# 静态成员变量：

**1.静态成员随着类的加载而加载。**

**2.静态成员是优先于对象存在的。**

**3.资源共享。被static修饰的内容，能被该类下所有的对象所共享。**

**4.可以通过类名点的形式被调用。**

## Static的注意事项：

**1.静态方法中是没有this关键字的。**

**2.静态只能直接访问静态。**

## Java中的Collection类详解

**数组的特点：**

**1.只能存储同一种数据类型的数据；**

**2.一旦初始化，长度固定；**

**3.数组中元素与元素之间的内存地址是连续的。**

**注意：object类型的数组可以存储任意类型的数据。**

**集合是存储对象数据的集合容器：**

**集合比数组的优势：**

**1.集合可以存储任意类型的数据，数组只能存储同一种数据类型的数据；**

**2.集合的长度是会发生变化的，数组的长度是固定的。**

**Collection类集合：单例集合的根接口：**

**1.如果对象实现了List集合类，具备的特点：有序、可重复。**

**2.ArrayList集合：底层是维护了一个Object数组实现的，特点：查询快、增删慢。**

**3.LinkedList集合：底层使用了链表数据结构实现的，特点：查询慢、增删快。**

**4.Set类集合：具备的特点：无序、不可重复。**

**5.HashSet:底层是使用了哈希表来支持的，特点：存取速度快。**

**6.TreeSet：如果元素具备自然顺序的特性，那么就按照元素自然顺序的特性进行排序存储。**

**集合只能存储引用类型（又叫基本类型的包装类），其长度是不固定的。**

**泛型：用来限定集合中应该存储什么类型的数据。**

**泛型一般只和集合相结合使用。**

**注意：char的引用类型是：Character；int的引用类型是：Integer**

**Jdk的新特性：自动拆装箱；**

**例：Integer i=10;自动装箱：把基本类型的数据，转成相应的引用类型。**

**int a=I;自动拆箱：把引用类型的数据转成相应的基本类型的数据。**

**HashMap类型的对象在定义是不要忘了初始化。**

**HashMap中的键具有唯一性，是无序的。**

**HashMap方法：put（）：向集合里面添加数据，第一次添加时，返回null；如果重复添加会用新值来覆盖旧值，并返回旧值。**

## 数组和集合的不同

**1.数组和集合的长度不同：数组的长度固定，集合的长度不固定。**

**2.数组既能存储引用类型，又能存储基本类型的数据；集合只能存储引用类型的数据。**

## 数组、集合、字符串分别怎么获得其长度

**数组：length属性**

**集合：size方法**

**字符串：length方法**

## 增强for循环

**格式：for(数据类型 变量名：要遍历的集合或者或数组)｛**

**｝**

## ArrayList集合

**ArrayList集合是有序（存取顺序一致），可重复的。**

**集合转数组：ArrayList#toArray()方法；**

**数组转集合：Arrays#asList()方法；**

**注意：1.数组转成集合之后，不能进行增加或者删除操作，但是可以进行修改操作；**

**2.如果是基本类型的数组转成集合，那么是把整个数组当作集合的一个对象来转的。**

## Final关键字修饰的类，方法、成员变量

**修饰的类：不能被继承，但是可以继承别的类。**

**修饰的方法：不能被子类重写**

**修饰的变量：**

**修饰的是基本类型的变量：是值不能发生改变，因为是一个常量。**

**修饰的引用类型的变量：是地址值不能发生改变，但是属性值可以发生变化。**

## 抽象类

**抽象类可以通过创建其子类对象来完成初始化。**

**抽象类中有构造方法：用于子类对象访问父类数据前，对父类数据进行初始化。**

**抽象类的子类：如果是抽象类，这样不用重写父类的抽象方法。**

**不能与abstract共存的关键字有哪些：private、final、static**

**Private、final属于设计冲突，static属于无意义。**

**个人笔记：**

**1.抽象类不能被实例化，如果被实例化，就会报错，编译无法通过。只有抽象类的非抽象子类可以创建对象。**

**2.抽象类中不一定包含抽象方法，但是有抽象方法的类一定是抽象类。**

**3.抽象类中的抽象方法只是声名，不包含方法体，就是不给出方法的具体实现（也就是方法的具体功能）。**

**4.构造方法，类方法（用static修饰的方法）不能声名为抽象方法。**

**5.抽象类的子类必须给出抽象类中的抽象方法的具体实现，除非该子类也是抽象类。**

## 接口

**接口中没有构造方法。**

**接口的子类：如果是抽象类可以不用重写抽象方法；如果是普通类，必须要重写父接口中的所有抽象方法。**

**JDK1.7及以前接口中定义的成员变量为final static，方法为abstract**

**JDK1.8以后可以定义非抽象的方法，若定义非抽象的方法时，需要加上default修饰符。**

**接口与接口之间可以单继承，也可以多继承。**

## Java中抽象类和接口的区别：

**1.成员特点不同：**

**抽象类中的成员比普通类多一种抽象方法（抽象方法也可以不写）。**

**接口中有且只有抽象方法（jdk1.8以后可以有带方法体的方法）和常量**

**2.关系特点不同：**

**类与类之间是继承关系，只能单继承，不能多继承，但是可以多层继承；**

**类与接口，实现关系，可以单实现，也可以多实现，还可以在继承一个类的同时实现多个接口。**

**接口与接口，继承关系，可以单继承，也可以多继承。**

**3.设计理念不同**

**抽象类，定义的是整个继承体系的共性内容。**

**接口，定义的是整个继承体系的扩展内容。**

## 多态

**多态中，只有访问非静态方法是编译看左，运行看右的，其它都是编译和运行都看左，原因是因为方法有重写。**

**多态在实际开发中经常用到的应用场景：**

**父类型可以做为方法的形参的参数类型，这样可以接受任意的子类对象。**

**个人笔记：**

**1.使用父类类型的引用指向子类的对象。**

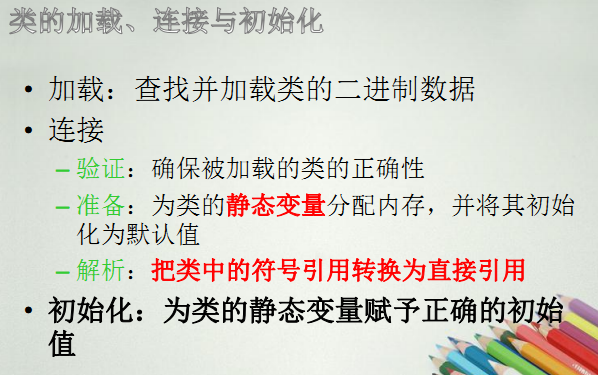
**2.该引用只能调用父类中定义的方法和变量。**

**3.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，会调用子类中的这个方法。**

**4.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。**

## Java中的类加载机制

**每个java程序执行前都必须经过编译、加载、连接、和初始化这几个阶段，后三个阶段如下图所示：**

****

## List去重

**思路1：创建新集合，遍历老集合，看新的集合中是否包含重复的元素，没有的话添加到新的集合中去。**

**思路2：创建一个set集合（无序，不重复），将集合中的元素放到set中去，清空旧的集合，再将set放到旧的集合中去。**

## try/catch块

**alt+shift+z：选中要处理的语句。**

**try{**

**//尝试要执行的代码**

**}catch {**

**//出现异常后要处理的代码**

**} finally {**

**//一般是用来释放资源的，正常情况下，里面的代码会永远执行**

**}**

## Java中的File类

**File类方法中的delete()方法要注意：**

**1.java中的删除是不走回收站的。**

**2.要删除的文件下面不能有其它内容。**

## IO流

**计算机中的最基本的单位是：字节**

**在GBK码表中，一个中文占两个字节，在UTF-8码表中，一个中文占三个字节; 数字、字母、特殊符号不管在什么码表中，都是占一个字节。**

**输入流：是用来读取数据的。**

**输出流：是用来写出数据的。**

**Io流的顶层都是抽象类，集合的顶层都是接口。**

**核心代码：**

**1.创建输入流对象，封装数据源文件。**

**2.创建输出流对象，封装目的地文件。**

**3.定义一个变量，用来记录读取到的字节。**

**4.循环读取，只要条件满足就一直读，并将读取到的内容记录给变量。**

**5.将读取到的内容写入到目的文件中。**

**6.关闭流，释放资源。**

### 字节流：

**可以操作任意类型的文件，但是操作中文的时候，有可能出现乱码。**

### 字符流：

**专门操作中文文件的。**

**当一个文件可以用微软自带的记事本打开的时候，并且里面的内容没有乱码，就可以考虑用字符流来处理它。**

## JDK1.7以后释放资源的try—catch

**Try () {**

**}**

**写在try小括号里面的代码会在大括号里面的代码执行完毕后自动释放资源。**

## 多线程

**一台电脑上可以有多个进程，这些进程之间的数据是相互隔离的；**

**一个进程可以有多个线程，这些线程共享当前的数据。**

**多线程并行：多个线程同时执行，要求：cpu是多核的；**

**多线程并发：多个线程同时请求执行，但是cpu只能同时执行一个，于是安排这些线程交替执行，由于时间间隔非常短，我们感觉不出来。**

**多线程实现方式：**

**方式一：**

**1.定义一个类去继承Thread类**

**2.重写Thread#run方法，把要执行的代码写到run()方法中。**

**3.创建线程对象**

**4.开启线程**

**注意：**

**a.开启线程调用stat()方法。**

**b.同一线程不能重复开启，否则会报线程状态非法异常。**

**方式二：**

**1.定义一个类去实现runnable接口**

**2.重写runnable接口的方法，把要执行的代码写到run()方法中去。**

**3.创建Runnable接口的子类的对象，将其作为参数传入Thread类的构造方法中去。**

**4.开启线程。**

## 锁对象

**匿名内部类访问其所在方法的局部变量时，该变量必须为final修饰，原因是为了延长该变量的生命周期，JDK1.8以后会自动加上final这个修饰符。**

**1.非静态方法的锁对象是this**

**2.静态方法的锁对象是该类的字节码文件对象**

## 面向网络编程

**三大要素：ip地址、端口号、协议**

**UDP协议：1.面向无连接**

**2.采用数据包的形式发送数据，每个包的大小不能超过64kb**

**3.不安全（不可靠协议）**

**4.效率高**

**5.不区分客户端和服务器端（发送端和接收端）**

**TCP协议：1.面向有连接（三次握手）**

**2.采用io流的形式发送数据，数据无大小限制**

**3.安全（可选协议）**

**4.效率低**

**5.区分客户端和服务器端**

**面向网络编程：**

**客户端：**

**1.创建客户端socket对象，指定ip地址和端口号。**

**2.通过socket对象来获取和服务器端交互的流对象。**

**getOutputStream()获取输出流对象，可以向服务器端写数据。**

**getInputStream()获取输入流对象，可以读取服务器端反馈的数据。**

**3.关闭流对象。**

**4.关闭socket对象。**

**服务器端：**

**1.创建服务器端对象（ServerSocket对象），指定端口号；**

**2.通过ServerSocket#accept()监听客户端的连接，如果有客户端请求连接，并且校验数据一致后，将返回一个socket对象，和该客户端进行交互。**

**3.通过serverSocket对象来获取和客户端交互的流对象**

**getOutputSteam()获取输出流对象，可以向客户端写数据。**

**GetInputStream()获取输入流对象，可以读取客户端反馈的数据。**

**4.关闭流对象。**

**5.关闭相应的客户端的Socket对象。**

## Switch

**Jdk1.5以前switch的小括号里面可以放byte、short、char、int。**

**Jdk1.5以后，可以放枚举**

**Jdk1.7以后，可以放String**

## 方法重载：

**同一类中，出现方法名相同，参数列表不同（个数不同，对应的数据类型不同）的两个（或以上）的方法时，称为方法重载。**

**方法重载和返回值类型无关。**

## 方法重写：

**子父类间：子类出现和父类一一模一样的方法时，称为方法重写；**

**方法重写要求返回值类型也一模一样（或者有子父类关系）**

## 代码块

**代码块分为四大类：构造代码块、静态代码块、同步代码块、局部代码块**

**局部代码块：让变量尽可能早的从内存中释放掉，从而续约资源，提高效率。**

## InstanceOf关键字

对象名 InstanceOf 数据类型 返回值是布尔类型

## Stringbuilder类

Delete(int start,int end) 包左不包右

## 泛型

泛型只针对引用类型

## 集合的遍历方式：

**单例集合：1.增强for；2.迭代器遍历；3.转数组遍历**

**双例集合：1.通过键获取值；2.通过键值对对象获取对应的键和值**

## Java中equals方法的使用：

一般将常量放到equals的前面，由常量去调用equals方法，去避免空指针异常。

## 集合操作：

**1.在遍历集合的时候，如果想要同时添加元素或者删除元素，会报异常，该异常名称为ConcurrentModificationException**

**有两种解决方案：方案一：用列表迭代器来解决（注意：要用列表迭代器中的添加或者删除方法）**

**方案二：使用CopyonWriteArrayList**

## Properties

**是一个双列集合，也是一个唯一能够和IO流交互的集合类。**

**它可以直接从流中读取数据，也可以直接写入数据流中。**

## 内存输出流

把整个内存当做一个缓冲区，可以往里边写入数据，然后一次性的从内存中获取所有的数据。ByteArrayOutputStream；先调用write()方法向内存中写入数据 ，然后再调用toString()方法从内存中读出来。

## Java中静态方法和实例方法的区别：

**1.在外部调用静态方法时，可以使用类名.方法名的方式来调用，也可以使用对象名.方法名的方式来调用。而实例方法只有后一种方式来调用。**

**2.静态方法在访问本类的成员时，只允许访问静态成员，而不允许访问实例成员变量和实例方法；实例方法则无此限制。**