# Day08

## 类的继承

### 为什么要有继承

|  |
| --- |
| 继承是java面向对象的三大特征(继承、封装、多态)之一，是软件复用的重要手段之一。  举例：Person类具有age和name属性，Student类也具有age和name属性外，还具有sno属性。他们有共同的属性age和name，这种情况下如果把相同的属性在student中重写一遍就是一种代码重复。这种情况下可以考虑继承。  public class Person  {  public int age;  public String name;  }  public class Student  {  public int age;  public String name;  public int sno;  } |

### 继承的概念

|  |
| --- |
| 一个类继承另一个类。实现继承的类称为子类(subclass)也叫派生类,而被继承的类称为父类,也叫**超类或基类。**  通过“继承”一个现有的类，可以在子类中使用被继承的类(父类)中的属性和方法。  **类的方法和属性可以被继承，但是类的构造器(构造方法)不能被继承**  **假如两个类有相同的属性和方法，那么就可以创建一个具有他们共同属性和方法的类作为父类，让子类来实现该父类，根据实际业务需求来添加新的属性和方法。** |

### Java中继承的语法

|  |
| --- |
| Java中继承使用关键字**extends**  [权限修饰符] class <类名> extends <父类名>  {  //类中代码  }  在java中一个类只能有一个直接父类，即：java只支持单继承(**亲爹只能有一个**。干爹可以有多个，学习接口时会学到)，不支持多继承。（与C++不同） |

### Java中继承的特点

|  |
| --- |
| 1. 子类可以访问父类中**非私有的成员变量和方法**。 2. java只支持单继承不支持多继承：一个类最多只能有一个直接父类   --- public class SubClass extends SuperClass{ //OK  }  --- public class SubClass extends SuperClass1, SuperClass2{ //**错误，不能继承多个父类**  }   1. java支持**多层**继承(继承体系)。B继承A， C可以再继承B   例如：  public class A{  }  public class B extends A{  }  public class C extends B{  }  4、一个儿子只能有一个亲爹。一个类可以有多个子类。 |

## 继承中成员变量的特点

|  |
| --- |
| 子类可以继承父类中的非私有变量。  使用this可以调用本类成员变量  使用super可以调用父类的变量 |

## 继承中方法的特点

子类可以继承父类中的非私有方法。

使用this可以调用本类方法

使用super可以调用父类的方法和属性

### 为什么子类要对父类的方法重写

|  |
| --- |
| 由于父类中的方法功能过于简单，或者不适用于子类，这种情况需要对父类方法在子类中重写。  例如：Animal中speak方法，输出为“我是一个动物”, Cat是Animal的子类，这时如果直接调用父类的方法就有点不太合适，就需要在子类中覆写speak方法 |

### 方法覆写规则

|  |
| --- |
| 方法覆写遵循规则：**两同两小一大**  **两同： 方法名相同、参数列表相同**  **两小：被覆写的方法在子类中的返回值类型的范围比父类小或者与父类相同。**  **被覆盖的方法在子类中抛出的异常范围比父类小或者与父类相同(异常后续会讲)**  **一大：被覆盖的方法在子类中访问控制权限比父类的大或者相等。** |

## 继承中构造方法的特点

|  |
| --- |
| 由于子类和父类的类名(包括包名)肯定不同，而构造方法的名字必须和类名一致，所以构造方法不能继承。  创建子类对象时，子类会继承父类的一些元素，所以子类会先访问父类的构造方法。 |

## 方法覆写和重载的对比

|  |
| --- |
|  |

## super关键字

使用super可以调用父类的成分：父类的成员变量、父类的方法、父类的构造方法。

**注意：使用super关键字和this关键字，均不能用在静态方法中，只能用在普通方法(实例方法)中。**

### 利用super调用父类的成员变量

|  |
| --- |
| 语法格式：**super.**父类变量名 |

### 利用super调用父类的方法

|  |
| --- |
| 语法格式：**super.**父类方法 |

### 利用super调用父类构造方法

#### 2.3.1 默认情况下构造方法的调用

|  |
| --- |
| 子类实例化时，必须首先调用父类的构造方法，然后才是子类的构造方法。**默认情况调用的为父类的无参构造方法。**  直接使用new ChildClass()测试  运行结果： |

#### 2.3.2 调用父类的特定构造方法

|  |
| --- |
| 有时，需要在子类的构造方法中来调用父类的某个特定的构造方法。语法：  **super(实参)**  **注意:1、super()调用父类构造方法时，必须是子类该构造方法的第一行代码。**  **2、super后面没有 .**  **3、实参可以为空，当实参为空时表示调用的是父类的无参构造方法，此时与省略super()效果相同.** |

## this和super的区别

**两者均不能在static修饰的方法中使用。**

在用在构造方法中的时候，两者均需要放在构造方法的第一行，所以二者不可同时使用。

|  |
| --- |
|  |

## 访问控制

|  |
| --- |
| Java中共四种权限修饰符：private、protected、default(什么都不写)、public |

## final关键字

|  |
| --- |
| 在有些情况下，我们不希望我们定义的某个类被子类继承，或者某个类中某个方法不希望在子类中被覆写，就需要用到final关键字  如果final修饰某个类，则该类变成”终类”，不能被任何类继承  如: **public final class Cat {**  **}** //**则该类不能被任何类继承**  **public class Cat{**  **public final void speak(){**  **} //则该方法不能在子类中被重写**  **}**  **另外：final也可以修饰变量。当final修饰变量时，则该变量只能被赋一次，则该变量就变成常量。常量的命名规则.** |