1.成员变量：定义时可以对其进行初始化，java使用默认的值对其进行初始化

1〉String null

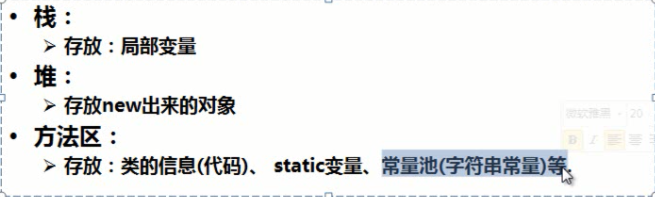
2〉int 0

3〉char

局部变量必须初始化，否则会报错。

2.类是由属性（成员变量） 和 方法组成

3.



例

**public** **class** TestMianxiang {

//静态的数据

String name;

**int** id; //学号

**int** age;

String gener;

**int** weight;

//动态的行为

**public** **void** study()

{

System.*out*.println(name+"留学生");

}

**public** **void** shouzhang(String sname){

System.*out*.println(name+"向"+sname+" 说首长好");

}

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

TestMianxiang aa=**new** TestMianxiang(); //先会通过类加载器寻找类的信息，然后方法区中就有了类的信息

aa.name="张宇晋";

aa.study();

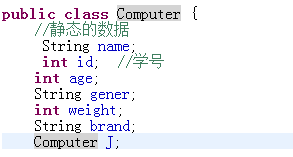
aa.shouzhang("赵春雨");

}

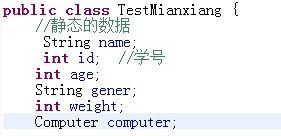
}

4.不同类之间的互相调用；

Computer类



MianXiang类



新类TestLeilianjie

**public** **class** TestLeilianjie {

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

TestMianxiang s1 = **new** TestMianxiang(); //调用TestMianxiang类

s1.name="张宇晋";

Computer s2 = **new** Computer(); //调用Computer类

s2.brand = "联想";

s1.computer = s2; //将时s2赋值给s1中的computer 联系两个类

s2.brand="戴尔"; //此时将brand改变后再次赋值给brand

String a=s1.computer.brand; //找到 brand

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

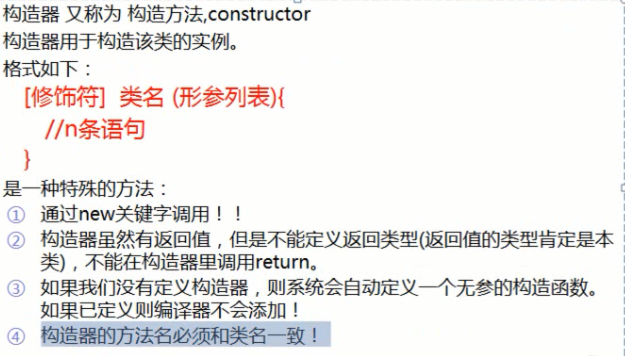
}

}

运行结果 戴尔 没有s2.brand="戴尔";时运行结果为 联想

5.构造方法（构造方法constructic）

1〉比如tesr01等对象 需要 new来调用



2〉没有定义构造方法，系统将会自动构造一个无参的构造函数。(要求方法中没有任何内容，比如3中的 System.*out*.println("张宇晋");因该是没有的，否则会出现错误)

**public** **class** TestConstructor {

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

Car aa = **new** Car();

}

}

此时不会运行错误

系统将会自动在Car中加上构造方法

public Car（）{

}

**public** **class** Car {

**int** speed;

}

**public** **class** TestConstructor

{

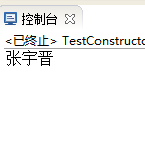
**public** **static** **void** main(String[] args){

Car aa = **new** Car();

}

}

运行（会调用构造方法中的内容）



3> 定义了构造器

**public** **class** Car {

**int** speed;

**public** Car(){

//什么都不用写void等不写 System.*out*.println("张宇晋");

}

}

5.方法的重载

或者关系

1〉

例1：参数的个数不同

**public** **class** TestOverload {

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

{

Mymath aa = **new** Mymath();

**int** h=aa.add(4,5);

**int** m=aa.add(4,5,5);

System.*out*.println(h+" "+m);

}

}

**class** Mymath{

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

**int** z=a+b;

**return** z;

}

**public** **int** add(**int** a,**int** b,**int** c){

**int** d=a+b+c;

**return** d;

}

}

例2：参数的类型不同

**public** **class** TestOverload {

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

{

Mymath aa = **new** Mymath();

**int** m=aa.add(4,5,5);

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

}

}

**class** Mymath{

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

**int** z=a+b;

**return** z;

}

**public** **int** add(**double** a,**double** b){

**int** z=(**int**)(a+b);

**return** z;

}

}

例3：参数的顺序不同

**public** **class** TestOverload {

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

{

Mymath aa = **new** Mymath();

**int** e=aa.add(6.12,5.1); //此处输入我们想要进行比较的数字

System.*out*.println(e);

}

}

**class** Mymath{

**public** **int** add(**double** a,**int** b){

**return** (**int**)(a+b);

}

**public** **int** add(**int** a,**double** b){

**return** (**int**)(a+b);

}

}

2〉构造器的重载

**public** Mymath(**int** a){

}

**public** Mymath(**int** a,**int** b)

{

3〉构造器的调用之2

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

**double** x,y,z;

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

{

x = \_x;

y = \_y;

z = \_z;

}

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

Point aa = **new** Point(2.1,3,5);

System.*out*.println(aa.x);

}

}

6.stastic静态变量

1〉通过类名直接调用