**Static**

方便在没有创建对象的情况下来进行调用（方法/变量）。

**重载：**

重载(overloading) 是在一个类里面，方法名字相同，而参数不同。返回类型可以相同也可以不同。

**递归调用：**

递归调用是一种特殊的嵌套调用，是某个函数调用自己或者是调用其他函数后再次调用自己的。

**面向对象简介**

面向过程就像是一个细心的管家，事无具细的都要考虑到。而面向对象就像是个家用电器，你只需要知道他的功能，不需要知道它的工作原理。

面向过程是一种是事件为中心的编程思想。就是分析出解决问题所需的步后用函数把这写步骤实现，并按顺序调用。面向对象是以“对象”为中心的编程思想。

汽车发动、汽车到站   
这对于“面向过程”来说，是两个事件，汽车启动是一个事件，汽车到站是另一个事件，面向过程编程的过程中我们关心的是事件，而不是汽车本身。针对上述两个事件，形成两个函数，之后依次调用。然而这对于面向对象来说，我们关心的是汽车这类对象，两个事件只是这类对象所具有的行为。而且对于这两个行为的顺序没有强制要求。

三大基本特性：封装、继承、多态

封装是 Java 类的编写规范、继承是类与类之间联系的一种形式、而多态为系统组件或模块之间解耦提供了解决方案。

**类与对象**

类：是抽象的概念集合，表示的是一个共性的产物，类之中定义的是属性和行为（方法）；   
对象：对象是一种个性的表示，表示一个独立的个体，每个对象拥有自己独立的属性，依靠属性来区分不同对象。

总结：类是对象的模板，对象是类的实例。类只有通过对象才可以使用，而在开发之中应该先产生类，之后再产生对象。

实例：定义类

class Person { // 类名称首字母大写

String name ;

int age ;

public void tell() { // 没有static

System.out.println("姓名：" + name + "，年龄：" + age) ;

}

}

实例化对象：

方式一

类名称 对象名称 = new 类名称 () ;

Person p =new Person();

方式二

类名称 对象名称 =null; Person p=null;

对象名称=new 类名称(); p=new Person();

**封装**

封装最主要的功能在于我们能修改自己的实现代码，而不用修改那些调用我们代码的程序片段。  
适当的封装可以让程式码更容易理解与维护，也加强了程式码的安全性。

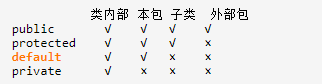
生活中的封装：我们日常使用的电脑主机，把cpu、内存、主板等等都封装到机箱里面去。假如没有机箱的话的出现什么问题，主机、主板全部都散落在一处，然后开机没有开机按钮，那么需要我们直接操作接跳线才能把电脑开启。这样子的话假如操作不慎的话，会让机器损坏危险,那么假如用机箱封装起来的话，那么就不需要这样子做了。体现了封装的---安全特性。

**public：**在任何地方，只要能访问到该类，即可访问到该成员。

**protected：**该类的扩展子类访问。

**默认：**该成员只能由同一个包中的其它类访问。

**private：**类中的该成员只能由该类自己的成员调用，而无法被该类以外的其它类访问。



体现到java中：未封装前

public class EmployeeDemo {

public static void main(String[] args) {

// 创建对象

Employee jack = new Employee();

// 进制通过类名.成员的形式调用成员。初始化实例变量

jack.name = "jack";

jack.id = "123456";

jack.gender = "男";

// 调用成员方法

jack.work();

System.out.println();

// 传入非法的参数

jack.gender = "不是男人";

jack.work();

}

}

class Employee {

String name;

String id;

String gender;

public void work() {

System.out.println(id + ":" + name + ":" + gender + " 努力工作中！！！");

}

}

有封装的时候

设置类的属性为private(关键字)，不能使用对象名.属性名的方式直接访问对象的属性

public class EmployeeDemo {

public static void main(String[] args) {

// 创建对象

Employee jack = new Employee();

//编译报错

jack.name = "jack";

jack.id = "123456";

jack.gender = "男";

// 编译报错

jack.gender = "不是男人";

jack.work();

}

}

class Employee {

//使用了private修饰了成员变量

private String name;

private String id;

private String gender;

public void work() {

System.out.println(id + ":" + name + ":" + gender + " 努力工作中！！！");

}

}

总结：如果不使用封装，很容易赋值错误，并且任何人都可以更改，造成信息的    不安全

4.构造方法与匿名对象

构造方法作用：为类中的属性初始化

构造方法使用：

class Person{

public Person(){ // 声明构造方法

System.out.println("一个新的Person对象产生。") ;

}

};

public class ConsDemo01{

public static void main(String args[]){

System.out.println("声明对象：Person per = null ;") ;

Person per = null ; // 声明对象时并不去调用构造方法

System.out.println("实例化对象：per = new Person() ;") ;

per = new Person() ;//实例化对象

}

};

使用注意：\*构造方法的名称必须与类名称一致；   
构造方法的声明处不能有任何返回值类型的声明；   
不能在构造方法中使用return返回一个值；\*

匿名对象：匿名对象就是没有明确的给出名称的对象。一般匿名对象只使用一次

class Person{

private String name ;

private int age ;

public Person(String n,int a){ // 声明构造方法，为类中的属性初始化

this.setName(n) ;

this.setAge(a) ;

}

public void setName(String n){

name = n ;

}

public void setAge(int a){

if(a>0&&a<150){

age = a ;

}

}

public String getName(){

return name ;

}

public int getAge(){

return age ;

}

public void tell(){

System.out.println("姓名：" + this.getName() + "；年龄：" + this.getAge()) ;

}

};

public class NonameDemo01{

public static void main(String args[]){

new Person("张三",30).tell() ;

}

};