Task1

1.避免菱形问题:

当一个类继承多个具有相同方法的父类时，会产生调用歧义,
编译器无法确定应该调用哪个父类的方法,
这会导致代码逻辑混乱和不可预测的行为.

Task2

shape类接口:

```
package com.duotai;

public interface Shape {
    public double calculateC();

    public double calculateS();
    public String getShapeType();
}
```

circ类:

```
package com.duotai;

public class circle implements Shape{
    private double r;
    public circle(double r){
        this.r=r;
    }
    @Override
    public double calculateC() {
        return 2* Math.PI*r;

    }

    @Override
    public double calculateS() {

        return Math.PI*r*r;
    }

    @Override
```

```
    public String getShapeType() {
        return "圆形";
    }
}
```

triangle类:

```
package com.duotai;

public class triangle implements Shape{
    private double a;
    private double b;
    private double c;
    public triangle(double a, double b , double c){
        this.a=a;
        this.b=b;
        this.c=c;
    }
    @Override
    public double calculateC() {
        return a+b+c;
    }

    @Override
    public double calculateS() {
        double s = (a + b + c) / 2;
        return Math.sqrt(s * (s - a) * (s - b) * (s - c));
    }

    @Override
    public String getShapeType() {
        return "三角形";
    }
}
```

rectangle类:

```
package com.duotai;

public class rectangle implements Shape{
    private double a;
    private  double b;
    public rectangle(double a,double b){
        this.a=a;
        this.b=b;
    }
```

```
    @Override
    public double calculateC() {
        return 2*a+2*b;
    }

    @Override
    public double calculateS() {
        return a*b;
    }

    @Override
    public String getShapeType() {
        return "矩形";
    }
}
```

最后运行：
model类：

```
package com.duotai;

public class model {
    public static void main(String[] args) {

        Shape[] shapes = new Shape[]{
                new circle(5),
                new triangle(2, 1, 2),
                new rectangle(3, 9)
        };


        for (Shape shape : shapes) {
            System.out.println(shape.getShapeType());
            System.out.println("周长: " + shape.calculateC());
            System.out.println("面积: " + shape.calculateS());
            System.out.println("--------------------");
        }
    }
}
```

Task3
public：用于可以任意访问的数据
private：用于只能被该类直接访问的隐私数据
````
public class BankAccount {
// TODO 修改属性的可见性
````

private String accountNumber ;
public String accountHolder;
private double balance;
private String password; // 敏感信息，需要严格保护

```
1.         BankAccount(String accountNumber, String accountHolder, double
   initialBalance, String password) {
2.             //TODO
3.             this.accountNumber = accountNumber;
4.             this.accountHolder = accountHolder;
5.             this.balance = initialBalance;
6.             this.password = password;
7.         }
8.
9.         void deposit(double amount) {
10.            //TODO
11.            if (validateAmount(amount)) {
12.                balance += amount;
13.            }
14.
15.        }
16.
17.        boolean withdraw(double amount, String inputPassword) {
18.            //TODO
19.            if (!validatePassword(inputPassword) || !validateAmount(amount)) {
20.                return false;
21.            }
22.            if (amount > balance) {
23.                return false; // 余额不足
24.            }
25.            balance -= amount;
26.            return true;
27.
28.
29.        }
30.
31.        boolean transfer(BankAccount recipient, double amount, String
   inputPassword) {
32.            //TODO
33.            if (!validatePassword(inputPassword) || !validateAmount(amount)) {
34.                return false;
35.            }
36.            if (amount > balance) {
37.                return false; // 余额不足
38.            }
39.            if (recipient == null) {
```

```
                return false; // 收款账户不存在
            }
            balance -= amount;
            recipient.deposit(amount);
            return true;
        }

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

        String getAccountInfo() {
            //TODO
            return "Account Number: " + accountNumber + ", Account Holder: " +
accountHolder + ", Balance: " + balance;

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