

基本概念

作者: 330974900

1.OOP 中唯一关系的是对象的接口是什么, 就像计算机的销售商她不管电源内部结构 是怎样的, 他只关系能否给你提供电就行了, 也就是只要知道 **can or not** 而不是 **how and why**。所有的程序是由一定的属性和行为对象组成的, 不同的对象的访问通过函数调用来完成, 对象间所有的交流都是通过方法调用, 通过对封装对象数据, 很大 限度上提高复用率。

2.OOP 中最重要的思想是类, 类是模板是蓝图, 从类中构造一个对象, 即创建了这个类的一个实例 (**instance**)。

3.封装:就是把数据和行为结合起在一个包中)并对对象使用者隐藏数据的实现过程, 一个对象中的数据叫他的实例字段(**instance field**)。

4.通过扩展一个类来获得一个新类叫继承(inheritance), 而所有的类都是由 **Object** 根超类扩展而得, 根超类下文会做介绍。

5.对象的 3 个主要特性

ehavior---说明这个对象能做什么。

tate---当对象施加方法时对象的反映。

dentify---与其他相似行为对象的区分标志。

每个对象有唯一的 **indentity** 而这 **3** 者之间相互影响。

6.类之间的关系:

use-a :依赖关系

has-a :聚合关系

is-a :继承关系--例:**A** 类继承了 **B** 类, 此时 **A** 类不仅有了 **B** 类的方法, 还有其自己的方法.(个性存在于共性中)

7.构造对象使用构造器:构造器的提出, 构造器是一种特殊的方法, 构造对象并对其初始化。

例:**Data** 类的构造器叫 **Data**

ew Data()---构造一个新对象, 且初始化当前时间。

Data happyday=new Data()---把一个对象赋值给一个变量 **happyday**，从而使该对象能够多次使用，此处要声明的使变量与对象变量二者是不同的。**new** 返回的值是一个引用。

构造器特点:构造器可以有 **0** 个，一个或多个参数；构造器和类有相同的名字；一个类可以有多个构造器；构造器没有返回值；构造器总是和 **new** 运算符一起使用。

8.重载:当多个方法具有相同的名字而含有不同的参数时，便发生重载.编译器必须挑选出调用哪个方法。

9.包(package)Java 允许把一个或多个类收集在一起成为一组，称作包，以便于组织任务，标准 **Java** 库分为许多包。**java.lang java.util java. net** 等，包是分层次的所有的 **java** 包都在 **java** 和 **javax** 包层次内。

10.继承思想:允许在已经存在的类的基础上构建新的类，当你继承一个已经存在的类时，那么你就复用了这个类的方法和字段，同时你可以在新类中添加新的方法和字段。

11.扩展类:扩展类充分体现了 **is-a** 的继承关系，形式为:**class (子类) extends (基类)**。

12.多态:在 **java** 中，对象变量是多态的.而 **java** 中不支持多重继承。

13.动态绑定:调用对象方法的机制。

(1)编译器检查对象声明的类型和方法名。

(2)编译器检查方法调用的参数类型。

(3)静态绑定:若方法类型为 **private static final** 编译器会准确知道该调用哪个方法。

(4)当程序运行并且使用动态绑定来调用一个方法时，那么虚拟机必须调用 **x** 所指向的对象的实际类型相匹配的方法版本。

(5)动态绑定:是很重要的特性，它能使程序变得可扩展而不需要重编译已存代码。

14.final 类:为防止他人从你的类上派生新类，此类是不可扩展的。

15.动态调用比静态调用花费的时间要长。

16.抽象类:规定一个或多个抽象方法的类本身必须定义为 **abstract**。

例: **public abstract string getDescription**

17.Java 中的每一个类都是从 **Object** 类扩展而来的。

18.object 类中的 **equal** 和 **toString** 方法。

equal 用于测试一个对象是否同另一个对象相等。

toString 返回一个代表该对象的字符串,几乎每一个类都会重载该方法,以便返回当前状态的正确表示。

(**toString** 方法是一个很重要的方法)

19.通用编程:任何类类型的所有值都可以同 **object** 类性的变量来代替。

20.数组列表:**ArrayList** 动态数组列表,是一个类库,定义在 **java.util** 包中,可自动调节数组的大校

21.class 类 **object** 类中的 **getClass** 方法返回 **Class** 类型的一个实例,程序启动时包含在 **main** 方法的类会被加载,虚拟机要加载他需要的所有类,每一个加载的类都要加载它需要的类。

22.class 类为编写可动态操纵 **java** 代码的程序提供了强大的功能反射,这项功能为 **JavaBeans** 特别有用,使用反射 **Java** 能支持 **VB** 程序员习惯使用的工具。能够分析类能力的程序叫反射器,**Java** 中提供此功能的包叫 **java.lang.reflect** 反射机制十分强大。

A.在运行时分析类的能力。

B.在运行时探索类的对象。

C.实现通用数组操纵代码。

D.提供方法对象。

而此机制主要针对是工具者而不是应用及程序。

反射机制中的最重要的部分是允许你检查类的结构.用到的 **API** 有:

java.lang.reflect.Field 返回字段。

java.reflect.Method 返回方法。

java.lang.reflect.Constructor 返回参数。

方法指针:**java** 没有方法指针, 把一个方法的地址传给另一个方法, 可以在后面调用它, 而接口是更好的解决方案。

23.接口 (Interface)说明类该做什么而不指定如何去做, 一个类可以实现一个或多个 **interface**。

24.接口不是一个类, 而是对符合接口要求的类的一套规范。若实现一个接口需要 **2** 个步骤:

A.声明类需要实现的指定接口。

B.提供接口中的所有方法的定义。

声明一个类实现一个接口需要使用 **implements** 关键字 **class actionB implements Comparable** 其 **actionb** 需要提供 **CompareTo** 方法, 接口不是类, 不能用 **new** 实例化一个接口。

25.一个类只有一个超类, 但一个类能实现多个接口。**Java** 中的一个重要接口: **Cloneable**

26.接口和回调.编程一个常用的模式是回调模式, 在这种模式中你可以指定当一个特定时间发生时回调对象上的方法。

例:**ActionListener** 接口监听。

类似的 **API** 有:

java.swing.JOptionPane

java.swing.Timer

java.awt.Toolkit

27.对象 clone:clone 方法是 **object** 一个保护方法，这意味着你的代码不能简单的调用它。

28.内部类:一个内部类的定义是定义在另一个内部的类。原因是：

A.一个内部类的对象能够访问创建它的对象的实现，包括私有数据。

B.对于同一个包中的其他类来说，内部类能够隐藏起来。

C.匿名内部类可以很方便的定义回调。

D.使用内部类可以非常方便的编写事件驱动程序。

29.代理类(proxy):

A.指定接口要求所有代码

B.object 类定义的所有的的方法(**toString equals**)

30.数据类型:Java 是强调类型的语言，每个变量都必须先申明它都类型，**java** 中总共有 **8** 个基本类型。**4** 种是整型，**2** 种是浮点型，一种是字符型，被用于 **Unicode** 编码中的字符，布尔型。