

# Java语言基础

【Day07】

# 构造方法和方法重载

## Constructor Overload

# 构造方法的基本概念

- 在Java语言中可以通过构造方法实现对对象成员变量的初始化。构造方法是在类中定义的方法。但不同于其他的方法，构造方法的定义有如下两点规则：
  - 构造方法的名称必须与类名同名。
  - 构造方法没有返回值，但也不能写void。

```
Point(int i, int j) {
```

```
    x = i;
```

```
    y = j;
```

```
}
```

```
Point p = new Point(100, 200);
```

构造方法使用传递的参数对成员变量x和y进行初始化。

new关键字后面是调用构造方法来对成员变量进行初始化。

# 默认的构造方法

- 任何一个编译后的类都必须含有构造方法。如果源程序中没有定义，编译器在编译时将为其添加一个无参的空构造方法（称之为“默认的构造方法”）。
- 例如，先前的Point源文件中没有构造方法，编译时将为其添加如下的构造方法：`Point() {}`
- 当定义了构造方法后，Java编译器将不再添加默认的构造方法。

# 编程经验的分享

- 建议自定义无参构造方法，不要对编译器形成依赖，避免错误发生。
- 当类中有非常量成员变量时，建议提供两个版本的构造方法，一个是无参构造方法，一个是全属性做参数的构造方法。
- 当类中所有成员变量都是常量或者没有成员变量时，建议不提供任何版本构造。

# 项目案例


- 编程实现向Point类添加构造方法


Point(int i, int j) 根据参数创建点对象


Point() 默认创建原点对象

# 构造方法的重载

- 为了方便对一个类定义多个构造方法，这些构造方法都有相同的名称（类名），方法的参数不同，称之为构造方法的重载。
- 在创建对象时，Java编译器会根据不同的参数调用不同构造方法。

Point p1 = new Point(100, 200);  Point(int i, int j) {  
                                  x = i; y = j;  
                                  }

Point p2 = new Point();  Point() {}

Point p3 = new Point(50);  Point(int a) {  
                                  x = a; y = a;  
                                  }

# 成员方法的重载

- 在Java语言中，允许多个方法的名称相同，但参数列表不同，称之为方法的重载(overload)。
- 编译器在编译时会根据其参数的不同，绑定到不同的方法。
- 方法重载要求方法名相同，参数列表不同，返回值类型可以相同也可以不同。最好还是相同。

```
public class PrintStream {
```

```
... ..
```

```
    print(char c) {...}
```

```
    print(int i) {...}
```

```
    print(double d) {...}
```

```
}
```

编译器在编译时会根据其参数不同绑定到不方法。  
例如，`print( 'a' )`调用的方法是`print(char c)`；  
而`print(123)`调用的方法是`print(int i)`。



# 方法重载的意义

- 在方法的调用者看来，似乎只有一个print方法，而它可以处理不同的数据。这样的设计方式，称为重载设计，即设计多个同名但不同参数的方法。适当的使用重载，可以使类的设计变得优雅。

# 项目案例

- 编程实现为Point类添加重载的成员方法：
  - up() – 实现纵坐标减1的功能。
  - up(int dy) – 实现纵坐标减去参数指定数值的功能。
- 测试重载方法的调用规则

# this关键字

# this关键字的作用

- 所有的成员变量不能重名，在同一区域的局部变量不能重名，但成员变量和局部变量可以重名。在局部变量的作用区域之外，变量名代表成员变量，在局部变量的作用区域之内，代表局部变量，如果想使用成员变量，需要this的方式访问。
- 在方法中可以通过this关键字表示“调用该方法的那个对象”。

```
public void down(int y) {  
    this.y += y;  
}
```

这样的写法语义更加明确。print方法的功能可以描述为打印出调用该方法的那个对象的x和y的值，在没有歧义的情况下可以省略this。

# this关键字和方法调用

```
class Point {  
    int x;  
    int y;  
    void down(int y) {  
        this.y += y;  
    }  
}
```

p1.down(100);  
p2.down(200);

一个类可以创建多个对象（存储于堆中），但方法只有一份（存储于方法区中）。对于上面的语句可以这样理解。可以这样理解：在p1.down(100);方法调用的过程中，除了有参数100传递给y之外，还有一个隐式的参数传递给了方法，这个参数就是引用类型变量p1的值，而在方法down中接受这个参数的就是this，因此this.y += y 就是调用该方法的对象即p1所指向的对象的y值增加y。

# this关键字和构造方法

- 在构造方法中，可以用this()调用本类的其它构造。
- this()必须放在构造的第一行。

```
public class Student{  
    public Student(){  
        this( "zhangfei" ,25);  
    }  
    public Student(String name,int age){  
    }  
}
```

# this关键字和空值

- 引用类型变量用于存放对象的地址，可以给引用类型赋值为null，表示不指向任何对象。
- 当某个引用类型变量为null时无法对对象实施访问（因为它没有指向任何对象）。此时，如果通过引用访问成员变量或调用方法，会产生 **NullPointerException** 异常。

```
Point p = null;
```

```
p.print();
```

 会产生 **NullPointerException**

# Java 传参过程



# 传参的过程分析

```
public int max(int ia, int ib) { ... .. }  
int a = 5; int b=6;  
int res = m.max(a,b);
```

1. 为main方法中的变量a、b、res分配空间并赋值。
2. 调用方法max，为max方法的参数变量ia，ib分配空间。
3. 将调用值传递到参数变量中。
4. max方法运行完返回，参数变量空间释放。
5. main方法中的res变量得到返回值。

ib	6	}	max 方法
ia	5		
res	6	}	Main 方法
b	6		
a	5		

# 传参的基本概念

- Java中的方法可以传递参数，参数的传递方法就是值传递。
- 参数有形参和实参，定义方法时写的参数叫形参，真正调用方法时，传递的参数叫实参。调用方法时，会把实参传递给形参，方法内部其实是在使用形参。
- 所谓值传递就是当参数是基本类型时，传递参数的值，比如传递*i*=10，真实传参时，把10赋值给了形参。当参数是对象时，传递的是对象的值，也就是对象的首地址。就是把对象的地址赋值给形参。

# 传参的过程总结

- 参数传递的步骤：
  - 1 分配实参空间，基本类型在栈中赋值，引用类型变量在栈中指向堆中的对象
  - 2 传递参数其实就是在栈中分配形参的空间，然后把栈中实参的值复制过来。
  - 3 在方法中使用形参，方法结束形参出栈(消失)，只剩下实参。
- gc主要针对堆中的对象，栈中数据是随时进出的。判断堆中对象是不是内存垃圾的条件看栈中是否有指向，直接或者间接的指向都可以，没有的就是内存垃圾。

# 方法的递归调用

# 项目案例

- 编程实现参数 $n$ 的阶乘并返回，所谓阶乘就是从1累乘到 $n$ 的结果。

# 递归的基本概念

- 方法自己调用自己就叫递归。  
递归有可能会大幅简化代码的编写。  
递归要考虑性能问题，有些时候可以使用循环而不是递归。
- 递归的使用原则：
  - 1 必须有退出条件。
  - 2 必须使问题变简单，而不是复杂化。

# 项目案例

- 编程实现费式数列中第n项的数值并返回。
- 费式数列：1 1 2 3 5 8 13 21 .....

# 总结与答疑

技术资料







IT兄弟连  
ITXDL.CN

变态严管 让学习成为一种习惯