Atitit.编程语言原理---方法重载的实现与设计 调用方法的原理

[1. 重载包括：普通方法的重载和构造方法的重载 1](#_Toc20251)

[1.1. 横向重载”和“纵向重载” 1](#_Toc20196)

[1.2. 方法签名通过 方法名称，参数数量，参数类型+返回类型 参数顺序（命名参数）来组成 1](#_Toc31195)

[1.3. 重载（重新载选方法流程，如下图所示 2](#_Toc29614)

[2. 重载的实现方法：：argus参数 vs 默认值。可选参数法 可选参数 3](#_Toc26877)

[2.1. \_\_call()函数是php类的默认魔法函数 3](#_Toc10217)

[2.2. Java通过反射调用方法重载 执行方法 3](#_Toc30990)

# 重载包括：普通方法的重载和构造方法的重载

 所谓重载，就是重复使用。而这个重复使用并不是反复使用同一个一模一样的东西，而是只是借用同一个名字而已。根据这个名字，根据不同的需求，提供不一样的功能。这就是重载。说一个例子，“父亲”这个词，其实是重载了的。如果你叫父亲，那是指你的父亲，而如果别人叫父亲，那是别人父亲。这也是一个重载。根据叫的人不同，也就是调用这个名称的对象不一样，产生的意义就不一样，这就是重载。所以，这里说的就是广义上的重载的概念。

## 横向重载”和“纵向重载”

 清楚了重载的本质分析，再来看看，“横向重载”和“纵向重载”的概念。函数特征标形式是在一个类中实现的多个函数版本，相当于在一个类中平铺多种形式，因此可以理解成横向（平铺）方式。而虚函数形式，是通过继承，构成一个垂直的关系。比如父亲跟儿子，属于两代人，是一个垂直的传递关系，父亲在上一层，儿子在下一层。也就是一个纵向的走向。C++中支持了虚拟函数，提供了重载虚函数的功能，因此就是我们说的“纵向重载”。

## 方法签名通过 方法名称，参数数量，参数类型+返回类型 参数顺序（命名参数）来组成

## 重载（重新载选方法流程，如下图所示

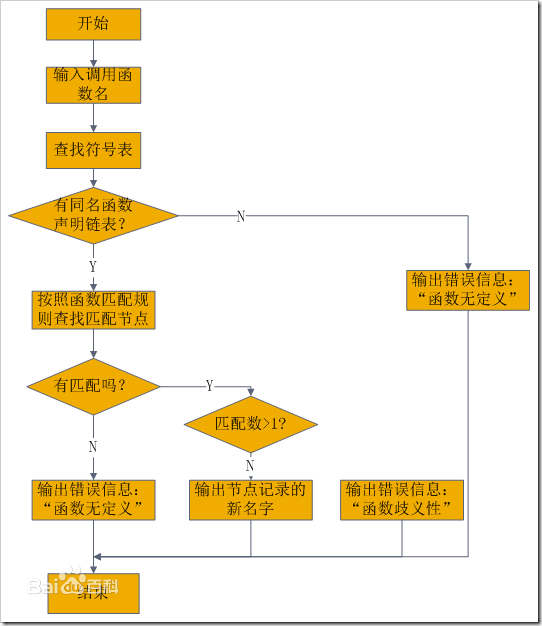
首先得到方法名称，查找符号表，没有跑出ex

通过参数数量判断

通过参数类型判断（强类型语言）

If(nomather()) ex

If(matchCount>1) ex cant select amigo



# 重载的实现方法：：argus参数 vs 默认值。可选参数法 可选参数

Php通过func\_num\_args();js通过argus参数

Python通过默认值

一个语言选择重载或者可选参数两者中的一种就是了。像C# 4这样同时支持这两种形式的，就比较容易混淆。

## \_\_call()函数是php类的默认魔法函数

，\_\_call() 在一个对象的上下文中，如果调用的方法不存在的时候，它将被触发:  
<?php  
class MethodTest {  
     public function \_\_call($name, $arguments) {  
        // Note: value of $name is case sensitive.  
        echo "Calling object method '$name' "  
             . implode(', ', $arguments). "\n";  
     }  
}

## Java通过反射调用方法重载 执行方法

paip.编程语言方法重载实现的原理及python,php,js中实现方法重载 - attilax的专栏 - 博客频道 - CSDN.NET.htm

PHP-PHP重载和多态的实现？ - 德问 编程社交问答.htm

java怎样实现重载一个方法 - 郑树恒的个人空间 - 开源中国社区.htm

C++重载：横向重载（静态重载）与纵向重载（动态重载）-C++语言-编程语言-C++技术网.htm