Java-06

一、Task1.继承

1. **同名变量访问顺序:**

   子类有的优先访问子类成员，然后才访问父类成员

```
class Animal {  1 个用法  1 个继承者

        private String name;  4 个用法
        private int id;  2 个用法

        int a = 1;  1 个用法
        int b = 2;  1 个用法
```

```
class Penguin extends Animal {
    /*int a = 10;
    int b = 20;*/
```

   输出结果

```
1
2
```

   子类中定义后:

```
10
20
```

   可知子类优先级更高

2. **super关键词**
   - 访问父类 **成员变量 方法 构造方法**
   - 方法调用: super.方法名()
   - 构造方法调用: **super()** 必须放在子类构造方法的第一行

     因此不能与 **this** 同时存在

     (**this()** 可以调用当前类的其他构造方法)

3.**不支持多继承**

- 调用方法不明确(复杂)

  假设继承A和B，都有方法 **a** ，则调用方法无法确定

- **解决方案:** 使用多接口实现替代

二、Task2.多态

代码如下:

```
public interface Shape {
    double getArea();
    double getCircumference();
    String getName();

}
//圆形类
class Circle implements Shape {
    private double radius;
    private String name;

    public Circle(double radius, String name) {
        this.radius = radius;
        this.name = name;
    }

    @Override
    public double getArea() {
        return Math.PI * radius * radius;
    }

    @Override
    public double getCircumference() {
        return 2 * Math.PI * radius;
    }

    @Override
    public String getName() {
        return name;
    }
}
//三角形类
class Triangle implements Shape {
    private double side1;
    private double side2;
    private double side3;
    private String name;

    public Triangle(double side1, double side2, double side3, String name) {
        this.side1 = side1;
        this.side2 = side2;
        this.side3 = side3;
        this.name = name;
        //判断三角形存在与否
        boolean istrue = (side1 + side2 > side3 && side1 + side3 > side2 &&
side2 + side3 > side1);
        if (istrue == false) {
            System.out.println("您创建的三角形" + name + "不存在");
        }
    }

    @Override
    public double getArea() {
```

```
        double c =  getCircumference()/2;
        return Math.sqrt(c * (c - side1) * (c - side2) * (c - side3));
    }

    @Override
    public double getCircumference() {
        return side1 + side2 + side3;
    }

    @Override
    public String getName() {
        return name;
    }
}
//矩形类
class Rectangle implements Shape {
    private double Side1;
    private double Side2;
    private String name;

    public Rectangle(double Side1,double Side2,String name) {
        this.Side1 =Side1;
        this.Side2 =Side2;
        this.name = name;
    }

    @Override
    public double getArea() {
        return Side1 * Side2;
    }

    @Override
    public double getCircumference() {
        return 2 * (Side1 + Side2);
    }

    @Override
    public String getName() {
        return name;
    }
}
//主类(运行)
class Test06{
    public static void main(String[] args) {
    Circle c1 = new Circle(2,"c1");
    Triangle t1 = new Triangle(3,4,5,"t1");
    Rectangle r1 = new Rectangle(2,4,"r1");
        System.out.println(c1.getArea());
        System.out.println(c1.getCircumference());
        System.out.println(t1.getArea());
        System.out.println(t1.getCircumference());
        System.out.println(r1.getArea());
        System.out.println(r1.getCircumference());

    }
}
```

三、Task3.封装

- **默认** 只能被**相同包**下的类访问
- **private** 只能在**当前类内部**被访问**(且不能修饰类和接口)**
- **public** 可以被所有类访问
- **protected** 可以被**相同包下的类**或**不同包中子类**访问
  (子类即继承关系的类)

代码如下:

```
public class BankAccount {
    // TODO 修改属性的可见性
    protected String accountNumber;
    protected String accountHolder;
    protected double balance;
    private String password; // 敏感信息，需要严格保护

   BankAccount(String accountNumber, String accountHolder, double
initialBalance, String password) {
        //TODO
       this.accountNumber = accountNumber;
       this.accountHolder = accountHolder;
       this.balance = initialBalance;
       this.password = password;
    }

    void deposit(double amount) {
        //TODO
        this.balance += amount;
        System.out.println("已成功存入" + amount + "元");
    }

    boolean withdraw(double amount, String inputPassword) {
        //TODO
        if (inputPassword.equals(this.password)) {
            if (this.balance >= amount) {
                this.balance -= amount;
                System.out.println("已成功取出" + amount + "元");
                return true;
            } else {
                System.out.println("密码错误");
                return false;
            }
        }
        else {
            System.out.println("余额不足");
            return  false;
        }
    }

    boolean transfer(BankAccount recipient, double amount, String inputPassword)
{
        //TODO
        //判断账户是否存在
```

```
        if (recipient.equals(null)) {
            System.out.println("您输入的账户不存在");
            return false;
        }
        else{
            //判断密码是否正确
            if (inputPassword.equals(this.password)) {
                if (this.balance >= amount) {
                    this.balance -= amount;
                    recipient.balance += amount;
                    System.out.println("已向账户" + recipient.accountNumber + "转
入" + amount + "元");
                    return true;
                }
                else{
                    System.out.println("余额不足");
                    return false;
                }
            }
            else {
                System.out.println("密码错误");
                return false;
            }
        }
    }

    double getBalance() {
        //TODO
        return this.balance;
    }

    String getAccountInfo() {
        //TODO
        return this.accountNumber;
    }
    // 只需修改可见性
   private boolean validatePassword(String inputPassword) {
        return true;
    }
    // 只需修改可见性
    boolean validateAmount(double amount) {
        return true;
    }
}
```