# Day 09

## Object类

|  |
| --- |
| 在java中任何类都会直接或者间接的继承Object类。  当定义一个类的时候，如果没有显式的使用extends关键字来声明继承某个类，则该类默认继承Object，所以Object类中方法，被所有的类继承过来了。**Object类可以看成是java中的“上帝”。**  **如果一个类****显式的继承某个类，则该类不再直接继承Object类.(Java的单继承，亲爹只能有一个)**  public class Person {  }  与  Pubolic class Person extends Object{  }  是等价的。 |

### getClass()方法(暂时了解，反射重点讲解)

|  |
| --- |
| 在Object中getClass()方法声明如下：  Class  该方法是一种native方法，即方法的实际实现是用c或c++写的。//JNI NDK  该方法的返回值是调用该方法的对象所属类的类型。此方法无法在子类中被覆写。 |

### hashCode()方法(重要)

|  |
| --- |
| 在Object中hashCode()方法声明如下：  该方法是一种native方法，即方法的实际实现是用c或c++写的。  该方法的返回值为**十进制**的int值，可以理解为调用该方法的对象在堆内存中的地址值(唯一)。该方法一般在子类中进行覆写。 |

### toString()方法(重要)

|  |
| --- |
| 在Object中toString()方法定义如下：  该方法的返回值是String类型，描述当前对象的有关信息。默认是返回：调用该方法的对象所属类的类名+@+十六进制的hashCode值。  该方法在子类中一般需要覆写，具体覆写样式根据业务需求来定。  Java SDK提供的一些系统类均对此方法进行了覆写：例如String、Date等(后续重点讲解) |

### equals()方法(重要)

|  |
| --- |
| 在Object中equals()方法的声明如下：  该方法的返回值为boolean值。默认情况下当调用该方法的对象和参数传入的对象时同一对象时返回true，否则返回false。  一般情况，根据实际的业务需求对该方法进行覆写，来定义自己的“相等”逻辑。  提示：当如果需要把对象存入set集合中，需要覆写该方法(后续重点讲解)  “==”和“equals”的区别：  1、== 比较的是两个对象是否为同一个对象(比较的内存地址是否相等)  2、equals默认也是比较的两个对象是否为同一对象。(从Objcet继承过来的)  3、一般通过覆写equals方法来定义两个对象逻辑上是否相等。 |

## 抽象类

### 为什么需要抽象类

|  |
| --- |
| 有些类没有必要实例化，或者实例化之后没有实在意义,或者类中的某些方法不知道如何实现，只能等到子类进行实现，这些情况下我们应该禁止对该对象进行实例化。如果想禁止该类在外界进行实例化，可以有两种方式：   1. 所有构造方法私有。会导致该类既不能被实例化，也不能被继承，该类基本也就废了。没有多大意义。 2. **可以把该类标记为抽象类。那么该类则无法在外界进行实例化，但是可以通过继承该类，来实例化该类的子类。** |

### 抽象类定义

|  |
| --- |
| 用abstract关键字修饰的类，我们称之为抽象类。  例如：  **public abstract class Animal{**  **}**  **其中abstract和权限修饰符的位置可以互换。** |

### 抽象方法定义

|  |
| --- |
| 用abstract修饰，并且没有方法体（没有大括号）的方法称为抽象方法。  例如：  **public abstract void sleep(); //抽象方法 用abstract修饰，没有方法体：大括号直接省略掉** |

### 抽象类特点

|  |
| --- |
| 1. 抽象类必须使用abstract来修饰 2. 不能直接使用new关键字来创建抽象类的对象。 3. 抽象类中可以有普通类中有的任何元素 (成员变量、静态变量、实例方法、静态方法等) 4. **抽象类中可以没有抽象方法。** 5. **如果一个类中出现了抽象方法，则该类必须标记为抽象类。** 6. 子类继承抽象类时，父类中的抽象方法可以实现也可以不实现，如果抽象方法没有被实现，则子类也必须标记为抽象类。 |

## 接口

### 接口的概念

|  |
| --- |
| 抽象类可以理解为是从多个类中抽象出来的模板，不但允许有抽象方法，还可以有变量和普通方法等。如果抽象的更彻底，只留下常量和抽象方法，则此时的抽象类有了新的名字：接口。  **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静态常量和方法的定义，而没有变量和方法的实现。** |

### 接口的声明

|  |
| --- |
| 接口声明语法：  [权限修饰符] **interface** 接口名 [**extends** 父接口1, 父接口2] {  //零到多个常量定义…  int MAX = 1; //定义变量，系统会自动添加public static final，把变量变成常量  public static final int MAX = 1; //与上面定义的效果一样  //零到多个抽象方法定义…  void speak(); //声明抽象方法，可以省略 public 和abstract的修饰符  **public abstract** void speak(); //与上面效果一样  String test(int a); //可以有返回值，也可以传递参数  }  说明：   1. 权限修饰符，和class的权限修饰符相同，只有public和default两种。如果是default，则对同一包内其他类或接口可见。 2. interface 标示接口的关键字。 3. extends 后面跟接口列表，表示该接口继承的父接口。接口只能继承接口，不能继承类。多个父接口之间用英文的”,”(逗号)隔开。 4. 声明常量时可以像声明变量一样声明，但是系统会自动把该变量转成常量 \*\*\* 5. 声明抽象方法。修饰符可以完全省略，系统会默认接口中的所有方法均是public abstract的，如果一定要添加修饰符，只能添加public和abstract |

### 接口的实现

|  |
| --- |
| 由于接口只有抽象方法，所以接口与抽象类一样不能直接创建对象。我们一般说**抽象类的子类**，说**接口的实现类**。  一个类实现接口使用关键字**implements。**语法格式：  [public][abstract] class 类名 implements 接口1, 接口2 …  说明：   1. 实现接口必须使用关键字implements。 2. 一个类可以实现多个接口，多个接口之间使用英文逗号隔开。 3. 类实现接口支持多实现。实现类必须实现所有接口的所有抽象方法，如果不能完全实现，则该类必须标记为抽象类。 4. 一个类实现多个接口的同时，还可以来继承一个类。继承类的时候写法是：**必须先写继承，再写实现。** |

接口定义：

|  |
| --- |
|  |

实现类：

|  |
| --- |
|  |

## 接口的用途

|  |
| --- |
| (1)通过接口实现不相关类的相同行为,而无需考虑这些类之间的关系.  (2)通过接口指明多个类需要实现的方法  (3)通过接口了解对象的交互界面,而无需了解对象所对应的类  (4)使用接口，具体代码可以不依赖具体实现类，当具体实现类有变化时，代码不需调整。使用接口使软件更容易扩展。 \*\*\*\*\*\*  **面向对象的最高境界，是面向接口编程！！！** |

## 抽象类和接口对比

|  |
| --- |
|  |