Day 07

2018年5月8日 8:55

课程回顾

计算机软件:软件最终操作的数据,操作的业务就方法(功能)

数组:

一维数组,二维数组

数组就一种数据结构

在java中可以按照数组的方式组织数据

数组可以存储一组数据,可以通过数组的名称来使用数组元素(数据)

数组的特点:相同数据类型,内存中连续内存空间,结合数据的内存存储图

数组的定义

数据类型[] 数组名称

数据类型[][] 数组的名称

重要的三件事

1.有声明

2.有声明有空间有默认值

3.有声明有空间还有指定初始数

异常

ArrayIndexOutofBoundException

NullPointerException

声明了,没给初始数据,使用的时候,报没有初始化

方法:提高代码的复用率

方法的重要的三件事

方法需要参数

方法完成什么功能

方法的返回值

这三个是指导原则来定义方法

访问修饰符 返回类型 方法名称(参数列表){

//方法体(根据实际的业务)

return 返回数据;

```
}
```

return是终止方法的执行

注意的问题:

无法触及的代码

可变参数

Api application program interface(就是方法)

方法的重载;

在同一个类内,有相同的方法名称,但方法的参数的个数或类型不同

方法的递归:

非递归调用a方法调用b方法

```
// 非递归调用
public int sum1(int n){
    int s=0;
    for(int i=1;i<=n;i++){
        s+=i;
    }
    return s;
}
```

递归方法调用是自己调用自己

```
//递归调用
public int sum2(int n){
    if(n==1){
        return 1;
    }
    return n+sum2(n-1);
}
```

以n=3为例子

```
sum2(3)
{
    //不会执行return1
    return 3+sum(3-1);//3+3
}

        sum2(2)  返回结果3
        {
            //不会执行return1
            return 2+sum(2-1);//2+1
        }

                sum2(1)  返回值是1
                {
                    会执行return1
                }
```

递归调用和非递归调用,都能完成需求,但递归调用要慎用,频繁方法的压栈和弹栈会消耗内存和cpu,性能会降低,
但是有些需求,二叉树的遍历(前序遍历,中序遍历,后序遍历)

面向对象:
把现实世界的物与事转到计算机中
把现实世界的物与事的数据转到计算机中内存中
类就是一种数据结构,用这种数据结构组织不同数据类型的数据
现实世界的物和事的数据 --> 类 --->用类在内存new对象

做面想对象最难的事,根据现实世界物或事转换(抽取)成类,
难点在类中要组织什么属性和什么方法(结合具体的业务需求)

类可以组织不同类型的数据,应该有一些方法来操作这些数据,考虑这些数据都被谁访问,考虑数据的安全性
用方法操作数据
可以用代码逻辑来保证数据准确性

```
        public void setAge(int age) {
            if(age>0 && age<120){
                this.age = age;
            }else{
                age=18;
            }

        }
        public void setStuNo(String stuNo) {
                char c=stuNo.charAt(0);
                if(c=='S' || c=='s'){
                    this.stuNo = stuNo;
                }else{
                    stuNo=null;
                }

            }
```

用方法的代码逻辑使用和处理数据

```
        public void study(){
                System.out.println(name+"在认真的学习");
            }
            public void liaomei(){
                System.out.println("小姐姐,我叫"+name+"  我年龄"+age+"  可以加个
                微信");
            }
            public void dajia(){
                System.out.println(name+"冲动打架了");
            }
```

创建一个类就是在做封装

类里封装了属性和方法

类的定义:

访问修饰符 class 类的名字{

//若干属性(建议私有的)

//若干方法(操作的属性的数据,建议公有)

}

用类的定义在内存实例化对象(白话就在内存申请内存空间)

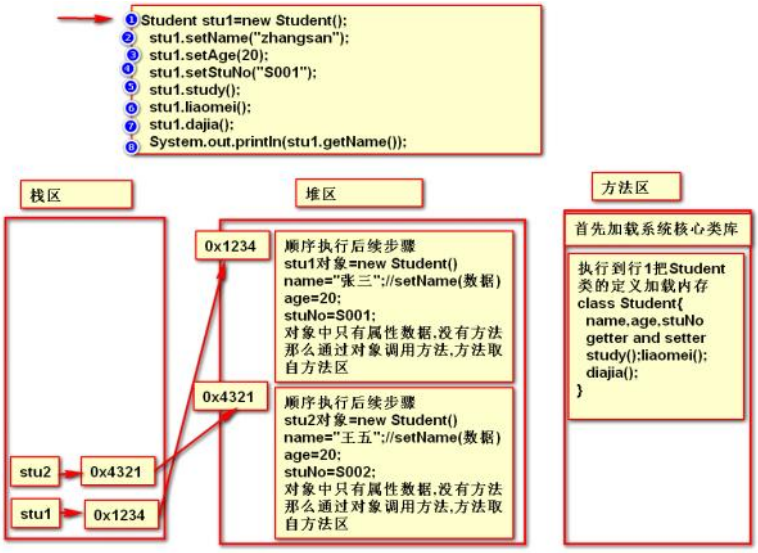
类的名字 对象的名字=new 类的名字();//构造函数/方法

使用对象中的属性或方法

尽量用对象的名称.类中的方法的名称();

类的定义内容来源现实世界的对象

用类在内存中new对象,在内存中申请了内存空间,对象存储数据



明确一些相关的名词概念:

成员变量:类里的属性,作用域整个类,可以放在类的任何位置,会有默认值

默认值取决于类型是谁,对象消失了说明成员变量也消失,

成员变量存储堆中

局部变量:方法里的变量,从声明位置开始,对应右大括号,不给初值就使用会错

局部变量超范围就消失,局部变量存储到栈中

匿名对象:没有对象名,但是会在堆中开辟空间,对象只能使用一次

使用场景,要调用的方法就调用一次

有名对象:(对象就一个)

```
Student  stu=new Student();//对象名是stu
stu.study();
stu.liaomei();
```

匿名对象:(对象多个)

```
new Student().study();
new Student().liaomei();
```

构造方法:

在new对象的时候给私有成员变量赋初值

用setter存数据,用getter取数据

```
Student stu1=new Student();
stu1.setName("zhangsan");
stu1.setAge(20);
stu1.setStuNo("S001");
String name=stu1.getName();
int age=stu1.getAge();
String  stuNo=stu1.getStuNo();
```

用构造方法给属性存数据

```
public  class Student{
    private  String name;
    private  int age;
    private String  stuNo;
    //构造方法
    public  Student(String name,int age, String  stuNo){
            this.name=name;
            this.age=age;
            this.stuNo=stuNo;
    }
    getter     setter
}
//用构造函数存储数据,用getter方式取数据
Student  stu2=new Student("wangwu",20,"S002");
 String name=stu1.getName();
 int age=stu1.getAge();
 String  stuNo=stu1.getStuNo();
```

说明:

构造方法名必须跟类名一样

不能有返回值

如果创建一个类,类中没有写任何构造,系统会给一个默认构造

空方法体的无参构造

public Student(){}

如果创建类的时候人为写了构造,那么默认构造系统就不会提供了

构造方法可以写多个,但必须参数的个数跟类型不同,重载构造

this(),是用来在构造函数中调用其他的构造函数,保证对象的数据一定有初值

建议:只要人为的写了构造,最好添加空的无参构造

this. 调用当前对象的属性或方法

this指代的"永远"是当前对象

如何判断当前对象:

this在哪个方法里,有this的这个方法被哪对象调用,this指代就那个对象

this不能调用静态的内容(属性和方法)

局部代码块:局限于方法中的用{}括起来的代码

能把变量的定义放在局部代码块就放里,尽量把变量的作用域缩短

在方法中有

```
for(int i=0;i<10;i++){

    System.out.println(i)
}
```

构造代码块/代码块:

在类中用{},括起来的若干代码,作用用来做初始化用

补充内容:

The field Student.name is not visible

属性/字段 name 是 不可见的

定义类的常规做法:

方案一

public class 类名{

私有的属性

公有getter和公有的setter

}

这种类的对象,适合存储数据,数据存储到私有的属性中(成员变量)

用公有的getter方法获取数据

用公有的setter方法存储数据

方案二:

public class 类名{

没有任何属性

只有方法

}

这种类不适合存储数据,但是表达业务逻辑(功能),根据需求定义功能

定义方法体的内容 ,功能方法需要数据,数据来源于方法参数.