|  |
| --- |
| Float不能和double比较， 输出false  -9 % 2 = -1 9 % 2 = 1 正负看前面的操作数  Swith 中可以放 byte、short、int、char、enum、String（jdk1.7之后）  Break 作用1 停止swith多分支判断，作用 2 可以停止当前循环  Return 作用 1 停止当前方法 作用 2 方法的返回值  Contiune : 停止循环的某一次，接着开始下一次循环。  方法的重载 同一个类中，多个方法名相同，但是参数不同的方法。  重载只参数列表有关系，和返回值没有关系。参数顺序不同也是重载  父类静态代码块🡪子类静态代码块🡪父类代码块🡪父类无参构造器🡪子类代码块🡪子类无参构造器  外部比较器 创建一个新类实现Comparator接口，重写compare方法。  内部比较器 在已有类的基础上实现Compare接口，重写compareTo方法。 |

|  |
| --- |
| Finally 无论try代码块是否报错，finally中的代码都会执行，这个位置一般只放资源的关闭操作。  Finally和final和finalize的区别  Finally处理异常的操作，无论try中代码是否报错finally中代码都会执行  Final:是一个关键字，可以修饰类，方法，变量。  Finalize：垃圾回收方法操作  Integer a = 10 没有创建对象，因为valueOf方法内部缓存了一个从-128~127的数组  String 常用方法   1. charAt( ) 返回指定索引处的char值 输入索引，查字符 2. compareTo（） 比较两个字符的值 根据ASCLL码比较 比大小 3. concat（） 连接字符串，必须用变量接受它，String是不可变的 4. contains（）返回字符串是否包含参数的字符串 5. endsWith（）判断字符串是否已指定后缀结束 6. equals（）将此字符串与指定的对象比较 只能是不是相等 7. indexOf（）输入字符 查索引 8. isEmpty（）判断字符串是否为空 9. length（） 长度 10. repalce（） 输入字符串 和想替换的字符串 11. split（） 想根据什么为分隔符 12. startsWith（）判断是否以XX开头 13. subString（） 输入开始和结束索引，来截取字符串 14. toLowerCase( ) 转换成小写 15. toUpperCase（）转换成大写 16. trim（）删除字符串中的开头和结束的空格   String 会存在常量池中  Jvm底层对String变量加法做了优化，其实是一个StringBuilder对象调用append方法，一个个添加。  多态含义是可以实现代码的扩展性，从而符合开闭原则， 对拓展开放，对修改关闭  父类引用指向子类对象。  StringBuffer JDK1.0提供的类，线程安全，做线程同步检查，效率较低  StringBuilder JDK1.5提供的类，线程不安全，步做线程同步检查，因此效率较高， |

|  |
| --- |
| Servlet 主要功能在于能在服务器中执行并生成数据，servlet 使用单进程多线程的方式运行，servlet 生命周期分为5个阶段 1. 类加载 2. new 实例化 3. init初始化 4.service执行服务 5. destroy回收销毁  ServletContext对象的作用   1. 相对路径转绝对路径 2. 获取容器的附加信息 3. 获取web.xml文件中的信息 4. 全局容器   转发的特点：转发的整个操作没有新的请求，只是把之前的请求进行了传递，转发的整个操作在服务器内部进行的，浏览器不知道服务器内部的转发，转发是服务器内部的跳转，浏览器不知道转发的操作，所以地址栏不会改变，转发的整个操作没有新的请求，所以可以实现数据的共享。  重定向原理：重定向整个操作产生了新的请求，浏览器知道发生了重定向操作，重定向操作对于浏览器是可知的，所以跳转前后浏览器的地址改变了 跳转前后产生了新的请求，所有没有办法实现数据的共享。  拦截器是springMVC的，而过滤器是servlet的  控制事务的传播行为   1. REQUIRED(默认值) 如果当前有事务，就在事务中执行，如果当前没有事务，新建一个事务。 2. SUPPORTS 如果当前有事务就在事务中执行，如果当前没有事务，就在非事务状态下执行 3. MANDARTORY（mandatory 强制的）必须在事务内部执行，如果当前有事务，就在事务中执行，没有事务报错 4. REQUIES\_NEW 必须在事务中执行，如果当前没有事务，新建事务，如果当前有事务，就把事务挂起。   获取主键自增的值   1. 在mybatis中 useGenerateKeys = ‘true’ keyProperty = ‘id‘ useGenerateKeys:开启获得主键自增的方式，keyProperty：把主键自增的值赋值给对象的指定属性中。 2. 线程更加安全 select @@identity keyProperty = ‘id‘ |

|  |
| --- |
| Js中的数据类型 number（数字） String（字符串）object（对象）boolean（布尔类型）  Undefined（未定义） NaN（not a number） null（空对象）  JS中 等值符== 先比较类型，如果类型一致，再比较内容，如果类型不一致，会强制转换成number再比较内容  等同符 === 数据类型不同，直接返回false 如果类型相同 才会比较内容。 |

+++++++

|  |
| --- |
| 设计模式  简单工厂模式  好处：如果有多个某一个类对象现在最直接交给工厂处理即可，我们在使用的时候直接调用工厂方法即可，至于工厂内部是如何创建的用户没有关系，用户直接使用更加方便  缺点：不符合开闭原则，对拓展开放，对修改并不符合要求  我们的目的是 在不修改源代码的基础上实现功能的拓展  工厂方法模式  好处：完全符合了开闭原则，我们新增一个类，直接增加该类的工厂即可，完全不需要修改源代码（补充：工厂方法有的时候不单单是return对象，有可能用到反射等，所以很有必要用）  缺点：代码书写的时候相对麻烦  单例模式————饿汉式（保证对象唯一）   1. 构造器私有化 2. 本类内部创建对象 3. 提供public static方法，暴露该对象   缺点：这种方式是在类加载的时候就把对象创建好了，如果我们只是类加载并没有使用对象，这个创建对象就会白白浪费了。  单例模式————懒汉式（保证对象唯一）   1. 构造器私有化 2. 本类内部声明对象，赋值为null 3. 提供public static 方法，判断这个对象是不是null，如果为空，new对象，否则直接返回对象即可，这样类加载的时候就不会创建对象，调方法的时候才会创对象。   优点：这种方式效率会更高一点，每一次获得对象的时候都会进行判断，如果存在就不会继续创建对象，保证了对象资源不会被浪费。  缺点：在多线程中没有办法保证对象的唯一（必须掌握）  弥补方案： 就是使用线程锁（锁前后两个if判断，不冲突，前面的提高效率，后面的防止创建多个对象，有可能不止一个线程进入第一个if判断 双重锁定）  拷贝模式（克隆模式） 重写clone方法  创建的对象里面引用了另一个对象，这样只用浅克隆的话，改里面的那个对象的话，就会全改，没有办法改其中一个的效果，只能用深克隆。  责任链模式 经典案例：报销流程 过滤器Filter利用了责任链模式 减少耦合，更加符合开闭原则  装饰模式：不在使用很多继承，为了改变在继承中出现众多子类的问题  观察者模式 监听器 listener |

|  |
| --- |
| Create database 数据库名 新建数据库  Show databases 查询数据库  User 数据库名 选中当前数据库  Show tables 查询数据库下所有表  Drop database 数据库名 删除数据库  Create table 表名（属性，属性类型） 创建表  Desc 表明 查询表结构  Show create table 表名 查询完成的建表语句  Alter table 表名 add 字段名 字段类型 （first或者after属性） 添加字段  Alter table 表名 modify 字段名 字段属性 修改字段的类型  Alter table 表名 change 字段名 字段名 字段属性 数据类型和字段名称都可以改  Alter table 表名 drop 字段名 删除字段  Alter table 表明 rename 表明 把表重新命名  Insert into 表名（字段） values（字段值） 插入  Update 表名 set 字段=‘ ‘ where 字段 = ’ ‘ 修改  Delete from 表明 where 字段 = ‘ ‘ 删除数据  Truncate table 表明 先把表复制一份，再删除整个表  Select 字段 from 表名 where 条件 group by 字段 having 条件  这个优先级很重要 from where group by select having order by  Distinct 去重操作  多行函数 sum count max min avg  查询重名操作 可以按名字分组，再计数 大于1 就是重名了  多表查询 分为 内连接 和 外连接  内连接分为 交叉连接 cross 返回笛卡尔积  自然连接 nutural 效率低  Using 效率高  On连接 用的最多  Union 把两个表连一块，相当于全连接。  类方法，指的是静态方法  JSP的本质就是java代码  JSP是被服务器解析 服务器解析JSP中的Java代码，然后把结果返回给浏览器  JSP = HTML + Java  JSP中书写Java代码的三种方式  小脚本 相当于方法 局部变量  声明式 相当于方法外的代码 成员变量  表达式 就是一条输出语句    服务器解析分为3步 1. 翻译 jsp—》java 2. 编译 java 🡪 .class 3.解析执行.class文件  Java 有5种方式来创建对象  使用new关键字 使用反射类的Class类的newInstance( )方法 使用反射的Constructor类的newInstance（）方法 使用对象克隆clone方法  使用反序列化的readObject方法。  构造方法不能通过方法名调用！！！！ |

|  |
| --- |
| 浏览器控制跳转方式一共三种  在浏览器中直接输入url地址 只能是get方式  提交表单 推荐式post  超链接 只能是get  解决post请求的中文乱码：req.setCharacterEncoding(“utf-8”)  解决响应的中文乱码 打印信息的时候：resp.setContentType(“text/html;charset = utf-8”)  解决响应响应集合或者数据的时候： resp.setCharacterEncoding(“utf-8”)  Servlet对象用户第一次访问的时候创建，对象只创建一次 但实例多线程应用  Servlet对象是Tomcat容器给我们创建的  Servlet的生命周期 类加载🡪实例化对象🡪初始化🡪服务请求🡪销毁操作  接口中的变量默认是public static final的，他是公共的，静态的，最终的常量，相当于全局常量，可以直接在实现类访问  ThreadLocalMap中使用开放定址法来处理散列冲突  Hashmap中使用的分离链表法。  Synchronize 修饰非静态方法锁的是this对象  修饰静态方法 锁的是class对象  转发和重定向的区别   1. 语句不同 2. 跳转前后地址显示 转发：显示的跳转以前的地址 重定向：显示的跳转以后的地址 3. 跳转前后是否可以共享数据 转发：可以 重定向：不可以 4. 原理不同 5. 效率不同 转发：高 没有新的请求 重定向：低 6. 跳转范围不同 转发：只可以在当前项目中跳转 重定向：服务器任意资源位置 7. 是否造成表单的重复提交 转发：会 重定向：不会 8. 路径不同 但是都支持根路径 转发不支持相对路径   Http协议的特点  支持客户端/服务器模式  简单快速  灵活  短连接 （请求一次，释放一次连接，所以无连接（短连接）表示每次请求只能处理一次请求，优点就是节省输出时间，实现简单）  单向性 服务端永远是被动的等待客户端的请求  无状态 就是对于事务管理没有记忆能力（原子性）cookie和session处理了无状态的  Get和post的区别  Get请求在url中传送的参数是有长度限制的，而post没有。对参数的数据类型get只接受ascll字符，而post即可是字符或者字节  Get比post更不安全 因为参数直接暴露在url上，所以不能用来传递敏感信息。 get参数通过url传递，post放在request body（请求体中），get的速度比post速度块 |

|  |
| --- |
| 泛型 可以认为int String之类的也可以自己定义  补充：new关键字说白了就是把class文件拿到内存中  This关键字 就是代表本对象，比如set方法，除了显式参数外，还有一个隐式参数，那个隐式参数把地址传过来了，this代表这个地址号    List接口的特点：有序 插入顺序 可以重复  Vector 一开始就初始化了长度为 扩容速度为二倍速度  Stack 是 Vector的拓展  抽象类可以继承实体类，前提是实体类必须有明确的构造函数，不写就有默认的 那也算有构造函数 只是要实体类只有有参构造的话，抽象类必须要实现实体类的有参构造器  一般关系数据模型和对象数据模型之间有一下对应关系：表对应类、记录对应对象、表的字段对应类的属性  因为++（）括号里面必须为变量 i++是一个常量  Final关键字：  Final定义的方法里面，不是必须用final定义变量  Final定义的变量，不是必须要在定义的同时完成初始化，也可以在构造方法中完成初始化  Final修饰方法，不能被子类重写，但是可以被重载  Java的访问权限由public protected private和default default不能修饰变量  普通变量不能用abstract修饰 abstract一般修饰方法和类  Abstract 不能和final共用  Boolean类型不能和任何类型进行转换，会报类型异常。  HashMap的长度为什么是2的N次方呢?  为了能让HashMap存数据和取数据的效率高，尽可能地减少hash值的碰撞，也就是说尽量把数据能均匀的分配，每个链表或者红黑树的长度尽量相等。我们首先可能会想到%取模操作来实现。  Hash%length 和 hash&（length-1）的前提都是length是2的n次方  数组复制的这几种方法中，按效率排序：system.arraycopy>clone>arrays.copyOf > for  算法的五个特征：有穷性，确切性，输入项，输出项，可行性。  Char<short<int<float<double 这样的几个类型的相加 总会变成大的那个  抽象类是可以实现接口的，而且抽象类也可以继承自抽象类  抽象方法可以包含抽象方法，也可以不包含  抽象类和接口都是不能被实例化的，只有具体的类才能实例化    补码：按位取反再+1  字符流 = 字节流+ 编码集  后缀是Stream的都是字节流，其他的都是字符流  抽象类和接口的区别   1. 关键字 abstract class interface 2. 子类extends继承抽象类-单继承 子类implements实现接口，多实现 3. 可以由构造器，构造器不是用来实例化的，用来给子类初始化的   接口不能有构造器   1. 成员变量权限public，protected，private和default都可以，   接口成员变量权限默认都是public static final 即接口声明的变量不能被继承   1. 抽象方法权限只有public protect 和default三种   接口方法权限默认为public   1. 抽象类可以包含静态代码块，也可以有静态方法   接口不可以静态代码块，可以有静态代码块   1. 可以有普通方法，而接口中方法都是抽象的，不可以有普通方法 |

|  |
| --- |
| File类的方法   1. getName（）获取文件名称 2. GetPath（）文件的路径 3. length（）文件占用多少个字节 4. Exists（）判断当前文件是否存在 5. getAbsolutePath( ) 文件的绝对路径 6. isDirectory（）判断当前文件是否是一个目录（文件夹） 7. isFile（）判断是否是一个文件 8. isHidden（）是否是隐藏文件 9. canWrite（）是否可写 10. canRead（）是否可读 11. canExecute（）是否可执行 12. madirs（）创建文件夹（目录） 13. createNewFile 创建文件 14. delete 删除 15. listFiles（） 把文件夹里的文件 放到一个数组里   ~n = （-n）-1 |

|  |
| --- |
| Java中的堆是 jvm所管理的最大的一块空间，主要用于存放各种类的实例对象。  在java中，堆被划分成两个不同区域：新生代（Young）、老年代（Old）。  新生代又被划分为三个区域：Eden、From Suuvivor、To Survivor.  这样划分的目的是为了使JVM能够更好的管理堆内存中的对象，包括内存的分配以及回收 堆的大致内存模型为：    静态变量只能在类主体中使用，不能在方法中定义  String split 这个方法默认返回一个数组，\*如果没有找到分隔符，\*会把整个字符串当成一个长度为1的字符串数组  在接口中，属性都是默认的public static final 修饰的，而且必须赋值  String 不可修改的 意思是 另外给string引用赋值，他会指向另外一块常量池的空间，而原来的那块空间里面东西没有改变，也不可改变 StringBuffer和StringBuilder可以修改  成员变量也叫实例变量，另一种是static修饰的静态变量（类变量）  Synchronized 保证三大性，原子性，有序性，可见性  Volatile 保证 有序性，可见性，不能保证原子性  File类是堆文件整体或者文件属性操作的类，不能操作文件内容  IO流是用来操作文件内容的 |

|  |
| --- |
| 线程的创建方式一：  继承了Thread类，重写run方法  Start（）开启线程， start（）🡪JVM🡪run（）  线程和线程之间是竞争关系，共同争抢CPU的资源 所以每次执行结果不一定相同，  SetPriority（） 设置线程的优先级 默认是5 1-10都可以  Thread.CurrentThread.setPriority( ) 设置主线程的优先级  方式二： 实现Runnable接口  优点：解决了java中单继承的特点  可以实现线程之间的数据共享  方式三：实现Callable接口  Join（）方法 可以使主线程进入阻塞状态，使其他线程继续执行，相当于插队实现。  进程是指正在运行的程序，当一个程序进入内存运行，即变成一个进程。  线程：是进程中的一个执行单元 |

|  |
| --- |
| 线程的5个状态  新建 调用start（） 变为就绪状态 获取CPU执行权 运行状态 sleep( ),join( ),wait( ),suspend( )变为阻塞状态 sleep时间到，join（）结束，notify（）/notifyAll( ) resume( ) 再变为就绪状态，当线程执行完或者调用stop（）就会死亡。  SetDaemon(true) 设置守护线程 当主线程结束自己设置的其他线程也会随之停止。  在那个线程里面设置 就守护的那个线程  守护线程的经典场景：JVM垃圾回收的线程就是一个守护线程。  Stop（）让当前线程停止  Yield（）礼让线程  线程安全问题  Synchronized 同步   1. 同步代码块 2. 同步代码函数 锁代表是this 不推荐使用synchronized 修饰run方法 3. 同步静态方法 锁是类的字节码对象，就是.getclass   同一个类的不同对象的类的字节码对象是一样的  读写操作：多线程情况下 读和读操作是共享的，写和写是互斥的，读和写都是操作互斥的。单单使用Lock锁是不行，那样读和读不是共享的。  ReadWriteLock（读写锁）才可以达到读和读之间是共享的。  线程通信的细节 基于synchronized的线程通信   1. 线程通信必须在锁情况下进行 2. 进行线程通信的多个线程 ，要使用同一个同步监视器（就是同一个锁）   还必须要调用该同步监视器的wait（），notify（），notifyAll（）  基于lock锁的线程通信问题 await（），signal（）  基于Locksupport下线程通信问题（解锁特定线程） park（），unpark（） 关键在于获得Thread对象  Main方法能重载  JUC java.util.concurrent包的简称，目的为了更好的支持高并发任务，让开发者利用这个包进行的多线程编程可以有效的减少竞争条件和死锁线程  CAS + volatile 就是lock的底层  CAS CompareAndSwap 比较并交换  关于CAS操作有个经典的ABA问题 解决方法：记录每次修改时间，或者记录每次修改的版本  Swith case 不加break的话，default里面的代码也会走  纯虚函数 就是java中的抽象方法  虚函数 就是java中的普通方法  JSP内置九大对象有：request，response，session，out，page,application,exception, pageContext, config  两个数值进行二元操作时，会有如下操作  如果两个操作数其中一个是double类型，另一个操作就会转换成double类型  否则，如果其中有一个操作数是float类型，另一个就会变成float类型  否则，如果其中一个操作数是long类型，另一个会转换成long类型  否则，两个操作数都换成int类型。  Java7 ConcurrentHashMap使用segment来分段和管理锁来保证安全，segment继承自ReentrantLock  Java8 ConcurrentHashMap使用Synchronized来CAS来操作  Servlet的生命周期分为5个阶段：加载，创建，初始化，处理客户请求，卸载   1. 加载：容器通过类加载使用servlet类对应的文件加载servlet 2. 创建：通过调用servlet构造函数创建一个servlet对象 3. 初始化 调用init方法初始化 4. 处理客户请求：每当有一个客户请求，容器会创建一个线程来处理客户请求 5. 卸载：调用destroy方法让servlet自己释放其占用的资源   Java的transient关键字的作用是需要实现Serilizable接口，将不需要序列化的属性前添加关键字transient，序列化对象的时候，这个属性就不会序列化到指定的目的地中。  线程私有：java虚拟机栈，程序计数器，本地方法栈  线程共享：java堆，方法区 |

|  |
| --- |
| Source一下 .一下（Oracle数据库可用），就是执行sql文件  配置文件命名必须是application  配置文件类型，可以是properties 或者 yml（yaml）  两种配置文件都存在的时候，优先读取properties  SpringBoot的配置文件存放位置   1. 项目根目录、config 2. 项目根目录 3. classpath（src/main/resources）:/config 4. classpath   filter和interceptor的区别  filter和interceptor技术层面不一样  filter是servlet的技术，interceptor是spring的技术  filter是过滤servlet层面上的，而servlet是javaweb开发最底层的技术，客户端发出的任意请求，过滤器都会处理。  Interceptor只能拦截spring容器内的，就是dispatchservlet才拦截  OutputStream实现了flushable接口 有flush方法  Final修饰是成员变量在赋值时可以有三种方式，   1. 在声明时候直接赋值 2. 在构造器赋值 3. 在初始代码块种进行赋值   ThreadLocalmap使用开放定址法解决冲突，hashmap使用链地址法解决hash冲突  Input是从文件读取进来，output是输出到文件  在java中，堆被分为两个不同的区域，新生代，老年代，  新生代分为 Eden，from survivor（幸存者） tosurvivor  Jvm每次只会使用Eden和其中一块Survivor区域来对对象服务，所以无论什么时候，总有一块survivor是空闲的 from和to之间会循环15次。  Native方法，native方法就是 不是用java代码写的方法  Byte+byte 是int类型的 所以要用int类型的变量来接  标识符定义由$ \_ 字母，数字组成 不能以数字开头  子类重写父类方法是，方法的访问权限不能小于原访问权限，在接口中，方法的默认权限是public 所以子类重写后只能是public  Hashtable 是一个散列表，它存储的内容是键值对  Hashtable的函数都是同步的，这意味着他是线程安全的，它的key和value都不可以是null。  HashTable直接使用对象的hashCode 继承了dictionary（字典）  HashMap 由数组+链表组成的，基于哈希表的map实现的，数组是Hashmap的主体，链表则是主要为了解决哈希冲突而存在的。 不是线程安全的，HashMap可以接受null的key和value HashMap重新计算hash值 继承了AbstractMap  面向对象的5个基本原则   1. 单一职责原则 2. 开放封闭原则 3. 里氏替换原则 4. 依赖倒置原则 5. 接口隔离原则   可以把局部内部类当作一个局部变量，所以他是不需要加任何修饰符的，局部内部类前不能用修饰符public 和private protected  内部类就随意了  所有对象创建在Eden区，当Eden区满后触发新生代的Minor GC  如果老年代和持久代满了 将会触发FullGC  堆分为新生代和老年代  方法区为永久代  JVM在判断两个class是否相同时，不仅判断两个类名是否相同，而且要判断是否由同一个类加载器实例加载的  Java中原子操作包括：   1. 除long和double之外的基本类型的赋值操作 2. 所有引用reference的赋值操作 3. java.concurrent.Atomic.\* 包中所有类的一切操作   byte<char<short<int<long<float<double 低级向高级是隐式转换，高级向低级必须是强制类型转换  B树和B+树的区别 ：B树叶子节点一个分出两个节点  B+树叶子节点一个只有一个节点 |