Atitit.c++ java的对象语法不同原理

* C++允许名字空间级别的常量,变量和函数. 而所有这样的 Java 声明必须在一个类或者接口当中.
* 在 C++ 的声明中,一个类名可以用来声明一个此类对象的值. Java 里没办法做到这点. 在Java里对象不是值. 在 Java 的声明中,一个类名声明的是对此类的一个对象的引用. 而在 C++ 里与之等价的做法是用 "\*" 来声明一个指针.
* 在 C++ 里,"."操作符将一个对象作为一个左操作参数来访问这个对象的成员. 因为对象在 Java 里不是值,所有的对象都通过引用来访问,刚才的做法在 Java 里是无法实现的. 在 Java 里,"." 操作符是将一个对象的引用作为左操作参数来访问这个对象的成员.在C++中和这种做法等价的是 "->".

|  |  |
| --- | --- |
| Foo a;// 声明 a 为一个 Foo 类的对象值,// 使用其缺省的构造函数// 如果你想要用其他的构造函数,// 你可以用 "Foo a(args);" | Foo a;// 声明 a 为一个 Foo 类的对象的引用  a = **new** Foo();// 使用缺省的构造函数初始化// 如果你想要用其他的构造函数,// 你可以用 "Foo a = new Foo(args);" |
|  |  |

|  |  |
| --- | --- |
| Foo b = a;// 拷贝 a 的内容到一个新的 Foo 类的变量 b 当中;// 另一种可以选择的语法是 "Foo b(a)" | Foo b = a.clone();// 拷贝所有a这个实例的成员到b,当且仅当,// Foo 实现了一个 public 的 clone() 方法,// 并且 clone() 返回一个新的这个对象的拷贝 |

|  |  |
| --- | --- |
| Foo \*c;// 声明 c 为指向一个 Foo 类对象的指针(初始值是// 未定义的;可能指向任何地方) | Foo c;// 声明 c 为一个指向 Foo 对象的指针// (如果 c 是一个类的成员,那么初始值为空;// 如果 c 是一个局部变量那么你在使用之前必须// 对它进行初始化) |
| c = new Foo();// 将 c 绑定为一个新的 Foo 对象的引用 | c = **new** Foo();// 将 c 绑定为一个新的 Foo 对象的引用 |
| Foo \*d = c;// 将 d 绑定为和 c 同一个对象的引用 | Foo d = c;// 将 d 绑定为和 c 同一个对象的引用 |
| c->x = 5;// 修改 c 指向的对象 | c.x = 5;// 修改 c 指向的对象 |
| a.bar(5); // 对 a 调用 Foo::bar()  c->bar(5); // 对 \*c 调用 Foo::bar() | a.bar(5); // 对 a 调用 Foo.bar()  c.bar(5); // 对 c 调用 Foo.bar() |
| cout << d->x << endl;// 输出 5,因为 d 引用的对象和 c 一样 | System.out.println(d.x);// 输出 5,因为 d 引用的对象和 c 一样 |

注意：既然JAVA中不存在指针，那么也不存在－>操作符。这里还有一些非常重要的”省略”

**(3) 在Java中，类定义采取几乎和C++同样的形式。**

但没有标志结束的分号。没有类声明，只有类定义。

**(4) Java中没有作用域范围运算符“::”.**

Java利用点号做所有的事情，但可以不用考虑它，因为只能在一个类里定义元素。即使那些方法定义，也必须在一个类的内部，所以根本没有必要指定作用域的范围。我们注意到的一项差异是对static方法的调用:使用ClassName.methodName()。

除此以外，package（包）的名字是用点号建立的，并能用import关键字实现C++的“#include”的一部分功能。

例如下面这个语句：

的方式来取代，而#include在Java中是不需要的，因为在Java中程序在执行时，会把类型数据记录在对象实体之中，我们不需要靠一些标头文件(header file)来知道我们使用的对象或数值是属于什么数据类型。

ava去掉了指针类型，并不表示程序设计师在开发高级数据结构，像堆栈(stack)、队列(queue)、二元树(binarytree)时，都必须要像在传统Basic上，利用大范围的数组来自行模拟系统内存，自行建构类似指针的表示方式。

相反地，Java提供了和Lisp语言中相似的Reference类型，通过Reference去读取配置到的内存内容，可以确保不会去读取到不属于自己的内存空间，而另一方面，程序的执行系统也可以动态地去做内存垃圾回收的工作，将没有被reference参考到的内存空间回收给系统使用。

C++和java技术特性对比 - lile269的专栏 - 博客频道 - CSDN.NET.html