# 继承

描述类的时候，我们如果发现几个类有重复的属性和方法我们就可以采用继承的方式来设计。

|  |
| --- |
| /\*\*  为了减少代码的重复性抽象出来一个公共的类  \*/  class Teacher{    String name;    int age;    public void sleep(){  System.out.println("人在睡觉");  }  }  /\*\*  class 子类 extends 父类{    }  \*/  class JavaTeacher extends Teacher{    public void teachJava(){  System.out.println(name+"老师在讲java课");  }      }  class PHPTeacher extends Teacher{    public void teachPhp(){  System.out.println(name+"老师在讲PHP课");  }    }  public class TestTeacher1{    public static void main(String[] args){  JavaTeacher jt = new JavaTeacher();  jt.name = "亮哥";  jt.age = 18;  jt.teachJava();  jt.sleep();    PHPTeacher pt = new PHPTeacher();  pt.name = "龙哥";  pt.age = 28;  pt.teachPhp();  pt.sleep();  }  } |

1.如果父类中包含了某些类中的共同的属性和行为，我们可以使用继承来设计程序。

2.子类使用extends关键字继承父类的共同属性以外，子类还可以有自己特有的属性或者方法。

3.父类更通用，子类更具体。

4.子类只能获得父类中的非private的属性，如果想要继承就得提供公共的set和get的方法。 私有的方法 是无法继承下来的

5.java中类只能做单继承

# 继承的内存结构

|  |
| --- |
| class Person{    String name;    int age;    public void sleep(){  System.out.println(name+"在睡觉");  }  }  class Hero extends Person{    String vehicle;    public void kill(){  System.out.println(name+"上阵杀敌");    }      }  public class TestPerson{    public static void main(String[] args){  Hero h = new Hero();  h.name = "吕布";  h.age = 28;  h.vehicle = "赤兔";    h.sleep();  h.kill();      }  } |

0x98 new Hero()

栈 堆

Vehicle = “赤兔”;

Super()

Hero h

Name = “吕布”

Age = 28

0x98

数据共享区

Hero.class

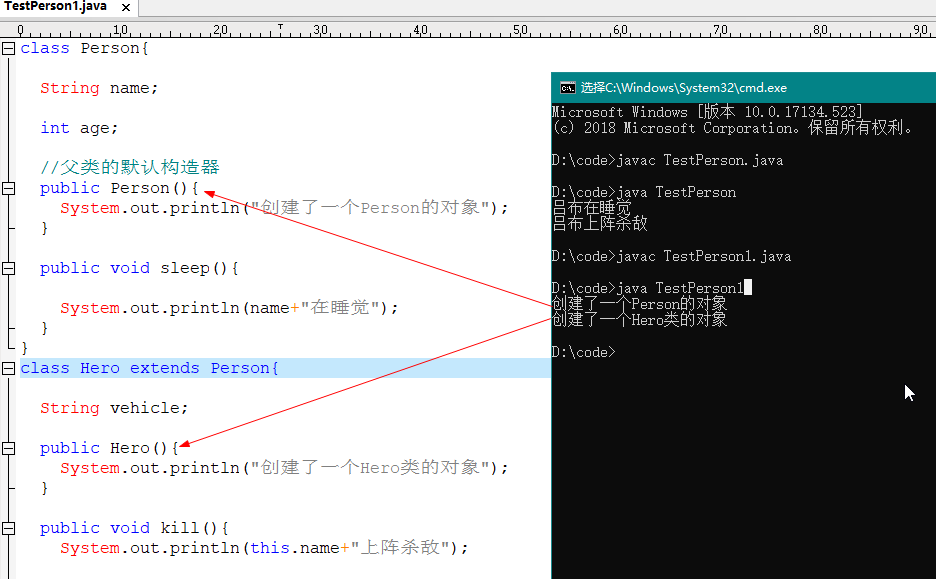
Kill(){ this.name }

Person.class

Sleep(){ this.name }

# super关键字

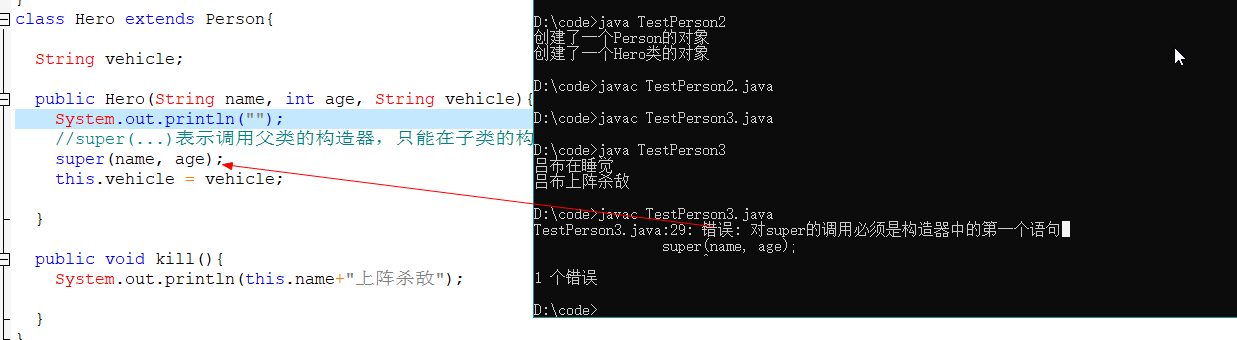
1. 子类实例化的过程中父类的构造器先被调用，然后再调用子类的构造器



|  |
| --- |
| class Person{    String name;    int age;    //父类的默认构造器  public Person(){  System.out.println("创建了一个Person的对象");  }    public void sleep(){    System.out.println(name+"在睡觉");  }  }  class Hero extends Person{    String vehicle;    public Hero(){  System.out.println("创建了一个Hero类的对象");  }    public void kill(){  System.out.println(this.name+"上阵杀敌");    }  }  public class TestPerson1{    public static void main(String[] args){    Hero h = new Hero();  h.name = "吕布";  h.age = 28;  h.vehicle = "赤兔";          }  } |

2.子类通过构造器实例化时要调用父类的默认构造器，是隐含的调用了super()这个方法。但是如果子类的 构造器中调用了父类中的有参数的构造器，那么无参数的构造器就不会被调用了。

3.super()的调用必须要放在方法的第一行。



4.super可以表示父类的引用，我们可以使用super和this来区分父类和子类中同名的属性。在子类中访问属 性的查找顺序首先在子类中查找指定的属性名字，如果在子类中没有该属性，那么就去父类中去查找该 属性。方法也是一样的当子类和父类有同样的方法时通过子类调用的这个相同的方法时调用的是子类的 方法。但是我们可以通过super对父类的引用来调用父类的方法。

0x98 new Hero()

this

栈 堆

Vehicle = null;

Name = “三姓家奴”;

super

Hero h

0x98

Name = “吕布”

Age = 0

数据共享区

Hero.class

Kill(){ this.name }

Person.class

Sleep(){ this.name }

# 方法的重写（override）

在子类继承父类时，子类的方法和父类的方法相同（访问修饰限制符， 返回值类型， 方法名， 参数列表），方法体不同。这种子类的方法将父类的方法覆盖叫做重写。

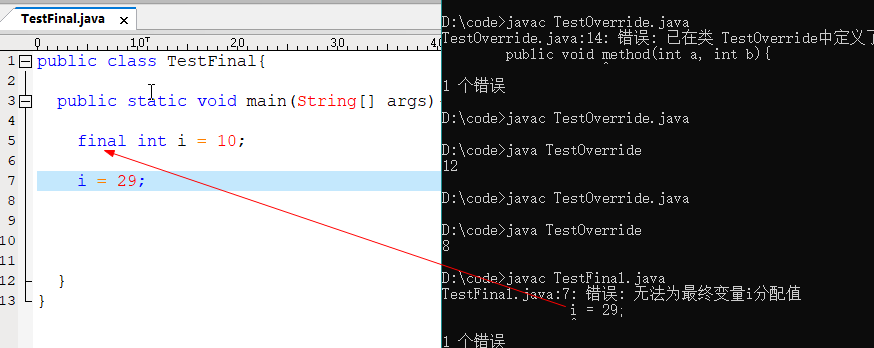
**重写和重载的区别：**

重写（override）：在父子类的继承中有相同的方法，唯一不同就是方法体，一般是父类的该方法满足不了子类的需求所以才发生重写。

重载（overload）：重载是在同一个类中，有着相同的方法名但是参数的数据类型或者参数的个数不同这两个方法就是重载。重载的目的：节省类中的命名资源和提高代码的可读性。

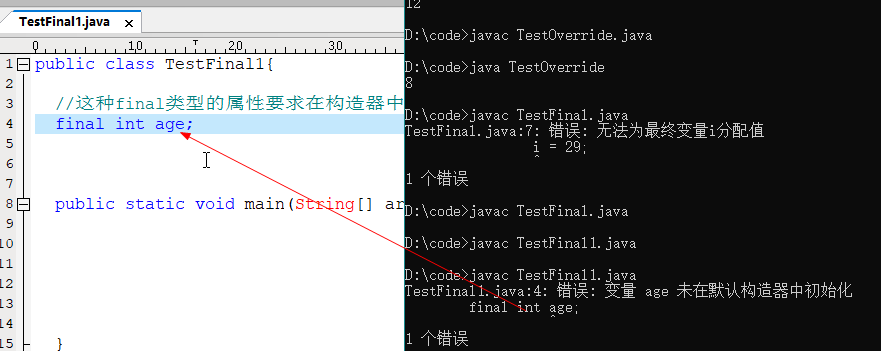
# final关键字

## final修饰在局部变量上



Final修饰局部变量的话，这个变量不能修改，就变成了常量。

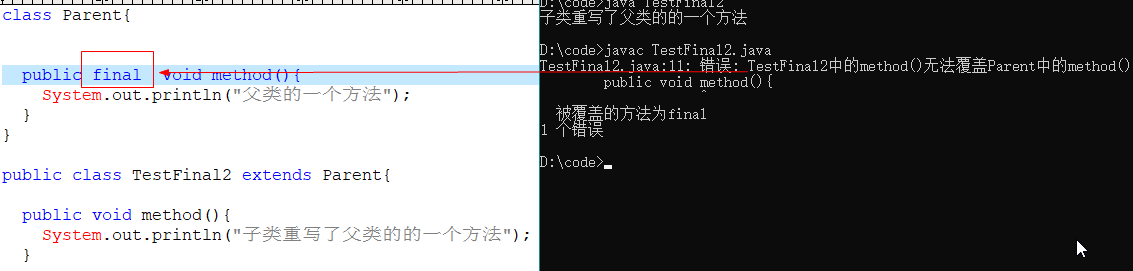
## final修饰在属性上



属性都有默认值，但是final修饰的话就没有默认值，要求要在构造器中给值， 一旦这个值赋值完毕就不能修改了，如果不在构造器中赋值，就直接给一个默认值

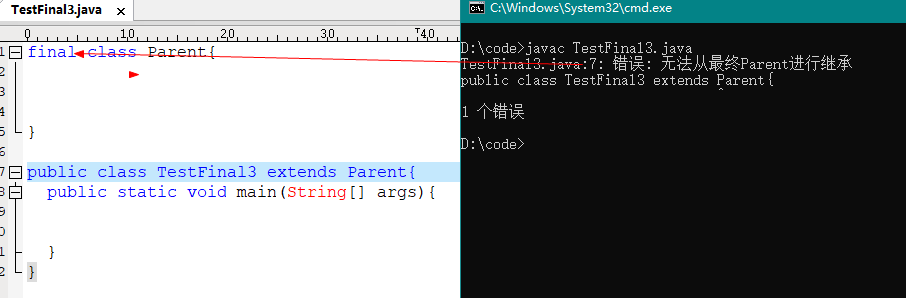
|  |
| --- |
| public class TestFinal1{    //这种final类型的属性要求在构造器中初始化  final int age = 10;      public TestFinal1(){  //age = 10;  }    public static void main(String[] args){    TestFinal1 tf = new TestFinal1();  //tf.age = 11;  }  } |

## final修饰在方法上



如果final修饰在方法上的话，不可以被子类重写。

## final修饰在类上



Final修饰在类上，不可以被继承。

# 抽象类

当多个具体的实体类存在着共同的行为，但是有不同的表现，我们在父类继承过程中父类的方法具体实现不同方式，但是能确定的是他们都有这种行为。

我们要把这种行为方法作为**抽象**方法。

Teacher

String name

Int age

Public void sleep(){

}

JavaTeacher PhpTeacher

Public void teachPhp(){

}

Public void teachJava(){

}

Teacher

Teacher

String name

Int age

Public void sleep(){

}

Public void teach();

JavaTeacher PhpTeacher

Public void teach (){

//教php

}

Public void teach (){

//教java

}

public abtract [返回值][void] 方法名 (参数列表); 没有方法体。如果在一个类中存在一个方法，那么这个类一定是抽象类。这个类上也要有abstract标识，

抽象类的语法：

abstract class 类名{

属性

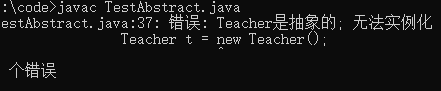
方法

}

注意：抽象类中可以没有抽象方法，但是有抽象方法的类一定是抽象类

## 抽象类的特点

1. 抽象类不能被实例化。



1. 抽象类必须被子类继承才能使用子类实例化。
2. 继承了抽象类的非抽象类必须实现抽象类的所有抽象方法
3. 抽象类也可以继承抽象类，这样不需要实现父抽象类的方法。
4. 抽象类的抽象方法不能和private，final， static共存。