1.**switch**中常见的注意事项

1> swich 中的case的连续

case 1：

case 2：

case 3：

System.out.println（);

2>除了已经有的case 则是default ：即可；

**switch** (c){

**case** 1:

**case** 2:

**case** 3:

System.*out*.println("");

**break**;

**default**://除了123剩下的所有

System.*out*.println("");

**break**;

}

2.return 的作用

1结束方法的运行

2.返回值

3.打包之后移动时我们会将原来的包内的java程序删除

4.使用递归非常重要（难懂篇章）

**public** **class** Textdigui {

**static** **int** *a* = 1;//方法外面的变量可以让面的方法是用

**public** **static** **void** main(String args[]){

*text*();

}

**public** **static** **void** text(){

System.*out*.println("我正在测试递归（就是相当于自己调用自己）"+ *a*);

*a*++;

**if**(*a*<3)

{

*text*();

}

}

}

注意事项：如果将a定义在方法的内部则是不可以使用递归的。而是

5.制作API

**package** 第一个试用包;

**import** java.util.Scanner;

/\*\*

\* 本程序是在测试制作API文件(计算a + b 的和) //解释说明本程序的功能

\* **@author** Lenovo

\* **@version** 1.8

\*/

**public** **class** ZhizuoAPI {

**public** **static** **void** main(String args[]){

**int** a,b,c;

Scanner s = **new** Scanner(System.*in*);

System.*out*.printf(" 请输入 a 的值：\t");

a = s.nextInt();

System.*out*.printf("\n");

System.*out*.printf(" 请输入 b 的值：\t");

b = s.nextInt();

c = *jiafa*(a,b);

System.*out*.printf(" a + b = ");

System.*out*.printf(" "+c);

}

/\*\*

\* 制作一个加法的方法（或者叫函数） //解释方法的作用

\* **@param** a 一个加数 //这个会自动弹出

\* **@param** b 一个被加数

\* **@return** a和b 的和 //解释返回值的作用

\*/

//上面的东西写在方发的前面

**public** **static** **int** jiafa(**int** a,**int** b){

**int** and;

and = a + b;

**return** and;

}

}

导出 - 》 指定导出的路径-》打开index。html即可

6.构造器

**public** **class** Point {

**double** x,y,z;

//通过构造器初始化对象的属性

**public** Point(**double** \_x,**double** \_y,**double** \_z){

x= \_y;

y = \_y;

z = \_z;

}

**public** **double** distance(Point a){

**double** result = (x-a.x)\*(x-a.x) + (y-a.y)\*(y-a.y) + (z-a.z)\*(z-a.z);

**return** result;

//x 代表的是初始化的x

}

**public** **static** **void** main(String args[]){

Point s = **new** Point(1,1,1); //初始化 x ,y z

Point s2 = **new** Point(2,2,2); //增加一个点

System.*out*.println(" 计算（1,1,1）和（2,2,2,）两个点的距离平方 ");

System.*out*.println();

System.*out*.println(" 距离平方 "+s.distance(s2));

//在 s 变量中调用 s2 使得距离有意义

}

}

重载构造器 通过this调用并且必须位于第一句

**public** **class** TextThis {

String name;

**int** age;

**public** TextThis(**int** a,**int** b){

**this**();//必须位于第一句否则将会是出错误；

或者是this（s）;这是有参数方法的调用

}

**public** TextThis(){

System.*out*.println("这时将要调用的构造器");

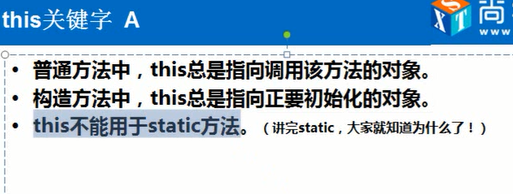
}

**public** TextThis(int a){ //含有参数的

System.*out*.println("这时将要调用的构造器");

}

7.this 隐式参数的使用 普通方法中可以使用this 在static中不可以使用，其实和形参一样的



1>

**public** **class** TextThis {

String name;

**int** age;

**public** **static** **void** main(){

} //不可以使用this

**public** **void** sui(){

**this**.age = 21 ;

}

}

8.继承

**public** **class** Animal {

String eye;

**public** **void** run(){

System.*out*.println("跑起来");

}

**public** **void** eat(){

System.*out*.println("吃起来");

}

}

**class** Mammal **extends** Animal{ //继承Animal 中的一切事物，下面的将会进行补充

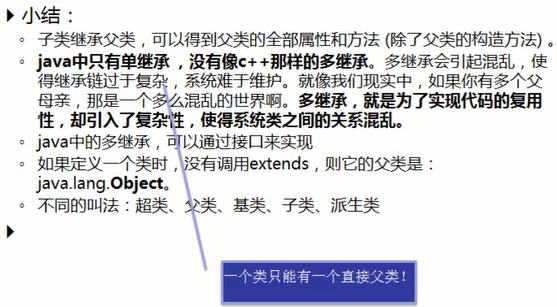
**public** **void** bird(){

System.*out*.println("飞起来");

**this**.eat(); //即可调用eat方法

}

}



注意 一个类只能有一个直接父类

9.接上1>继承中方法的重写

**public** **class** Animal {

String eye;

**public** **void** run(){

System.*out*.println("跑起来");

}

**public** **void** eat(){

System.*out*.println("吃起来");

}

}

**class** Mammal **extends** Animal{ //继承Animal 中的一切事物，下面的将会进行补充

**public** **void** bird(){

System.*out*.println("飞起来");

**this**.eat(); //即可调用eat方法

**public** **void** run(){ //run方法前面的Animal中有了已经了。此时我们时进行了覆盖

System.*out*.println("这是一个方法的重写");

}

}

}

}

2> 方法重写之后我们还可以调用，但是我们只能在该类中调用

**public** **void** run(){ //run方法前面的Animal中有了已经了。此时我们时进行了覆盖

super.run();

System.*out*.println("这是一个方法的重写");

}

10.**object**

1.Object类中有下面的一个方法

public String toString(){

return 类的名字和16进制的数字（地址变化而来）

}

例如

**public** **class** TextObject {

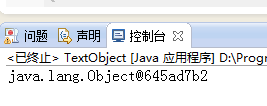
**public** **static** **void** main(String args[]){

Object abj = **new** Object();

System.*out*.println(abj.toString());

}

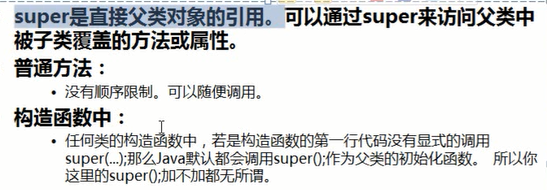
}



运行之后的是

11.小技巧 查看类型层次结构 鼠标移动到类的名字上面----》ctrl + t 就会出现；

12super



1》构造器中的super的使用

**public** **class** TextSuper {

**public** TextSuper(){

**super**(); // 第二步：调用的是Object构造器的内容

System.*out*.println("构造器");

}

**public** **void** run(){

System.*out*.println("跑起来");

}

}

**class** Paxing **extends** TextSuper{

**public** Paxing(){

**super**(); // 第一步：调用的是上面的父类TextSuper 的构造器的内容

System.*out*.println("爬行");

}

}

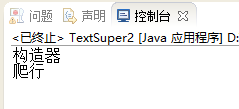
调用时候的程序及结果为

**public** **class** TextSuper2 {

**public** **static** **void** main(String args[]){

Paxing a = **new** Paxing();

}

} 

13.final

**public** **class** TextFinal {

**public** **static** **void** main(String args[]){

**final** **int** a = 4; //之后这个a将会是一个常量

a = 5; //这里就是错误，不可以对a进行指定了

}

}

**1》修饰常量**

**不可以进行更改常量的**

**大小**

**2》修饰方法（不可以进行重写了，但是可以重载）**

**public** **class** TextXiushifangfa {

**public** **final void** run(){

System.*out*.println("这里将不会被下面的子类重写因为有了fainal");

}

**public** **void** fei(){

}

}

**class** jicheng **extends** TextXiushifangfa {

**public** **void** run(){ //这里将会是一个错误的，因为上面的类下面的方法上final的 不可以重写

System.*out*.println("wosji");

}

}

3》**修饰类 不可以被继承！**

**public** **final** **class** Textxiushilei {

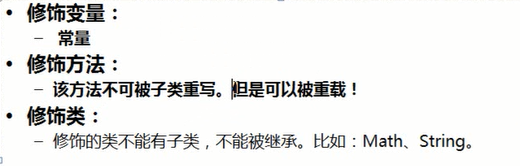
//这个类将不会被继承了，因为有了final

}

//class lei extends Textxiushilei{

//这个将会是一个错误因为不会继承上面的类

//}



14.封装 那些该封装（哪些该让别人知道哪些不该让别人知道）

1.private 自己的类

**public** **class** Text02 {

**public** **static** **void** main(String args[]){

Text01 s = **new** Text01();

// s.wo();不可以调用的这里要调用的方法是private的只能在自己的类里面调用，不可被其他的类调用

}

}

**public** **class** Text01 {

**private** String suiyi;

**private** **void** wo(){

System.*out*.println("测试private封装");

}

}

2.default 自己的包内（前面什么都不加）

**public** **class** TextDefalult2 {

**public** **static** **void** main(String args[]){

TextDefault s = **new** TextDefault();

s.ma(); //**可以调用**

}

}

**public** **class** TextDefault {

**void** ma(){ //前面什么都不加时就是default

System.*out*.println(" 这里我们是在检测default");

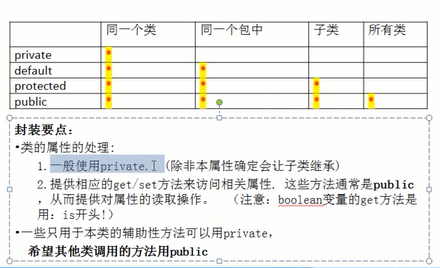
}

}

3.protected 自己的子类也可以

如果实在其他包中的话，我们必须导入@import包（像Scanner导入一样）

4.public 所有的类 （我们学习过了）



15多态（必须使用继承）

1>在没有多态的时候

**public** **class** Text {

**public** **static** **void** Textsheng(mao s){ //重载

s.sheng();

}

**public** **static** **void** Textsheng(gou s){

s.sheng();

}

**public** **static** **void** Textsheng(pig s){

s.sheng();

}

**上面的部分得一个一个写不可以省略**

**public** **static** **void** main(String args[]){

gou s = **new** gou();

*Textsheng*(s);

}

}

**class** Animal{

**public** **void** sheng(){

System.*out*.println("哈哈");

}

}

**class** gou **extends** Animal{

**public** **void** sheng(){

System.*out*.println("汪汪");

}

}

**class** mao **extends** Animal{

**public** **void** sheng(){

System.*out*.println("喵喵");

}

}

**class** pig **extends** Animal{

**public** **void** sheng(){

System.*out*.println("哼哼");

}

}

2》使用多态的时候（下面我们直接看调用过程）

**直接从父类着手这里的父类是** Animal

或者我们可以直接写成

Animal s = new pig();

Textjiao(s);

**public** **class** Duotaikaishi {

**public** **static** **void** Textjiao(Animal a){

a.sheng();

}

**public** **static** **void** main(String args[]){

pig s = **new** pig();

Animal b = s ;

*Textjiao*(b);

}

}

父类的引用指向子类的对象

3》Animal 中有的才可以 比如pig中新定义的方法不可以直接通过Animal（因为Animal中没有这个方法） 变量进行调用但是我们可以通过强制转型

**public** **class** Qiangzhizhuanxinglei {

**public** **static** **void** zhuanxing(Animal a){

a.sheng();

}

**public** **static** **void** main(String args[]){

Animal s = **new** pig();

*zhuanxing*(s);

s.xinjiade(); **//这是错误的Animal没有此方法**

}

}

**public** **static** **void** main(String args[]){

Animal s = **new** pig();

*zhuanxing*(s);

s.xinjiade(); **//这是错误的Animal没有此方法**

**pig a = （pig）s; 强制转型**

**a.xinjiade();**

}

**//也可以使用 new pig调用**

注意：变量一般是看前面的类型不管是什么类型

4》学习一个instance

**public** **class** Instanceof {

**public** **static** **void** zhuanxing(Animal a){

**if**( a **instanceof** pig){ //确认如果传进来的是pig

((pig)a).xinjiade(); //必须进行强制转型转型，因为编译器只看前面的Animal 中有的属性

}

}

**public** **static** **void** main(String args[]){

Animal s = **new** pig();

*zhuanxing*(s);

}

}

16.抽象类abstract（包含了抽象方法的类，只有方法的声明没有方法体）

1>当我们的子类中都要重写父类中的某个方法时，我们没有必要写出父类中的该方法所以我们

1.直接声明即可

2.方法前面加上abstract

3.类的前面也要加上 abstract

4**.子类中必须重写**

**public** **abstract** **class** Chouxiang {

**public** **abstract** **void** run();//每个继承的子类都要重写，所以我们直接省略了算了，这里只是声明

}

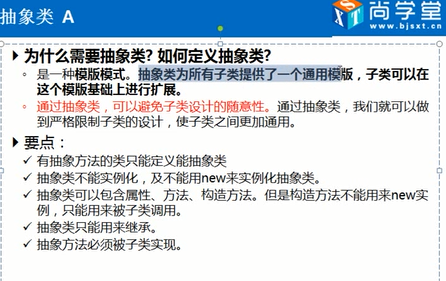
**class** cat **extends** Chouxiang{

**public** **void** run(){

}

}

2>



17.接口（将来时用来实现**implents**的任务，而不是继承）

1.>接口中的出抽象将会必须实现

---》新建 ——》 接口（之前我们的都是类）-》

-----》下一步是对他进行实现而不是继承了-🡪

**public** **interface** interfance {

//接口中只有常量和抽象方法

/\*public static final接口中常量定义时都是自动具有的\*/

String *chagnliang* = "常量";

**public** **void** fangfa(); //之后的函数将会对他进行实现

**public** **int** fa(**int** a,**int** b);

}

//之后我们的设计人员来进行实现它的抽象

**public** **class** Shixianjiekou **implements** interfance{ //implements是实现的意思

**public** **void** fangfa(){

System.*out*.println("实现接口中的一个方法");

}

**public** **int** fa(**int** a,**int** b){

**return** a + b;

}

}

2**>接口是公开的 ，public 方法和常量**

3》实现双接口实现（体现分离 例如，我是一个学生 必须满足的条件看下面的程序）

**class** bird **implements** Julizi,shitou{

//这里是俩个接口我们下面必须进行实现

**public** **void** fly(){

System.*out*.println("我想在天上飞");

}

**public** **void** main(){

}

}

4》实现对他的引用

和多态的地方的引用是一个意思！

5》接口的多继承

**interface** wo **extends** shitou,Julizi{

void mas();

}

18.回调//啥也没不用管

19.内部类 的使用和调用

注意：1>我们直接使用nose new 是不可以的

2>第一中引用方法 nose b = a.**new** nose();

3>或者 face.nose c = **new** face().**new** nose()

**public** **class** Lizi {

**public** **static** **void** main(String args[]){

// nose a = new nose();错误的不可以直接调用内部类

face a = **new** face();

nose b = a.**new** nose(); //这利用的时候必须导入

face.nose c = **new** face().**new** nose(); //这样也是可以的

}

}

**class** face{

**class** nose{ **内部类**

**public** **void** mao(){

System.*out*.println("内部类");

}

}

}

2>内部如何使用外部类的变量

1>方法：face.nose.变量（一般的类）

**class** face{

**int** a;

**class** nose{

**public** **void** mao(){

int a;

System.*out*.println("内部类");

System.*out*.println(face.**this**.a); //重名的时候可以这样，不重名时候我们可以直接使用 a

}

}

}

2>内部类不可以定义为静态的方法

比如 static void mao() 是不可以的

3>静态类的使用

**class** face{

**int** a;

**static** **class** fan{ //静态内部类

// 静态里面不可以调用外面的非静态的

**public** **void** fa(){

System.*out*.println("这里检测一个静态的内部类");

}

**}**

}

**public** **class** Lizi {

**public** **static** **void** main(String args[]){

face.fan e = **new** face.fan();

//只有静态才可以这样调用否则是上面的调用

e.fa();

}

}