**继承的实现与规则：**

1. 使用extends关键字可以实现类的继承
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. 子类继承父类后，在子类中是不能直接访问父类中的私有成员的，但是子类可以调用父类中的非私有方法间接地访问私有成员

**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之后再调用自己的构造方法完成实例化

**方法的覆写：**

1. 方法覆写是指子类中定义了与父类中同名的方法，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

private < default < public

1. 方法覆写后，子类对象调用的将是被覆写后的方法。如果此时要在子类中访问父类中的方法，可以通过“super.方法名( )”的形式即可。
2. 访问权限由private变为其他访问控制权限不算是方法的覆写，因为private声明的方法无法被覆写。

**方法重载与覆写的区别：**

|  |  |  |
| --- | --- | --- |
| 区别点 | 方法重载 | 方法覆写 |
| 定义 | 方法名称相同，参数的类型或个数的不同 | 方法名称、返回值、参数类型个数全部相同，被覆写的方法不能拥有更严格的权限 |
| 范围 | 发生在一个类中 | 发生在继承关系中 |

**属性的覆写：**

子类如果和父类声明了相同名称的属性，则在子类中直接访问的时候肯定是采用就近访问原则，即先找到本类中的属性，如果此时要调用父类中的属性，直接使用“super.属性”格式即可。

**super关键字的作用：**

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，语句必须放在子类构造方法中的第一行，且不能与this调用构造方法同时出现，因为this调用构造方法时也必须放在构造方法中的第一行。

**this与super关键字的区别：**

|  |  |  |
| --- | --- | --- |
| 区别 | this | super |
| 属性或方法访问 | 访问本类中的属性或方法，如果本类没有则从父类查找 | 访问父类中的属性或方法 |
| 调用构造 | 调用本类中的构造方法，必须放在本类构造方法的第一行，且必须留有一个出口。 | 调用父类构造，必须放在子类构造方法的第一行 |
| 特殊 | 表示当前对象 | 无 |

**final关键字:**

1. 使用final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声明的变量将变成常量，常量不可以被修改。在Java中常量变量名要大写。
4. 使用public static final声明的变量称为全局常量。

**抽象类的基本概念与定义：**

抽象类是一种专门用来充当父类的类，作用类似于模板。抽象类不能实例化对象。使用abstract关键字可以定义抽象类和抽象方法。

**抽象类的使用规则：**

1）抽象类中不一定有抽象方法，但有抽象方法的类一定是抽象类。

3）抽象类必须被子类继承，抽象方法必须被子类覆写，所以抽象类和抽象方

法不能使用final关键字声明。

1. 抽象方法只需声明不需要实现。

**接口的基本概念与定义：**

接口可以理解为一种特殊的类。接口是由全局常量和公共的抽象方法所组成。接口不能实例化对象。

使用interface关键字定义接口。

接口的定义已明确声明了接口的组成，所以接口中的全局常量和抽象方法可以简写。

**接口的实现规则：**

1. 接口必须通过子类才能使用，子类可以通过implements关键字实现接口，同时允许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. 如果一个子类既要继承一个父类的同时又要实现接口的话，必须按照以下格式定义：

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

}

即一定要先继承后实现。

4）抽象类也可以实现一个或多个接口。

3）Java中只允许接口继承接口，接口可以通过extends继承接口，同时允许一个接

口可以继承多个接口。

**对象多态性的基本概念：**

对象的多态性分为向上转型和向下转型：

1. 向上转型：子类转为父类

**父类 父类实例=子类实例；**

1. 向下转型：父类转为子类

**子类 子类实例=（子类）父类实例；**

对于向上转型，程序会自动完成，而对于向下转型必须明确的指明要转型的子类类型。

**对象多态性的使用规则：**

1. 在进行对象的向下转型之前，必须发生对象的向上转型。只有发生了对象的向上转

型，父类才知道谁是它的子类，才可以发生对象的向下转型。

2）对象多态性使用的条件是存在继承关系和方法覆写。

**intstanceof关键字**：

1. 在Java中可以使用instanceof关键字一个是否是某个类的实例，结果返回boolean值。

**对象 instanceof 类名**

1. 父类对象通过子类实例化后，该父类对象同时是子类和父类的实例

**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和接口进行实例化。

**抽象类的实际应用（模板设计）：**

抽象类的作用类似于模板。例如定义一个学生类，拥有姓名、年龄、三个属性，定义一个工人类，拥有姓名、年龄、工资三个属性。但此时这两个类有共性。这时我们可以将这两个类的共性拿出来单独定义一个抽象类，学生类和工人类继承此抽象类，此时抽象类就相当于一个模板。

**接口的实际应用（制定标准）：**

接口是Java解决多继承局限的一种手段，但接口在实际中更多的作用是用来制定标准的。用接口制定标准后，实现该接口就相当于拥有了该标准。

**工厂设计：**

当一个接口存在多个子类时，若想根据需要选择想要的不同的子类来实例化接口调用不同子类中的方法时，就可以在接口对象实例化时加入一个过渡端，通过此过渡端来取得通过不同子类的实例化的接口对象并返回赋给主方法中的接口对象，达到以不同子类实例化接口对象并调用不同子类中的方法的目的。一般把这个过渡端称为工厂类。

**代理设计：**

代理设计是指一个代理主题来操作真实主题，代理设计所涉及到的角色如下：

·抽象角色：真实主题与代理主题的共同接口，可以是抽象类，也可以是接口。

·代理类：代理类内部含有真实类对象的引用，从而可以操纵真实类对象。

·真实类：代理类所代表的真实对象。

接口或抽象类

实现

实现

通过真实类对象调用

真实类

代理类

**适配器设计：**

对于Java程序来说，如果一个实现了一个接口，则必须覆写此接口中的全部抽象方法，如果此时一个接口中定义的抽象方法过多，但在子类中又用不到这么多抽象方法的话，则使用起来很麻烦，所以此时就需要一个中间的过渡，但是这个过渡类又不希望被直接使用，所以将此过渡类定义成抽象类最合适，即一个接口首先被一个抽象类实现（此抽象类通常被称为适配器类），并在此抽象类中实现若干方法（方法体为空），则以后的子类直接继承此抽象类，就可以有选择的覆写所需要的方法。适配器所解决的问题是子类不想覆写接口中的全部抽象方法。

**内部类的扩展：**

在一个抽象类中可以定义多个抽象类或接口；同时在一个接口中也可以定义多个接口或抽象类。

**抽象类和接口之间的关系：**

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 区别点 | 抽象类 | 接口 |
| 1 | 定义 | 包含一个抽象方法的类 | 公共抽象方法和全局常量所组成 |
| 2 | 组成 | 构造方法、抽象方法、普通方法、常量、变量 | 全局常量、公共抽象方法 |
| 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，只允许继承多个接口 |
| 5 | 常见设计模式 | 模板设计 | 工厂设计、代理设计 |
| 6 | 对象的产出 | 都通过对象的多态性产生实例化对象 | |
| 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 | 模板设计 | 制定标准 |
| 9 | 选择 | 如果抽象类和接口都可以使用的话，优先使用接口，避免单继承的局限 | |
| 10 | 特殊 | 一个抽象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：**

接口是由全局常量和公共的抽象方法所组成，但从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。使用default声明的普通方法需要用接口对象来调用，使用static声明的静态方法可以由接口名称直接调用。

**宠物商店实例：**

制定宠物标准，编写宠物类，编写宠物商店，编写测试类。

**Object类的基本概念：**

Object类是java中最为常用的一个系统类。利用Object类可以实现参数的统一。Object类是所有类的父类，只要一个类没有明显的继承一个类，则肯定是Object类的子类。

**Object类中的主要方法：**

1. public String toString( )方法：在输出某个类的对象时会默认调用Object类中的toString( )方法为对象进行字符处理。如果该类覆写了Object类中的toString( )方法，那么在直接输出此类对象时会默认调用此类中的toString( )方法，以完成一些特定的功能。
2. public boolean equals( )方法：equals( )方法主要的功能是用于对象的比较。因为String类也是Object类的子类。所以String类中的equals( )方法是覆写Object中的

**Object接收任意引用对象：**

Object类是所有对象的父类。所有的对象都可以向Object类进行转换，同时也包含了数组和接口类型，即一切引用数据类型都可以使用Object类进行接收。

**包装类的基本概念：**

包装类是表示基本数据类型的类，是基本数据类型作为类的一种表现形式。通过包装类可以将基本数据类型变为引用数据类型。每一种基本数据类型都有其对应的包装类。其中数值型基本数据类型所对应的包装类是Number类的子类，字符型与布尔型所对应的包装类是Object的直接子类。

**自动装箱与拆箱操作：**

装箱是指把基本数据类型变为包装类对象；拆箱是指把包装类对象变为基本数据类

型。

**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，但需要注意的是变为基本数据类型的字符串必须是由数字组成的。

**匿名内部类：**

匿名内部类是没有类名称的内部类,是针对抽象类和接口不能实例化对象而来的。在直接实例化接口或抽象类之后，因为接口和抽象类不能实例化对象，所以要在实例化后带有一个方法体用于实现接口或抽象类中的抽象方法。然后通过匿名类实例化抽象类或接口对象，此时该对象可以调用匿名类方法体中实现的方法。格式如下：

new 接口或抽象类（）{ 要实现的方法 };

**类的继承原则：**

在类的设计中，永远不要去继承一个已经实现好的类，只建议继承抽象类或实现接口，如果抽象类和接口都可以的话，那么优先使用接口，避免单继承的局限。