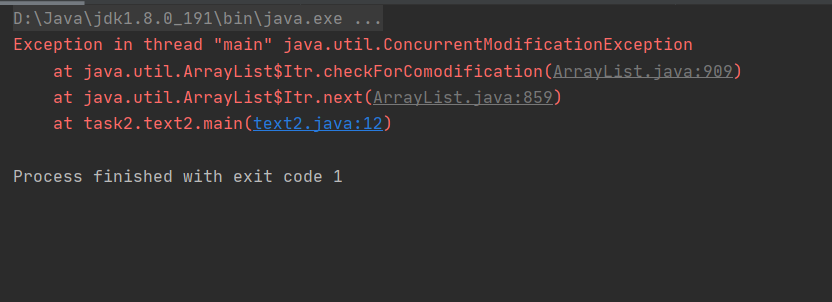
**集合元素删除**

**普通FOR循环删除集合元素会报错**

**}**

**ArrayList<Integer> m = new ArrayList<>();  
Collections.*addAll*(m,1,2,3,4,5,6,7,8,9,0);  
for(int i : m){  
 m.remove(i);  
}**

****

**删除集合元素应该用迭代器删除**

**}**

**Iterator <Integer> iter = m.iterator();  
while (iter.hasNext()){  
 Integer next = iter.next();  
 iter.remove();  
}**

**IDEA jar包导入 ：**

**https://blog.csdn.net/plpldog/article/details/82794541**

**正则表达式**

[**https://www.runoob.com/java/java-regular-expressions.html**](https://www.runoob.com/java/java-regular-expressions.html)

[**https://www.cnblogs.com/hd92/p/11322415.html**](https://www.cnblogs.com/hd92/p/11322415.html)

**IDEA导入项目**

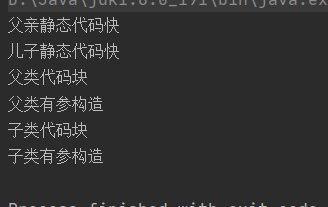
[**https://www.cnblogs.com/qiyebao/p/12189519.html**](https://www.cnblogs.com/qiyebao/p/12189519.html)

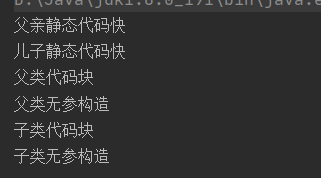
**2.在项目中为什么要使用接口以及好处：解耦合(减少层与层之间的依赖关系)**

**面试中的接口的问题对大家是如何提问的。  
在项目中为什么要使用接口，使用接口的好处是什么？  
1.先要将接口的概念说清楚  
 接口是一种约定(规范)：接口中只能有常量以及抽象方法  
 接口可以实现多继承 A extends B,C  
 接口也可以多实现 AClass impl A,B,C  
 接口也是Java中多态的表现形式(因为接口也相当于父类型)  
  
2.在项目中为什么要使用接口以及好处：解耦合(减少层与层之间的依赖关系)**

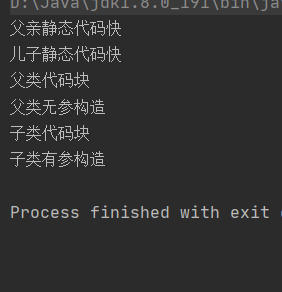
**JAVA 父类子类执行顺序**

* **new一个类对象，类中各部分执行顺序：静态代码块—非静态代码块—构造函数—一般方法。**
* **子类继承父类各部分执行顺序为：父静态代码块--子静态代码块--父非静态代码--父无参构造函数--子静态代码块--子构造函数--方法。**
* **调用子类构造方法时默认先调用父类构造方法，如果子类父类无参构造，有参构造都有的话，调用子类无参构造先调用父类无参构造，调用子类有参构造先调用父类有参构造**

****

****

* **如果子类有有参构造，无参构造，父类只有无参构造，调用子类有参构造时先调用父类无参构造，或者子类有参构造没有super(name) 父类，也是调用父类无参构造**

****

**当创建子类对象时，父类对象先进内存。这样子类才能自动拥有父类的成员。**

* **Super 在构造方法里类似一个指针，指向父类的构造方法，无参构造中默认写super(),**

**所以调用子类时先调用父类构造方法事实上是从super()调用的**

* **这个是Java的规定：子类继承父类，子类的构造方法必须调用super（）即父类的构造方法，而且必须放在构造方法的第一行。**

**那么问题又来，为什么我写的类有时候都不用去调用父类的构造方法？**

**因为：如果父类"只"有无参构造方法，且不打算重写子类的构造方法，为节省代码量，子类构造方法可以不写，但是实际上是已经写了，系统默认调用父类无参构造方法super()，即默认会在子类的构造方法中的第一行加上父类的无参构造方法**

**下面是我自己的研制代码截图：当父类有无参构造函数或者，没有生命构造函数的时候，是不会报错**

**最好再次总结：子类构造函数必须在构造函数第一行调用父类构造函数，**

**当子类构造函数没有调用父类构造函数的时候是因为父类构造包含无参构造函数，**

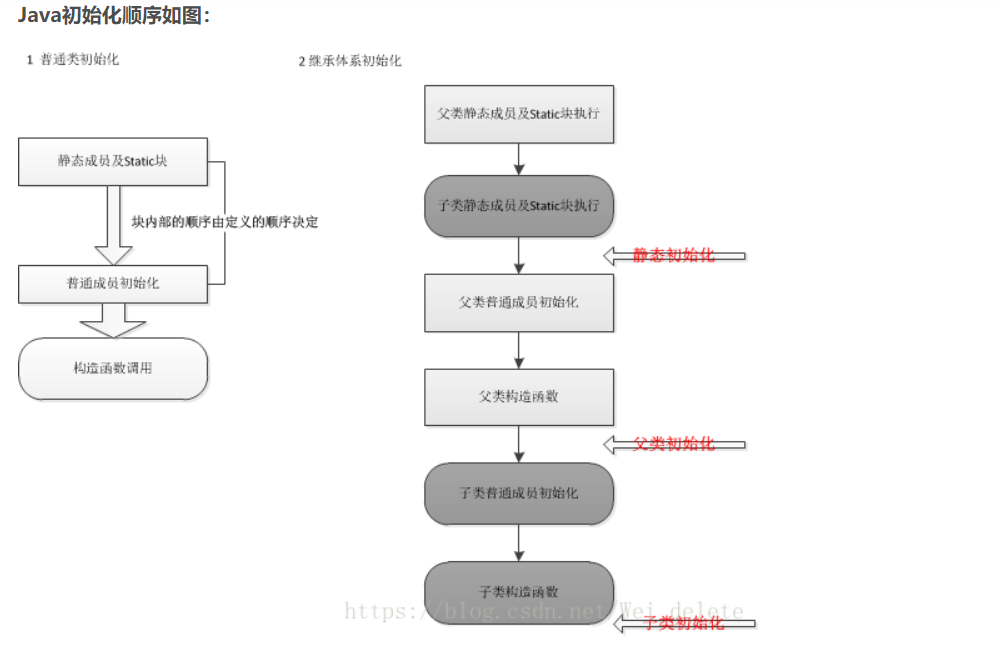
**为了节省代码，系统会自动添加调用**

* **注意：创建子类对象调用子类的构造方法的时候会先调用父类的构造方法，在子类的构造方法中调用父类的构造方法是用super()，如果没有写super()，则默认调用父类的无参构造方法。**
* **子类继承父类，调用方法时先是调用子类中的方法，如果没有就调用父类中的方法，还有一点就是try{ }、catch{ }、finally{ }返回值的问题，一旦try{ }中返回了某一个值，如果finally有返回值，finally中的返回值会覆盖try的返回值，如果finally没有返回值，就是try中的返回值**

**构造器**

**子类是不继承父类的构造器（构造方法或者构造函数）的，它只是调用（隐式或显式）。如果父类的构造器带有参数，则必须在子类的构造器中显式地通过 super 关键字调用父类的构造器并配以适当的参数列表。**

**如果父类构造器没有参数，则在子类的构造器中不需要使用 super 关键字调用父类构造器，系统会自动调用父类的无参构造器。**

****

/\*\*  
\* 接口(interface)：接口主要用于描述现实世界中拥有同种行为的事物。比如：飞机 ufo可能都会飞，但是是完全不同的类。  
\* 可以将接口理解为是一种特殊的抽象类(没有属性只有抽象方法)实际上接口的官方文档的定义中它不是一个类.  
\*  
\* 接口的特征：它是一个常量和抽象方法的集合  
\* 1.关键字是interface,不是calss  
\* 2.接口中的方法全部都是public的公有的抽象(abstract)方法,可以省略public和abstract关键字。  
\* 3.有接口就要有实现类，实现类来负责实现接口中所有的抽象方法  
\* 4.可以使用implements来实现接口  
\* 5.接口是不能被实例化的(不能直接创建对象)  
\* 6.接口之间可以互相继承，且是多继承的(extends) public interface C extends A,B  
\* 7.类可以实现接口，且可以多重实现(implements) public class BImpl implements A,B  
\* 8.一个类可以继承另一个类的同时去实现一个或者是多个接口  
\*  
\* 笔试题：  
\* 接口与抽象类的区别  
\* 1.抽象类是一个class,接口是interface，这是一个概念  
\* 2.抽象类中可以有抽象方法，也可以有普通方法；接口中是抽象方法与常量的一个集合  
\* 3.抽象类中可以有属性；而接口中只能是常量；  
\* 4.抽象类只能被单继承，而接口可以实现多继承

重写：

<https://www.runoob.com/java/java-override-overload.html>

父类引用指向子类对象

<https://www.jianshu.com/p/8ee1834f2071>

多态

<https://www.runoob.com/java/java-polymorphism.html>

成员变量，局部变量

1、局bai部变量：在方法内定义的变量du称为“局部变量”或“临时zhi变量”，方法结dao束后局部变量占用的内存将被释放。

2、成员变量：在类体的变量部分中定义的变量，也称为字段。

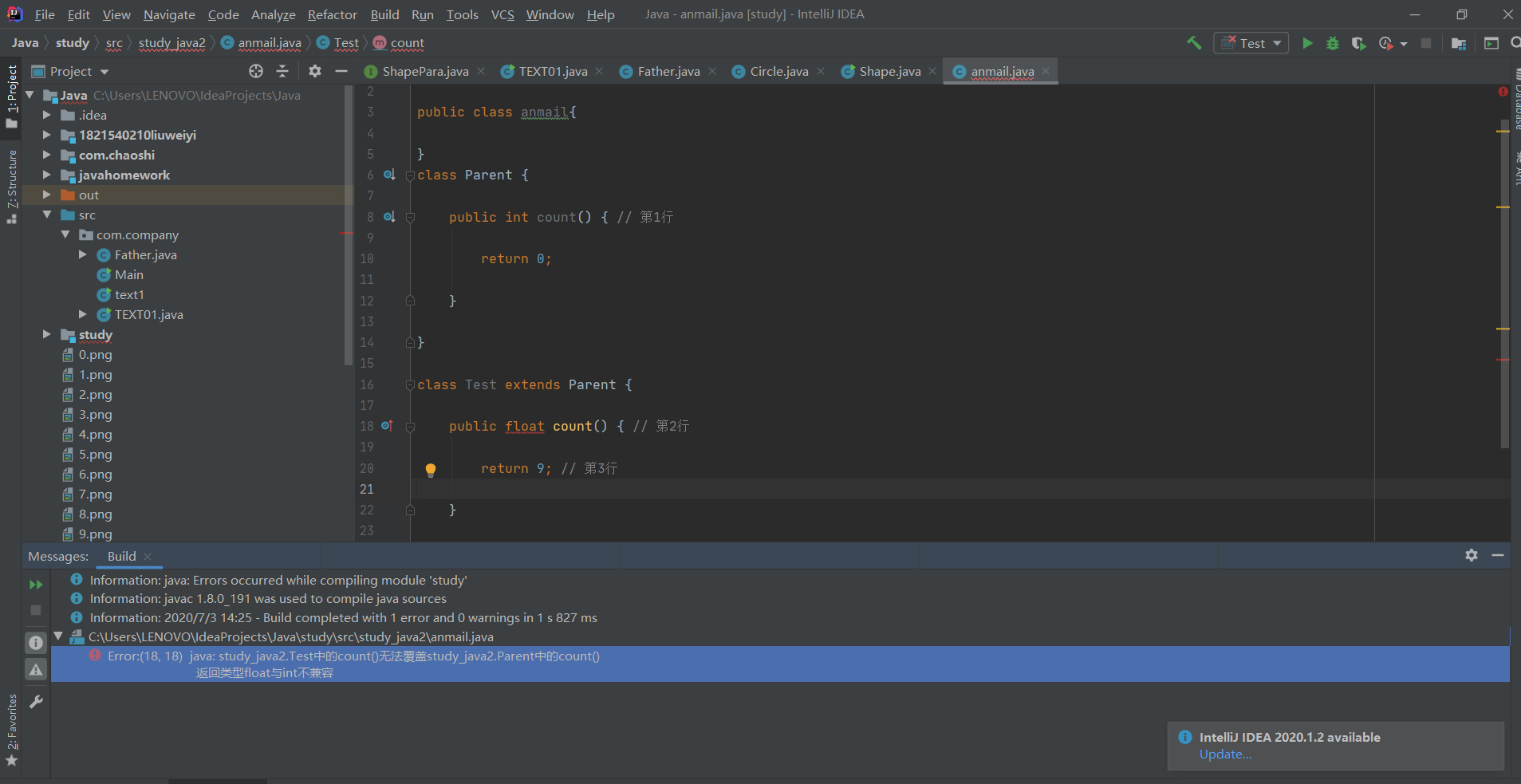
3、全局变量：全局变量，又称“外部变量”，它不是属于哪个方法，作用域从定义的地址开始到源文件结束。

4、局部变量和成员变量主要是他们作用域的区别：

成员变量个是类内部；局部变量是定义其的方法体内部（或者方法体内部的某一程序块内——大括号，主要看定义的位置）。

成员变量可以不显式初始化，它们可以由系统设定默认值；局部变量没有默认值，所以必须设定初始赋值。

在内存中的位置也不一样。成员变量在所在类被实例化后，存在堆内存中；局部变量在所在方法调用时，存在栈内存空间中。



重写的方法，外壳必须相同（名称，返回值类型）

4.对于多态总结一下

一、使用父类类型的引用指向子类的对象；（就是一个纯父类）  
  
    二、该引用只能调用父类中定义的方法和变量；（调用子类有父类没有的会报错）  
  
    三、如果子类中重写了父类中的一个方法，那么在调用这个方法的时候，将会调用子类中       的这个方法；（动态连接、动态调用），变量则相反，变量不会被覆盖，变量的值看等号左边，是谁就是多少。  
  
 四、变量不能被重写（覆盖），”重写“的概念只针对方法，如果在子类中”重写“了父       类中的变量，那么在编译时会报错。

属性没有多态的概念，在 Java 中只有普通实例方法才可以实现多态。   
非private(invokespecial)、static(invokestatic)修饰的方法可以实现运行时的多态。

class A {  
 public int i = 10;  
}  
class B extends A{  
 public int i = 20;  
}  
public class test02 {  
 public static void main(String args[]) {  
 B b = new B();  
 A a = new B();  
 System.*out*.println(b.i); //20  
 System.*out*.println(a.i); //10 成员变量不会被多态  
 }  
  
}

**在现实生活中的体现:**

      自己用自己的，儿子有父亲一样的，就用儿子的。省点钱是点。儿子独有的，父亲不会用，所以不能用

# 接口：

接口中这样的定义是可以通过的

interface interf {

    public int a = 100;

    public void p();

}

但是变量会自动转换为public final static类型

就是说是不可以改变实现了该接口的类中的变量a的

(1)**float** 型 内存分配4个字节,占32位,范围从10^-38到10^38 和 -10^38到-10^-38  
例float x=123.456f,y=2e20f; 注意float型定义的数据末尾必须 有"**f** "或"F",为了和double区别

(2)**double** 型 内存分配8个字节,范围从10^-308到10^308 和 -10^-308到-10^-308  
例double x=1234567.98,y=8980.09d; 末尾可以有"d"也可以不写

JAVA内存

<https://blog.csdn.net/qq_32127759/article/details/80725115>

<https://blog.csdn.net/qq_28009065/article/details/79087831>

**JAVA锁**

<https://blog.csdn.net/u011068702/article/details/80646679?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-1.nonecase>

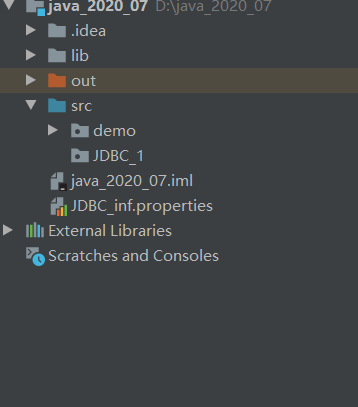
1、synchronized加在非静态方法前和synchronized(this)都是锁住了这个类的对象，如果多线程访问，对象不同，就锁不住，对象固定是一个，就可锁住。

2、synchronized(类名.class)和加在静态方法前，是锁住了代码块，不管多线程访问的时候对象是不是同一个，能缩小代码段的范围就尽量缩小，能在代码段上加同步就不要再整个方法上加同步，缩小锁的粒度。 好用

# 使用JAVA读写Properties属性文件

<https://blog.51cto.com/fluagen/409262>

配置位置



存储方式：key = value

Password = root

Class = com.mysql.jdbc.Driver  
Connection = jdbc:mysql://127.0.0.1:3306/lol?characterEncoding=UTF-8  
User = root  
Password = root

Properties prop = **new** Properties();// 属性集合对象

FileInputStream fis = **new** FileInputStream("prop.properties");// 属性文件输入流

prop.load(fis);// 将属性文件流装载到Properties对象中

 fis.close();// 关闭流

// 获取属性值，sitename已在文件中定义

        System.out.println("获取属性值：sitename=" + prop.getProperty("sitename"));

 prop.getProperty() 为根据指定key 获取指定value

IDEA上传GIT

<https://blog.csdn.net/qq_22194659/article/details/82113560>

抽象类除了不能实例化对象之外，类的其它功能依然存在，成员变量、成员方法和构造方法的访问方式和普通类一样。

#### 以举例的方式说到如何区分抽象类和接口，这里我们从Java语法谈起，使我们更加了解这两者的内在区别。它们的语法区别：

#### 1）接口不能有构造方法，抽象类可以有。

#### 2）接口不能有方法体，抽象类可以有。

#### 3）接口不能有静态方法，抽象类可以有。

#### 4）在接口中凡是变量必须是public static final，而在抽象类中没有要求。