**类的继承：**

1. 使用extends关键字可以实现类的继承，实现类的扩充。
2. 在Java中只允许单继承，即一个子类只能继承一个父类；同时允许多层继承，即一 个子类可以有一个父类，一个父类还可以有一个父类。、
3. 在继承关系中，子类是不能直接访问父类中的私有成员的，但是子类可以调用非私 有的方法间接的访问。

**子类对象的实例化过程：**  
 子类对象在实例化之前必须首先调用父类中的构造方法然后再调用自己的构造方法完 成实例化。

**方法覆写：**

1. 方法覆写指的是在继承关系中，子类中定义了与父类中同名的方法。但方法覆写需 要考虑到访问权限，即被子类覆写的方法不能拥有比父类方法更加严格的访问权限。
2. 方法覆写后，子类对象调用的将是被覆写后的方法，即调用子类中的方法，如果此 时要在子类中访问父类中的方法，则使用super关键字即可。
3. 父类中private的方法无法被覆写，即子类覆写父类中private方法不算是方法的覆 写。

**方法覆写与重载的区别：**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 区别点 | 方法重载 | 方法覆写 |
| 1 | 单词 | Overloading | Overriding |
| 2 | 定义 | 方法名称相同，参数的类型或个数不同 | 方法名称、返回值类型、参数类型和个数全部相等 |
| 3 | 对权限没有要求 | 被覆写的方法不能拥有更严格的访问权限 |
| 4 | 发生范围 | 发生在一个类中 | 发生在继承关系中 |

**属性覆写：**

属性覆写指的是在子类中定义了与父类同名的属性，属性覆写后，子类对象调用的将是 本类中的属性，如果需要在子类中调用父类中的属性，则可以使用super关键字。（需 要注意属性的访问权限问题）。

**super关键字：**

1. 调用父类中的属性或方法。
2. 调用父类中的构造方法。使用super关键字调用父类中的构造方法时，必须放在子 类构造方法的首行，且不能与this调用构造方法同时存在，因为使用this调用构造 方法也必须放在构造方法的首行。

**this关键字与super关键字的区别：**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 区别点 | this关键字 | super关键字 |
| 1 | 访问属性 | 访问本类中的属性，如果没有则从父类中找 | 访问父类中的属性 |
| 2 | 访问方法 | 访问本类中的方法，如果没有则从父类中找 | 访问父类中的方法 |
| 3 | 调用构造 | 调用本类构造，必须放在构造方法首行 | 调用父类构造，必须放在子类构造方法的首行 |
| 4 | 特殊作用 | 表示当前对象 | 无此概念 |

**final关键字：**

1. 使用fianl声明的类不能被继承。
2. 使用final声明的方法不能被子类覆写。
3. 使用final声明的变量即为常量，常量不可以被修改，并且常量名要大写。

使用public static final声明的变量为全局常量。

**抽象类的定义与使用：**

抽象类是一种专门用来充当父类的类，使用abstract关键字 定义抽象类和抽象方法。

1. 可以创建抽象类对象，但抽象类对象不能实例化对象。
2. 抽象类中不一定有抽象方法，但有抽象方法的类必须定义为抽象类。
3. 抽象方法只须声明不需要实现，即抽象方法只有方法头，没有方法体。
4. 抽象方法的访问权限不能为private，因为private声明的方法无法被覆写。
5. 抽象类被子类继承后，子类必须覆写抽象类的全部抽象方法，除非子类是也是抽象 类。
6. 抽象类中的构造方法一般通过子类super关键字调用。
7. 抽象类和抽象方法不能使用fianl关键字声明，因为final关键字声明的类不能被继承， 声明的方法不能被覆写。

需要注意的是，抽象类的与普通类的区别仅仅是不能实例化对象和有抽象方法。

**接口的基本概念：**

接口是由全局常量和公共的抽象方法所组成，使用interface关键字定义接口。由于接 口的定义已明确声明接口的组成，所以，接口中的全局常量和抽象方法可以简写。

1. 子类可以使用implements关键字实现接口，同时一个子类可以实现多个接口，子类 实现接口后必须覆写接口中的全部抽象方法。
2. 如果在开发中一个子类既要继承父类的同时又要实现接口的话，可以按照以下格式：

class 子类 extends 父类 implements 接口A，接口B{  
 }

1. 抽象类也可以实现一个或多个接口，且可以根据需要来决定是否覆写其中的抽象方 法。
2. 在Java中，允许接口使用extends关键字继承一个或多个接。但不允许接口继承类。

可以创建接口对象，但接口不能实例化对象。

**对象的多态性：**

对象的多态性是多态性的体现之一，对象的多态分为向上转型和向下转型。

1. 向上转型：向上转型指的是子类对象转为父类对象。向上转型后，父类对象调用的 将是父类中的方法，但如果父类中的方法被子类覆写了，则调用子类中覆写后的方 法。需要注意的是，向上转型后，父类对象不能调用子类特有的方法。

**父类 父类对象=子类对象；**

1. 向下转型：向下转型指的是父类对象转换为子类对象。向下转型时，需要进行强制 转换。并且在发生对象的向下转型时，一定要发生对象的向上转型，否则父类不知 道谁是它的子，会出现转换异常。向下转型后，子类对象调用的将是子类自己的方 法，如果子类中没有，则从父类中查找。

**子类 子类对象=（子类）父类对象；**

对于向上转型，系统将自动完成，而对于向下转型，必须明确声明要转换的类型。

**instanceof关键字：**

1）在Java中使用instanceof关键字判断一个对象是否是某个类的实例，结果返回布尔 值。 **对象 instanceof 类名称**

2）通过子类对象实例化的父类对象同时是子类和父类的实例，同时，通过父类对象实 例化的子类对象同时也是子类和父类的实例。

一般在程序中，发生向下转型时使用intanceof关键字进行判断，以保证正确性。

**抽象类与接口的实例化：**

在Java中可以通过对象的多态性为抽象类对象和接口对象进行实例化。

**抽象类的实际应用（模板设计）：**

抽象类在实际中的应用更多的是用来当做模板，即将两个类或两个以上的类的共同属性 和共同方法拿出来编写一个抽象类，然后这些类继承抽象类，对于这两个类的各自特有 的属性，可以在各自类中进行声明，特有的行为则可以在抽象类中写成一个抽象方法， 然后这些类覆写其抽象方法，完成不同的特殊内容。

**接口的实际应用（制定标准）：**

接口在实际中更多的作用是用来制定标准的，只要是实现此接口，就相当于拥有了此标 准。

**工厂设计模式：**

工厂设计模式在子类和接口间加入一个了一个过渡端，通过此客户端取得接口的实例

化对象，一般将这个客户端称为工厂类。

**代理设计模式：**

代理设计模式是指用代理主题来操作真实主题，真实主题执行具体的业务操作，而一些 其他相关的业务则交个代理主题。真实主题一般只执行更重要的业务或者只关注某种结

果，而代理主题执行一些次要的业务，或者将某种行为的可行性反馈给真实主题。代理 设计模式的特征：

1. 真实类与代理类有着共同接口。
2. 在代理类中存在一个真实类的对象作为代理类的属性。

**适配器设计模式：**

对于Java来说，如果一个类要实现一个接口，则必须覆写接口中的全部抽象方法，但 如果此接口中抽象方法过多，但子类又用不着这么多的抽象方法，则使用子类实现接口 很麻烦。所以此时就需要一个中间的过渡，但是此过渡类又不希望直接使用，所以将过 渡类定义为抽象类最为合适，即一个接口首先被抽象类先实现，并在此抽象类中覆写接 口中的全部方法，且方法体为空。则以后的子类直接继承此抽象类，就可以由选择地覆 写抽象类的所需要的方法了。这样的过渡类叫做适配器。在图形界面编程的事件处理中

经常使用此设计模式。

**内部类的扩展：**

1）在一个抽象类中可以定义多个抽象类或接口，在接口中也可以定义多个接口或抽象 类。

2）对于一个子类来讲，如果继承或实现了拥有内部抽象类或内部接口的抽象类或接口， 该子类也可以在内部声明一个类来继承或实现其内部类或接口的内部抽象类或内部 接口。

**抽象类与接口之间的区别和比较：**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 区别点 | 抽象类 | 接口 |
| 1 | 定义 | abstract声明的类 | 抽象方法和全局常量的集合 |
| 2 | 组成 | 构造方法、属性、方法、抽象方法 | 常量、抽象方法 |
| 3 | 使用 | 子类通过extends继承抽象类 | 子类通过implements实现接口 |
| 4 | 关系 | 抽象类可以实现多个接口 | 接口不能继承抽象类，但允许继承多个接口 |
| 5 | 常用设计模式 | 模板设计 | 工厂设计、代理设计、制定标准 |
| 6 | 对象 | 都通过对象的多态性实例化对象 | |
| 7 | 局限 | 抽象类有单继承的局限 | 接口没有单继承的局限 |
| 8 | 实际应用 | 模板设计 | 制定标准 |
| 9 | 选择 | 在两者都可以使用的话，优先使用接口，能够避免单继承的局限 | |
| 10 | 特殊 | 一个抽类中可以包含多个接口，一个接口中也可以包含多个抽象类 | |

**接口定义的加强：**

从JDK1.8开始，允许在接口中定义使用default声明的普通方法和使用static声明的静 态方法。

1. default声明的普通方法需要通过接口对象调用。
2. static声明的方法可以直接使用接口名称进行调用。

对于接口定义的普通方法，会被子类所继承。

**Object类：**

1. 在Java中所有的类都有一个公共的父类Object类，只要一个类没有明显的继承一个 类，则肯定是Object类的子类。即Object类是所有类的父类。
2. 既然Object类是所有类的父类，则每个对象都可以调用Object类中的几个方法。

**Object类中的主要方法：**

在Object类中有着很多的方法，其中有着存在着三个比较重要的方法，即 equals( ),hashCode( ),toString( )三个方 法。

1）public String toString( )：在直接输出某个对象时，会默认调用Object类中的toString( ) 方法为对象进行字符编号处理，但如果在对象所在的类中覆写了此方法，则在输出对 象时会默认调用类中覆写的方法，已完成一些特殊的功能，例如取得对象的全部属性。 （toString( )方法也可以手工调用）

2）public boolean equals(Object obj)：equals( )方法主要用于对象的比较，但Object类的 equals( )方法是进行地址的比较。如果一个类需要进行对象的比较操作，则在类中直 接覆写此方法即可。例如String类中的equals( )方法就是覆写Object类中的方法。

3）public int hashCode( )：此方法在区别集合中对象是否重复时使用。

**使用Object类接收任意引用数据类型：**

Object类是所有类的父类，则也是所有对象的父类，所以所有的对象都可以向Object 类进行转换，当然这不仅包括类，还包括了数组和接口。即引用数据类型都可以向Objec t类对象进行转换。

因为这种特性，所有在很多的类库中方法的设计上都采用了Object作为方法的参数。

**包装类：**

1）包装类是基本数据类型作为类的一种表现，通过包装类，基本数据类型就可以作为 引用数据类型。

2）在Java中每一种基本数据类型都有其对应的包装类，除了Integer类与Character类 外，其他包装类与其基本数据类型区别不大，只是首字母大写了。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 基本数据类型 | 包装类 | 序号 | 基本数据类型 | 包装类 |
| 1 | int | Integer | 5 | float | Float |
| 2 | char | Character | 6 | double | Double |
| 3 | short | Short | 7 | boolean | Boolean |
| 4 | long | Long | 8 | byte | Byte |

其中，数值型包装类都是Number类的子类，Number类是一个抽象类，里面的抽 象方法的主要功能是将包装类变为基本数据类型。

而对于Character类和Boolean类，它们是Object类的直接子类。

**装箱和拆箱：**

将基本数据类型变为包装类对象称为装箱，将包装类对象变为基本数据类型称为拆箱。

对于装箱和拆箱，Java已支持自动装箱和拆箱。

**包装类的应用：**

包装类在实际应用中用的最多的是将字符串变为基本数据类型的操作上。例如，一个 纯数字组成的字符串可以转变为整型数据，或者一个仅包含布尔值的字符串转变为布尔 值，再或者一个包含小数点和数字组成的字符串可以转为浮点型数据。

**匿名内部类：**

匿名内部类是在抽象类和接口的应用上发展起来的。利用匿名内部类可以为抽象类或接 口对象进行实例化的，但需要带有一个方法体，用于覆写其中的全部抽象方法。其语法 格式为：

**接口 接口对象=new 接口名或抽象类名{覆写接口或抽象类中的全部抽象方法}；**

这样一来，接口对象就可以调用接口中的抽象方法了，但抽象方法已被对象的匿名内部 类覆写了，所以会调用覆写后的方法。同时该对象也可以访问类中的其他成员。