成员变量与局部变量

2018年6月16日 11:26

成员变量：在类方法外 堆内存
局部变量：在方法定义中或方法声明上 栈内存

生命周期不同：成员变量随着对象的创建而存在，随着对象的消失而消失
局部变量随着方法的调用而存在，随着方法的调用完毕而小时
初始值不同：成员变量：有初始值
局部变量：没有初始值，必须初始值才能调用

注意事项：局部变量名称可以与成员变量名称相同，在方法使用的时候，采用的是就近原则

```
class Variable{
	//成员变量
	int num = 10;
	public static void show(){
		int num2 = 20;//局部变量
	}
}
class VariableDemo{
	public static void main(String[] args){
		Variable v = new Variable();
		System.out.println(v.num);
	}
}
```

形式参数问题

2018年6月17日 13:24

```
//形式参数是基本类型
class Demo{
	public int sum(int a, int b){
		return a+b;
	}
}
//形参是基本类型
class Student{
	public void show(){
		System.out.println("我爱学习");
	}
}

class StudentDemo{
	//如果一个方法的形式参数是一个类类型（引用类型），这里其实需要的是该类的对象
	public void method(Student s){
		s.show();
	}
}

class ArgsDemo{
	public static void main(String[] args){
		Demo d = new Demo();
		int result = d.sum(10,20);
		System.out.println("result="+result);
		//需求：调用StudentDemo中的method()方法
		StudentDemo sd = new StudentDemo();
		Student s = new Student();
		sd.method(s);
	}
}
```

匿名对象

2018年6月17日 20:39

匿名对象：就是没有名字的对象

应用场景：
1.调用方法，仅仅只调用一次的对象（调用多次的时候不适合）
好处：匿名对象调用结束就是垃圾，可以被垃圾回收期回收，提高内存使用效率
2匿名对象可以作为实际参数传递

```
class Student{
	public void show(){
		System.out.println("study");
	}
}

class StudentDemo{
	public void method(Student s){
		s.show();
	}
}

class NoNameDemo{
	public static void main(String[] args){
		//带名字对象
		Student s = new Student();
		s.show();
		//匿名对象
		new Student().show();
		new Student().show();//重新调用一次新的对象

		//匿名对象作为实际参数
		StudentDemo sd = new StudentDemo();
		sd.method(new Student());
		//改进
		new StudentDemo().method(new Student());
	}
}
```

封装概念

2018年6月17日 20:57

通过对象去给成员变量赋值，可以赋值一些非法数据，这是不合理的
在赋值之前先对数据进行判断
判断应该在Student类中

private：可以修饰成员变量与成员方法
被private修饰的成员只能在本类中使用，隐藏对象的属性

```
class Student{
	String name;
	private int age;
	//写一个方法对数据进行校验
	public void setAge(int a){
		if(a<0||a>120){
			System.out.println("年龄有问题");
		}else{
			age = a;
		}
	}
	public void show(){
		System.out.println("姓名："+name);
		System.out.println("性别："+age);
	}
}

class StudentDemo{
	public static void main(String[] args){
		Student s = new Student();
		//s.age=27;
		s.setAge(-27);
		s.show();
	}
}
```

private

2018年6月17日 21:54

private是权限修饰符
可以修饰成员变量和成员方法 （一般用于修饰变量）

```
class Demo{
	private int num = 10;
	public void show(){
		System.out.println(num);
	}
	private void method(){
		System.out.println("method");
	}
	public  void function(){
		method();
	}
}

class PrivateDemo{
	public static void main(String[] args){
		Demo d = new Demo();
		d.show();//访问私有成员变量方法
		d.function();//访问私有方法
	}
}
```

private应用

2018年6月17日 22:20

1.把成员变量用private修饰

2.提供对应的getXxx()和setXxx()方法

```
public int setAge(int a){
        age = a;
}
```

this概述与应用

2018年6月17日 22:31

this：当前类的对象引用
它就代表当前类的一个对象
方法被哪个对象调用，this就代表那个对象

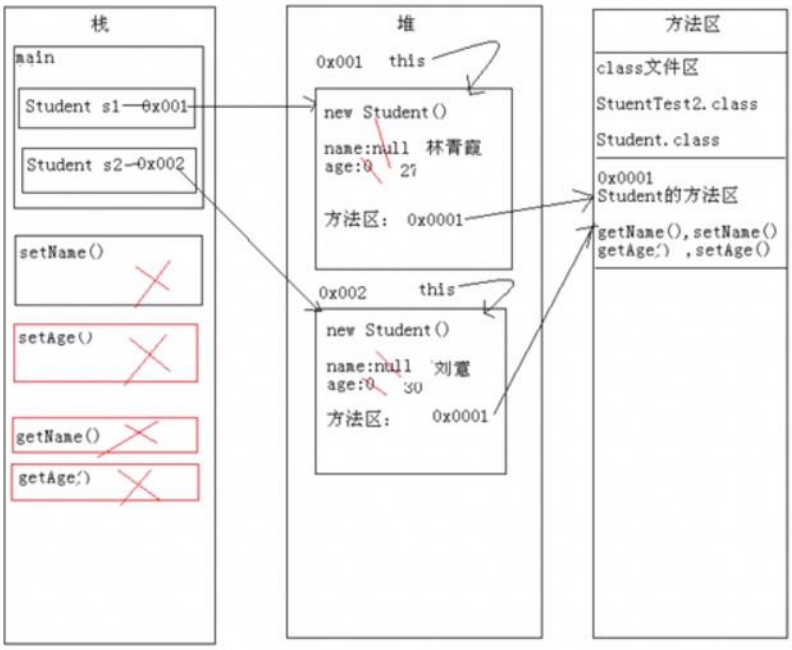
应用场景：解决局部变量隐藏成员变量

```
class Student{
	private String name;
	private int age;
	public String getName(){
		return name;
	}
	public int getAge(){
		return age;
	}
	public void setName(String name){
		//name = name; //错误：就近原则
		this.name = name;
	}
	public void setAge(int age){
		this.age = a;
	}
}

class StudentTest{
	public static void main(String[] args){
		Student s = new Student();
		s.setName("王若潇");
		s.setAge(22);
		System.out.println("name="+s.getName()+"age="+s.getAge());
	}
}
```

this内存图

2018年6月18日 13:57

```
class Student {
        private String name;
        private int age;

        public String getName() {
                return name; //这里其实是隐含了this
        }

        public void setName(String name) {
                this.name = name;
        }

        public int getAge() {
                return age;
        }

        public void setAge(int age) {
                this.age = age;
        }
}

class StudentTest2 {
        public static void main(String[] args) {
                //创建一个对象
                Student s1 = new Student();
                s1.setName("林青霞");
                s1.setAge(27);
                System.out.println(s1.getName()+"---"+s1.getAge());
                                                林青霞 "---" 27
                //创建第二个对象
                Student s2 = new Student();
                s2.setName("刘意");
                s2.setAge(30);
                System.out.println(s2.getName()+"---"+s2.getAge());
        }
}
```



构造方法

2018年6月18日 14:42

```
/*
构造方法:给对象的数据进行初始化
格式：A：方法名与类名相同
      B：没有返回值类型，连void都没有
        C：没有具体返回值
系统会自动提供一个无参方法；
如果自己给出了构造方法，系统将不再提供默认的构造方法
给成员变量赋值有两种方法：setXxx; 构造方法
*/
class Student{
        private String name;
        private int age;
        public Student(){
                System.out.println("这是无参构造方法");
        }
        //构造方法的重载格式
        public Student(String name){
                System.out.println("这是带string类型的构造方法");
                this.name = name;
        }
        public Student(int age){
                System.out.println("这是带int类型的构造方法");
                this.age = age;
        }
        public Student(String name,int age){
                System.out.println("这是带多个类型的构造方法");
                this.age = age;
                this.name = name;
        }
        public void show(){
                System.out.println(name+"---"+age);
        }
}

class ConstructDemo{
        public static void main(String[] Args){
                Student s1 = new Student();
                System.out.println(s1);//Student@15db9742
                Student s2 = new Student("王若潇");
                s2.show();
                Student s3 = new Student(22);
```

类的组成：成员变量+成员方法+**构造方法**

无参---有参

无返回值---有返回值

```
        s3.show();
        Student s4 = new Student("王若潇",22);
        s4.show();
    }
}
```

类的标准代码写法

2018年6月18日 14:58

```
/*
标准代码
学生类：成员变量：name age
        构造方法：无参 两个参数
        成员方法：getXxx()/setXxx()/show()
*/

class Student{
    private String name;
    private int age;
    public Student(){}

    public Student(String name,int age){
        //System.out.println("这是带多个类型的构造方法");
        this.age = age;
        this.name = name;
    }

    public String getName(){
        return name;
    }

    public void setName(String name){
        this.name = name;
    }

    public int getAge(){
        return age;
    }

    public void setAge(int age){
        this.age = age;
    }

    public void show(){
        System.out.println(name+"---"+age);
```

```
    }
}

//测试类
class StudentTest{
    public static void main(String[] args){
        Student s1 = new Student();
        s1.setName("王若潇");
        s1.setAge(22);
        System.out.println(s1.getName()+"---"+s1.getAge());
        s1.show();

        Student s2 = new Student("王若潇",22);
        s2.show();
    }
}
```

定义成员变量

2018年6月18日 19:43

变量什么时候定义为成员变量：
如果这个变量用来描述这个类的信息时，这个变量就应该定义为成员变量

import放在所有class前

变量的范围越小越好，因为能及时被回收

```
class Demo{
	public int sum(int a,int b){
		return a+b;
	}
}

class Test{
	public static void main(String[] args){
		Demo d = new Demo();
		System.out.println(d.sum(10,20));
	}
}
```

没有必要把a，b定义为成员变量
因为a，b不能描述类

static

2018年6月19日 11:18

```
//针对多个对象有共同的成员变量值时，java提供关键字static修饰
class Person{
	String name;
	int age;
	static String country;
	public Person(){}
	public Person(String name,int age){
		this.name = name;
		this.age = age;
	}
	public Person(String name,int age,String country){
		this.name = name;
		this.age = age;
		this.country = country;
	}
	public void show(){
		System.out.println("姓名"+name+",年龄"+age+",国籍"+country);
	}
}

class PersonDemo{
	public static void main(String[] args){
		Person p1 = new Person("王若潇",22,"中国");
		p1.show();//姓名王若潇,年龄22,国籍中国
		Person p2 = new Person("C罗",33);
		p2.show();//姓名C罗,年龄33,国籍中国
		Person p3 = new Person("梅西",33,"阿根廷");
		p3.show();//姓名梅西,年龄33,国籍阿根廷
		p1.show();//姓名王若潇,年龄22,国籍阿根廷
		p2.show();//姓名C罗,年龄33,国籍阿根廷
	}
}
```

static特点

2018年6月19日 16:03

static特点：（它可以修饰成员变量，还可以修饰成员方法）

1.随着类的加载而加载

2.有限于对象存在

3.被类的所有对象共享

如果某个成员变量是被所有成员变量共享的，定义为静态

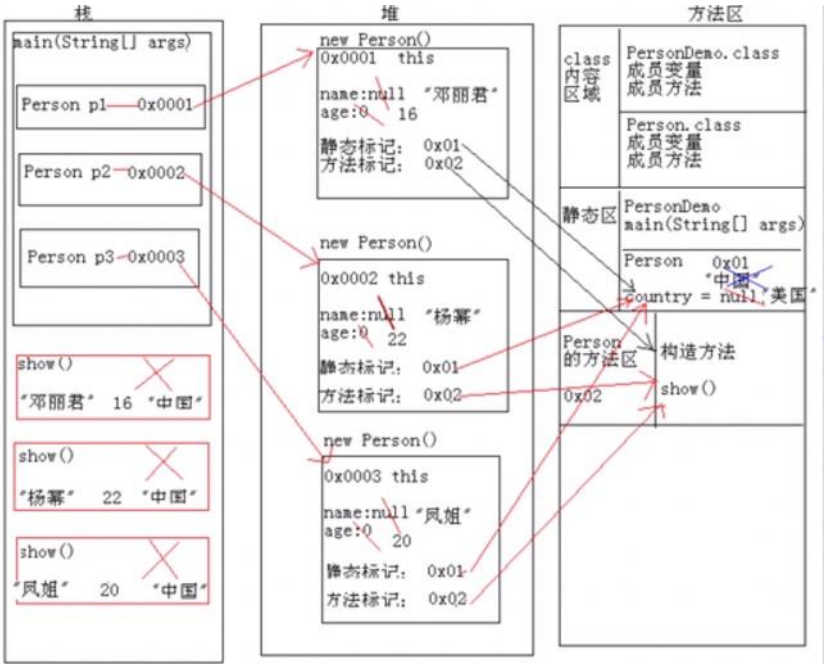
4.**可以通过类名调用**（也可以用对象调用） Teacher.name

推荐使用类名调用，成为类成员

static内存

2018年6月19日 21:56

```
class Person {
        String name;
        int age;
        static String country;

        public Person(){}

        public Person(String name,int age){}

        public Person(String name,int age,String country){}

        public void show() {
                System.out.println(name+"---"+age+"---"+country);
        }
}

class  PersonDemo {
        public static void main(String[] args) {
                Person p1 = new Person("邓丽君",16,"中国");
                p1.show();  "邓丽君"  16  "中国"

                Person p2 = new Person("杨幂",22);
                p2.show();

                Person p3 = new Person("凤姐",20);
                p3.show();

                p3.country = "美国";
                p3.show();
                p1.show();
                p2.show();
        }
}
```



static注意事项

2018年6月19日 21:57

1.在静态方法中是没有this关键字的

静态是随着类的加载而加载，this是随着对象的创造而存在

2.无法从静态上下文访问非静态变量

静态方法只能访问静态变量和静态方法

静态只能访问静态

main方法

2018年6月19日 22:14

main方法的格式
public static void main(String[] args){}

public：公共的，访问权限最大。由于main是被jvm调用，因此权限要最大
static：静态的，不需要创建对象，通过类名就可以，方便jvm的调用
void：方法的返回值是返回给调用者，而main方法是被jvm调用，返回给jvm没有任何意义
main：是一个常见的入口方法
String[] args：这是一个字符串数组 ——————————→

早期是为了接受键盘录入的数据的 **了解即可**
格式是dos命令行中输入
java MainDemo hello world java
args变为hello world java

```
class MainDemo{
	public static void main(String[] args){
		System.out.println(args);//[Ljava.lang.String;@15db9742
		System.out.println(args.length);//0
	}
}
```