2021.5.19.

P215 - P227 Java_递归

递归就是方法自己调用自己，每次调用都传入不同的变量，有助于编程者解决一些复杂的问题。
但递归要避免死龟（即无限递归问题），会导致栈溢出。△与方法执行机制有关

- 斐波那契数列 fibonacci
- 猴子吃桃
- 老鼠出迷宫
- 汉诺塔
- 八皇后

递归重要原则

1）执行一个方法时，就创建一个新的受保护的独立空间（栈空间）
2）方法的局部变量是独立的，不会相互影响，比如n变量
3）如果方法中使用的是引用类型变量（比如数组），就会共享该引用类型的数据
4）递归必须向退出递归的条件逼近，否则就是无限递归，出现 StackOverflowError，死龟了
5）当一个方法执行完毕，或者遇到 return 语句，就会返回，遵守"谁调用，就将结果返回给谁"的规则，同时当方法执行完毕或者返回时，该方法也就执行完毕。

P228 - P233 Java_方法重载

方法重载

Java中允许同一个类中，多个同名方法的存在，但要求形参列表不一致

重载的好处

减轻了起名和记名的麻烦

重载的注意事项和使用细节

1）方法名必须相同
2）形参列表必须不同（形参类型或个数或顺序，至少有一样不同，参数名无要求）
3）返回类型：无要求

```
22      //下面的四个 calculate方法构成了重载
23      //两个整数的和
24      public int calculate(int n1, int n2)  {
25          System.out.println("calculate(int n1, int n2) 被调用");
26          return n1 + n2;
27      }
28
29      public void calculate(int n1, int n2)  {
30          System.out.println("calculate(int n1, int n2) 被调用");
31          int res =  n1 + n2;
32      }
```

⇒返回类型不同不能构成重载

● 课堂练习题

1. 判断题：
与void show(int a,char b,double c){}构成重载的有：[_____]
a) void show(int x,char y,double z){} ×
b) int show(int a,double c,char b){} ✓
c) void show(int a,double c,char b){} ✓

d) boolean show(int c,char b){} ✓
e) void show(double c){} ✓
f) double show(int x,char y,double z){} ×
g) void shows(){} ×

P334 - P336 Java_可变参数

基本概念

Java允许将同一个类中多个同名同功能但参数个数不同的方法，封装成一个方法

基本语法

访问修饰符 返回类型 方法名（数据类型... 形参名）{ }

注意事项和使用细节

1）可变参数的实参可以为0个或任意多个

2）可变参数的实参可以为数组

3）可变参数的本质就是数组.

4）可变参数可以和普通类型的参数一起放在形参列表，但必须保证可变参数在最后

5）一个形参列表中只能出现一个可变参数

P337 - P339 Java_作用域

基本使用

1）在Java编程中，主要的变量就是属性（成员变量）和局部变量.

2）我们说的局部变量一般指在成员方法中定义的变量

3）Java中作用域的分类

全局变量：也就是属性，作用域为整个类体

局部变量：也就是除了属性之外的其他变量，作用域为定义它的代码块中.

4）全局变量可以不赋值，直接使用，因为有默认值；局部变量必须赋值后，才能使用，（无默认值）.

注意事项和使用细节

1）属性和局部变量可以重名，访问时遵循就近原则.

2）在同一个作用域中，例如同一个成员方法中，两个局部变量，不能重名.

3）属性生命周期较长，伴随着对象的创建而创建，伴随着对象的销毁而销毁。局部变量，生命周期较短，伴随着它的代码块的执行而创建，伴随着代码块的结束而销毁。即在一次方法调用过程中.

4>作用域范围不同

全局变量/属性：可以在本类使用，或其他类使用(通过对象调用)

局部变量：只能在本类中 对应的方法中使用

5>修饰符不同

全局变量/属性可以加修饰符

局部变量不可以加修饰符

```
class Cat {
    //全局变量：也就是属性，作用域为整个类体 Cat类：cry eat 等方法使用属性
    //属性在定义时，可以直接赋值
    int age = 10; //指定的值是 10

    //全局变量(属性)可以不赋值，直接使用，因为有默认值，
    double weight;  //默认值是0.0

    public void hi() {
        //局部变量必须赋值后，才能使用，因为没有默认值
        int num;
        System.out.println("num=" + num);
    }
```

报错

```
        /*
        属性生命周期较长，伴随着对象的创建而创建，伴随着对象的销毁而销毁。
        局部变量，生命周期较短，伴随着它的代码块的执行而创建，
        伴随着代码块的结束而销毁。即在一次方法调用过程中
         */
        p1.say();//当执行say方法时，say方法的局部变量比如name,会创建，当say执行完毕后
        //name局部变量就销毁,但是属性(全局变量)仍然可以使用

    }
}

class Person {

    String name = "jack";

    public void say() {
        //细节 属性和局部变量可以重名，访问时遵循就近原则
        String name = "king";
        System.out.println("say() name=" + name);
    }
```

输出 King.

P140 - P Java_构造器 constructor

基本语法：

[修饰符] 方法名(形参列表){

方法体;

}

说明

1>构造器的修饰符可以默认，也可以是 public protected private

2>构造器无返回值

3>方法名和类名字必须一样.

4>参数列表和成员方法规则一样

5>构造器的调用，由系统完成，自动调用该类构造器完成对对象的初始化.

注意事项和使用细节

1) 一个类可以定义多个不同的构造器，即构造重载
2) 构造器是完成对象的初始化，并不是创建对象.
3) 如果程序员没有定义构造器，系统会自动给类生成一个默认无参构造器，也叫默认构造器)，如 Dog(){}，可以使用 Javap 反编译 .class 文件查看.
4) 一旦定义了自己的构造器，默认的构造器就覆盖了，就不能再使用默认的无参构造器，除非显式定义一 Dog(){}.