;
    double b = random.nextDouble()*100;
```

```

        double c = random.nextDouble()*100;
        Triangle triangle = new Triangle(a, b ,c);
        System.out.println("三角形边长: "+triangle.a+",
        "+triangle.b+", "+triangle.c+"; 面积: "+triangle.area());
    }

```

结果

-----Triangle-----

三角形边长: 37.63768786280631, 64.3838718271164, 79.17536035577129;
面积: 1198.6739438611119

2、编写一个复数类 Complex 验证两个复数 $1+2i$ 和 $3+4i$ 相加产生一个新的复数 $4+6i$ 。

复数类 Complex 必须满足如下要求:

(1) 复数类 Complex 的属性有:

realPart 是 int 型, 代表复数的实数部分

imaginPart 是 int 型, 代表复数的虚数部分。

(2) 复数类 Complex 的方法有:

Complex() 构造方法, 将复数的实部和虚数都置 0

Complex(int r, int i) 构造方法, 形参 r 为实部的初值, i 为虚部的初值。

Complex complexADD(Complex a) 将当前复数对象与形参复数对象相加, 所得的结果仍是一个复数值, 将其返回给此方法的调用者。

String toString() 把当前复数对象的实部、虚部组合成 $a+bi$ 的字符串形式, 其中, a 和 b 分别为实部和虚部的数据。

分析

定义一个类 Triangle 并声明两个成员变量, 在之后的计算时要注意符号输出
代码

```

public class Complex {
    private int real, imagin;
    Complex() {
        real = 0;
        imagin = 0;
    }
    Complex(int real, int imagin){
        this.real = real;
        this.imagin = imagin;
    }
    Complex complexAdd(Complex complex){
        return new Complex(this.real+complex.real,
        this.imagin+complex.imagin);
    }
}

```

```

    }

    @Override
    public String toString() {
        if(imagin >= 0) {
            return real+" "+imagin+"i";
        } else {
            return real+" "+imagin+"i";
        }
    }
}

private static void test_02() {
    System.out.println("-----Complex-----");
    Complex complex = new Complex(1, 2);
    System.out.println(complex);
    Complex _complex = new Complex(3, 4);
    System.out.println(_complex);
    System.out.println(complex.complexAdd(_complex));
}

```

结果

```

-----Complex-----
1+2i
3+4i
4+6i

```

3、编写一个银行账户类 **Account**，并编写一个 **main** 方法，对 **Account** 类进行测试，在 **main** 方法中要调用 **Account** 的所有方法。存取款的步骤：先存入帐号若干元（随机产生），然后取款若干元（随机产生），最后查询帐号信息。

该类的成员变量如下（访问权限均为 **private**）：

变量名	含义	数据类型
id	帐号	String
owner	账户持有人姓名	String
balance	余额	double

该类的成员方法如下（访问权限均为 **public**）：

方法名	返回类型	参数	说明
构造方法	void	无	构造一个账户实例，将 id , owner 设为 null , balance 设为 0.00
构造方法	void	String id , String owner , double amount	构造一个账户实例，用参数设置成员变量 id , owner , balance 的值
setID	void	String id	用参数设置成员变量 id 的值。
setOwner	void	String owner	用参数设置成员变量 owner 的值。

deposit	void	String id, double amount	将金额 amount 存入 id 帐户, 如果 id 帐号为 null, 则打印信息为“帐号未知!” 否则打印“成功存款×××元, 当前余额为×××元!”。
withdraw	void	String id, double amount	从 id 帐户支取金额 amount, 如果帐号为 null 或者余额小于 amount, 则打印信息分别为“帐号未知!” 和“余额不足!”, 否则打印“成功取款×××元, 当前余额为×××元!”
queryBalance	void	无	打印 id, owner, balance。
getBalance	double	String id	返回当前帐号 id 的余额, 要求保留 2 位小数

代码

```
import java.text.DecimalFormat;

class Account {

    public void setId(String id) {
        this.id = id;
    }

    public void setOwner(String owner) {
        this.owner = owner;
    }

    public void deposit(String id, double amount) {
        if(!this.id.equals(id)) {
            System.out.println("账号未知");
        } else {
            if (amount<0) {
                System.out.println("存款不能小于 0");
                return;
            }
            balance+=amount;
            System.out.println("成功存款 "+amount+" 元, 当前余额 "+balance+" 元");
        }
    }

    public void withdraw(String id, double amount) {
        if(!this.id.equals(id)) {
            System.out.println("账号未知");
        } else if(amount > balance) {
```

```

        System.out.println("余额不足");
    } else {
        if (amount < 0) {
            System.out.println("取款不能小于 0");
            return;
        }
        balance -= amount;
        System.out.println("成功取款 "+amount+" 元, 当前余额 "+balance+" 元");
    }
}

```

```

public void queryBalance(){
    System.out.println("id: "+id+", owner: "+owner+", balance: "+balance);
}

```

```

public String getBalance(String id) {
    return new DecimalFormat("#.00").format(balance);
}

```

```

private String id, owner;
private double balance;

```

```

Account(){

```

```

    id = null;
    owner = null;
    balance = 0.00;
}

```

```

Account(String id, String owner, double amount) {

```

```

    this.id = id;
    this.owner = owner;
    this.balance = amount;
}

```

```

}

```

```

private static void test_03() {

```

```

    System.out.println("-----Account-----");
    Account account = new Account("10000", "abc", 100.00);
    account.queryBalance();
    Random random = new Random();
    account.deposit("10000", random.nextDouble()*100);
    account.withdraw("10000", random.nextDouble()*100);
    account.setId("10001");
}

```

```
account.setOwner("def");  
System.out.println(account.getBalance("10001"));  
account.queryBalance();  
}
```

结果

-----Account-----

id: 10000, owner: abc, balance: 100.0

成功存款 47.483933651460674 元, 当前余额 147.4839336514607 元

成功取款 48.71075510085468 元, 当前余额 98.773178550606 元

98.77

id: 10001, owner: def, balance: 98.773178550606