内容

- Object 类
- 对象转型(casting)
- 多态
- 抽象类
- 接口
- 可变参数

一、 Object 类

Object 类是所有 Java 类的根基类

如果在类的声明中未使用 extends 关键字指明其基类，则默认基类为 Object 类

```
class Person{

}
```

等价于

```
class Person extends Object{
}
```

对象的实例化过程

实例化一个类是从最顶级的超类开始实例化的， 是一层一层的包裹结构. “先父类后子类,先静态后成员”。

(1)toString 方法

toString：对象的字符串表示

Object 类中定义有 public String toString() 方法，其返回值是 String 类型，用来描述当前对象的有关信息。

在进行 String 与其他类型数据的连接操作时（如：System.out.println("hello " + person)），将自动调用该对象类的 toString()方法

可以根据需要在用户自定义类型中重写 toString()方法。

(2)equals 方法

equals:比较相等，默认地址比较(" 第一个盒子的比较 ")，要比较第二个盒子需要重写该方法

Object 类中定义有：`public boolean equals(Object obj)`方法

提供定义对象是否"相等"的逻辑

Object 的 equals 方法定义为：x.equals(y)当 x 和 y 是同一个对象的引用时返回 true,否则返回 false

JDK 提供的一些类，如 String，Integer，Date 等，都已经重写了 Object 的 equals 方法，调用这些类的 equals 方法，x.equals(y),当 x 和 y 所引用的对象是同一类对象且属性内容相等时（并不一定是相同对象），返回 true 否则返回 false.

可以根据需要在用户自定义类型中重写 equals 方法

练习：重写前面定义的 "Person" 类和"Student"类的 toString 及 equals 方法，并测试。

二、 对象转型（casting）

一个基类的引用类型变量可以 "指向" 其子类的对象。

一个基类的引用不可以访问其子类对象的新增成员（包括属性和方法）。

可以使用 "引用变量 **instanceof** 类名" 来判断该引用类型变量所 "指向" 的对象是否属于该类或该类的子类。

子类对象可以当作基类的对象来使用，称作向上转型(upcasting)，反之称为向下转型(downcasting)

例 1：有

Animal 类（父类）

Cat 类 (子类)

Dog 类（子类）

测试：父类引用 = 父类对象

子类引用 = 子类对象

父类引用 = 子类对象

父类引用调用父类继承而来的属性和方法

父类引用调用子类特有属性和方法(向下转型)

子类对象 instanceof 父类

强制类型转换后能调用子类特有成员（属性和方法）

例 2：

测试将父类形参用子类对象作为实参使用

1、 重载

2、 通过强制类型转换调用子类特有成员

三、 多态

静态绑定(静态联编): 在编译期完成，可以提高代码执行速度。静态绑定的方法包括：

1. 静态方法
2. 构造器
3. private 方法
4. 用关键字 super 调用的方法

动态绑定(动态联编)：指在“执行期间（而非编译期间）”判断所引用对象的实际类型，根据其实际的类型调用其相应的方法。 这虽然让我们编程灵活，但是降低了代码的执行速度。这也是 JAVA 比 C/C++速度慢的主要因素之一。

多态，polymorphism 即多种形态，模糊策略，以不变应万变，使用多态可以编写更加通用的代码。

多态的概念发展出来，是以封装和继承为基础的。子类以父类的身份出现，但做事情时还是以自己的方法实现。

相同的事物，调用其相同的方法，参数也相同时，但表现的行为却不同。

要发生多态有三个必要条件：要有继承，要有重写，父类引用指向子类对象

父类是父类

子类是子类

子类是父类

父类不是子类

子类不是其他子类

多态例子：

例 1(必须掌握看透):

Animal Cat Dog

例 2 (提高理解多态)：先思考再运行看结果

做题四大原则：

1、继承链，自己没有找父亲；

2、编译看类型+确定方法表，运行找对象

3、就近最优原则：自己没有找父亲

4、发生多态，基类对子类的新增方法不可见

四、 抽象类

用 abstract 关键字来修饰一个类时，这个类叫做抽象类；用 abstract 来修饰一个方法时，该方法叫做抽象方法。

含有抽象方法的类必须被声明为抽象类，抽象类必须被继承，抽象方法必须被重写

抽象类不能被实例化

抽象方法只需声明而不需要实现

例：动物高兴了都会叫 但对于不同种类的动物，各自的叫声不同，动物（这个父类）的叫法/叫声永远不可能满足子类的需求

五、 接口

接口（interface）是抽象方法和常量值的定义的集合。

从本质上讲，接口是一种特殊的抽象类，这种抽象类中只包含常量和方法的定义，而没有变量和方法的实现

从语义上可以理解为，对于某一种动作、行为、功能的抽象，我们将其定义为接口，作为一种标准完全的规范而已，不适宜定义为类。例如：飞这个功能，飞机可以飞（具有飞这个功能），小鸟能飞（具有飞这个功能），昆虫也能飞，一般我们不会定义一个类 Fly,从语义上不通，所以这只是一种功能，一个规范，我们可以将其定义为一种接口，供其他类来实现

接口定义举例：run 这个功能

接口的特性

多个无关的类可以实现同一个接口

一个类可以实现多个无关的接口

与继承关系类似，接口与实现类之间存在多态性

定义 Java 类的语法格式：

<modifier> class <name>[extends<superclass>]
[implements<interface>[,<interface>]…]{…}

接口中声明属性默认为 public static final 的，也只能是 public static final 的；

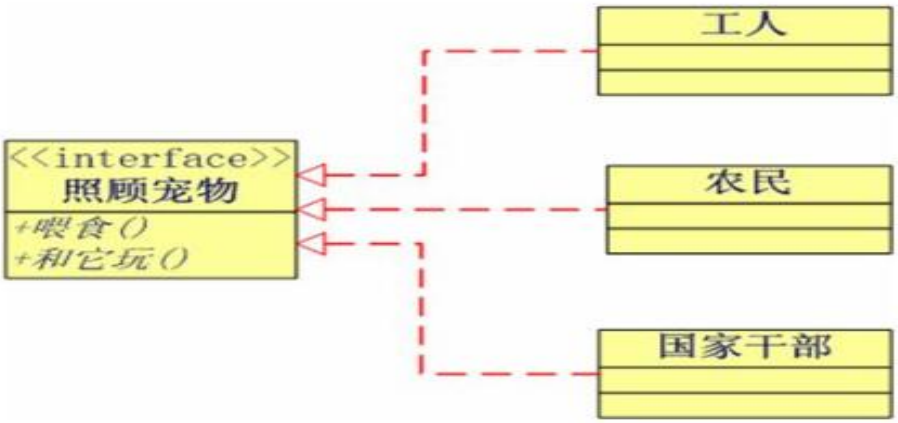
接口中只能定义抽象方法，而且这些方法默认为 public 的，也只能是 public 的

接口可以继承其他的接口，并添加新的属性和抽象方法

接口不能实现另一个接口，但可以继承多个其它接口

接口举例：Sing 这种能力 Paint 这种能力

练习：设计接口实现下面结构



面向对象三大特性的理解：

在编写代码时，我们追求”高内聚 低耦合” ,达到重用与规范，则需要使用面向对象的三大特性来实现:

封装：encapsulation 隐藏信息，整合数据和操作

继承:inheritance 延续+扩展父类信息

多态:polymorphism 模糊策略 以不变应万变

封装作用： a) 实现专业的分工，工作中的分模块、分功能开发。b) 隐藏信息和实现细节。使得对代码的修改更加安全和容易

继承作用：实现代码的复用，延续+扩展父类信息

多态作用：以不变应万变（如 USB 接口，只要你实现了我的标准，就能插上电脑）

注意:**java 三大特性虽说简单，但真正能理解其中的含义，没有个一年半载的学习，是理解不了的。**

六、可变参数

Java1.5 增加了新特性：可变参数：适用于参数个数不确定，类型确定的情况，java 把可变参数当做数组处理。

注意：可变参数必须位于最后一项。当可变参数个数多余一个时，必将有一个不是最后一项，所以只支持有一个可变参数。因为参数个数不定，所以当其后边还有相同类

型参数时，java 无法区分传入的参数属于前一个可变参数还是后边的参数，所以只能让可变参数位于最后一项。

可变参数的特点：

（1）只能出现在参数列表的最后；

（2）...位于变量类型和变量名之间，前后有无空格都可以；

（3）调用可变参数的方法时，编译器为该可变参数隐含创建一个数组，在方法体中以数组的形式访问可变参数。

例：

```
public class Param {
	public static void main(String[] args) {
		/*System.out.println(add(3,6));
		System.out.println(add(3,6,8));*/
		System.out.println(add(3,4,5,6,67));
	}
	//若不使用可变参数，则需要重载
	/*public static int add(int a,int b){
		return a+b;
	}
	public static int add(int a,int b,int c){
		return a+b+c;
	}*/
	/*
	public static int add(int a,int ... arr){
		int sum=a;
		for(int i=0;i<arr.length;i++){
			sum +=arr[i];
		}
		return sum;
	}
}
```

七、面向对象总结

内存分析贯穿始终

对象和类的概念

类（对象）之间的关系

面向对象设计思想

class -- 类

new -- 对象

引用的概念

构造方法的概念

构造方法重载

this

static

package & import

private default protected public

extends

Override/Overloading

final

Object

toString

equals

upcasting downcasting

polymorphysm/dynamic bingding

abstract class

interface

implements

思考作业

1、何为多态?多态的作用是什么?

2、多态的两种应用，试编写实例

3、多态可以调用子类的一切方法，对吗？为什么?

4、ClassCastException 错误是如何产生的?

5、instanceof 的作用是什么？任何地方都可以使用 instanceof 来判断，对吗？

6、何为抽象类?有什么作用?

7、抽象类不能实例化，因此不需要构造器，这句话对吗?

8、抽象类可以没有抽象方法，抽象方法一定存在于抽象类中，对吗?

9、抽象类的子类必须实现所有的方法，否则编译错误，对吗?

10、抽象类为部分规范(不变+可变)，试编写模板模式(也称为钩子模式)实例。

11、接口与抽象类的联系与区别是什么？

12、接口如何定义？

13、请将以下修饰符 补充完整

____interface Usb{

_____int MAX_VALUE =1024;

_____void use();

}

14、空接口没有任何的意义，不必在意它们的存在，这句话对吗？为什么

15、Java 中是单继承类多实现接口，对吗？

16、Java 中的接口是多继承接口，对吗？

寄语

千里之行，始于足下；

对的，坚持；错的，放弃！

你可以很有个性，但某些时候请收敛！！！

谢谢大家